11-2017

From Quantum to Classical Interactions Between a Free Electron and a Surface

Peter Beierle

FROM QUANTUM TO CLASSICAL INTERACTIONS BETWEEN A FREE ELECTRON AND A SURFACE

by Peter James Beierle

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Physics and Astronomy

Under the Supervision of Professor Herman Batelaan

Lincoln, Nebraska

November, 2017

FROM QUANTUM TO CLASSICAL INTERACTIONS BETWEEN A FREE ELECTRON AND A SURFACE

Peter James Beierle, Ph.D.

University of Nebraska, 2017

Advisor: Herman Batelaan

Quantum theory is often cited as being one of the most empirically validated theories in terms of its predictive power and precision. These attributes have led to numerous scientific discoveries and technological advancements. However, the precise relationship between quantum and classical physics remains obscure. The prevailing description is known as decoherence theory, where classical physics emerges from a more general quantum theory through environmental interaction. Sometimes referred to as the decoherence program, it does not solve the quantum measurement problem. We believe experiments performed between the microscopic and macroscopic world may help finish the program. The following considers a free electron that interacts with a surface (the environment), providing a controlled decoherence mechanism.

There are non-decohering interactions to be examined and quantified before the weaker decohering effects are filtered out. In the first experiment, an electron beam passes over a surface that's illuminated by low-power laser light. This induces a surface charge redistribution causing the electron deflection. This phenomenon's parameters are investigated. This system can be well understood in terms of classical electrodynamics,

and the technological applications of this electron beam switch are considered. Such phenomena may mask decoherence effects.

A second experiment tests decoherence theory by introducing a nanofabricated diffraction grating before the surface. The electron undergoes diffraction through the grating, but as the electron passes over the surface it's predicted by various physical models that the electron will lose its wave interference property. Image charge based models, which predict a larger loss of contrast than what is observed, are falsified (despite experiencing an image charge force).

A theoretical study demonstrates how a loss of contrast may not be due to the irreversible process decoherence, but dephasing (a reversible process due to randomization of the wavefunction's phase). To resolve this ambiguity, a correlation function on an ensemble of diffraction patterns is analyzed after an electron undergoes either process in a path integral calculation. The diffraction pattern is successfully recovered for dephasing, but not for decoherence, thus verifying it as a potential tool in experimental studies to determine the nature of the observed process.

Preface

The experimental work "A Low-Power Optical Electron Switch" described in Chapter 3 has been published in *Journal of Physics D: Applied Physics*

The experimental work "Experimental Test of Decoherence Theory using Electron Matter Waves" described in Chapter 4 has been submitted for publication in *Physical Review Letters*.

The theoretical work "Spatial Correlation in Matter Wave Interference as a measure of Decoherence, Dephasing and Entropy" described in Chapter 5 has been submitted for publication in *Physical Review A*.

We appreciate funding support from the National Science Foundation and the Department of Education.

Acknowledgments

I would like to thank my research advisor Herman Batelaan for his graciousness in giving me the precious opportunity to work and learn under his mentorship whilst exploring and contributing to a subject that I find meaningful. I appreciate the patience, passion, and wisdom that you have delivered in this process.

I would like to thank the members of my Ph.D committee, Herman Batelaan, Kees Uiterwaal, Mathias Fuchs, Christian Binek, and Reina Hayaki. Your contribution in providing feedback was quite useful in improving the quality of this product.

Thank you to my group members Herman Batelaan, Roger Bach, Wayne Huang, Maria Becker, Eric Jones, Derek Ruffner, Maaneli Derakhshani, Liyun Zhang, Phillip Wiebe, Sam Kemerati, and Zilin Chen. The many instances of inciteful collaboration has been very valuable. A special thanks to Roger Bach and Wayne Huang for their contribution for their work in Chapter 3, as well as Liyun Zhang and Zilin Chen for their contribution to the work in Chapters 4 and 5.

Surface preparation and characterization was very important in this work. Thank you to those who provided me helpful advice and assistance; including Keith Foreman, George Peterson, Vijay Singh, Uday Singh, Stephen Ducharme, Kishan Sinha and Sy-Hwang Liou.

Thank you to my classmates and friends who fought alongside me to conquer the graduate school program. The endlessly long hours of studying and problem solving together was much appreciate

I would like to thank my previous teachers and mentors in science, particularly Linda Walters-Wallace, Gregory Wallace, Kimberly Skinner, and Xu Du. I would not be at this position without your early fostering of my curiosity and interest in physics, and opportunities that you granted me through your guidance.

Last, but not least, thank you to my wife Yao, my parents Peter and Debe, and my siblings Stephanie, Rebecca and Michael for your love and support and has been crucial to my success. I greatly appreciate your sacrifices, care and understanding.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation

Particle-wave duality is the bedrock of quantum theory, and can be most famously realized in the controlled electron double slit experiment, where an individual electron "interacts" with two neighboring slits (figure 1.1a, also known as a double slit), then travels to a phosphorous screen "detector" which lights up in an individual location indicating the position in which the electron has landed (the backstop detector in figure 1.1) [1]. If multiple electrons followed classical Newtonian motion, they would statistically form a pattern at the detector resembling that of the sum two smooth Gaussian-like distributions corresponding to the electron travelling through either the first or the second slit (figure 1.1b). However, successive iterations of such electron events result in a buildup of a histogram (figure 1.1 i-v) on the detection screen resembling an interference pattern (figure 1.1c). This is interpreted to mean that each electron originally behaved as a wave and the interaction is that of wave diffraction through both slits, thus constructive and destructive interference from both sources occurs. It is this behavior which prompts us to implement the Schrödinger equation to explain the dynamics, where the electron propagates from a superposition of two separate position states.

Yet when we detect or measure the electron at the phosphorous screen, we only ever observe the electron at one position, never in multiple locations or a continuum of locations (hence its particle behavior). It is not obvious why this would be the case given the prior description of the evolution of the quantum state. Given that one can compute a

nonzero probability of the electron being at two spatially separate positions at a detector from the electron's wave function, it would seem reasonable to expect to observe this; but this does not occur. Bohr indeed admits that, within the Copenhagen interpretation of quantum theory, a measurement of the state changes the state of the system which cannot be described by quantum mechanics [2]. Thus, there is an additional measurement axiom in which the wave function collapses to one or more states according to the Born Rule. But it is not clear under what conditions a "measurement" is said to occur. For example, a particle in a bubble chamber has presumably undergone measurement, as only single particle tracks are observed. On the other hand, should the same particle travel through free space, a dilute gas, or in an electromagnetic field, we may or may not defer to regarding to a quantum measurement collapse, or consider the system undergoing unitary evolution in quantum mechanics. The physical environment which the particle interacts with is not sufficient to a priori determine what type of evolution to invoke. These are the central issues behind what is known as "the quantum measurement problem" [3].

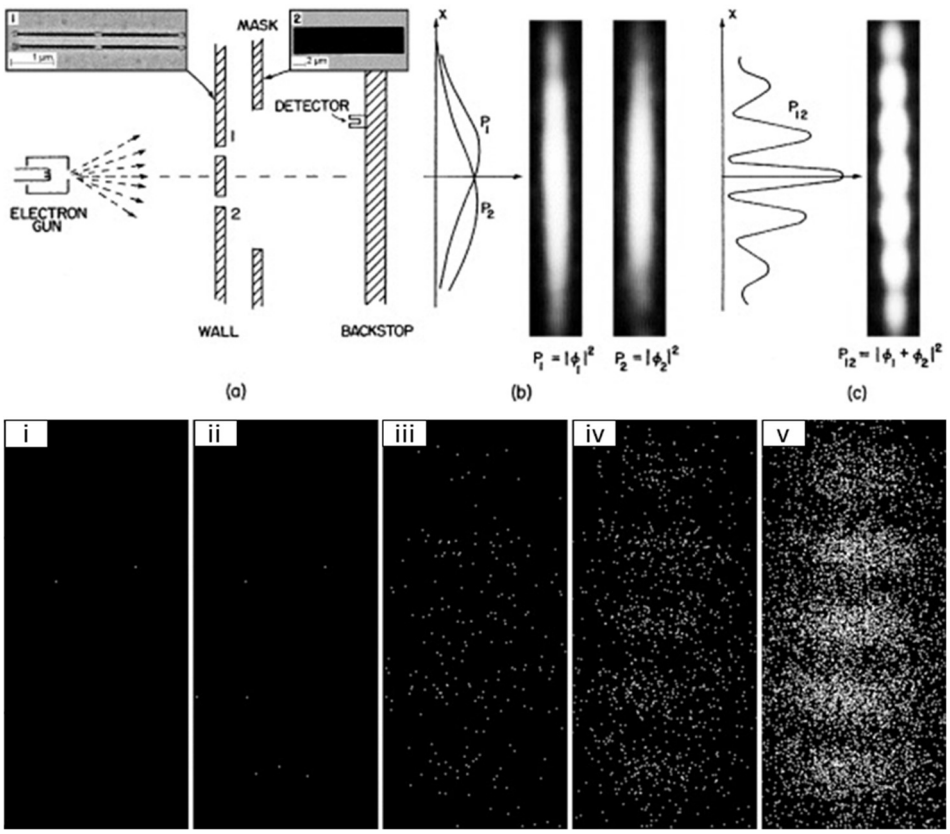

Figure 1.1: Double Slit Diffraction: Wave Interference vs Classical Distribution (images and experiment from Roger Bach et al. in [1]). An electron gun fires individual electrons through two slits (double slit (a)) and land on the backstop detector. After successive buildup of each electron at the detector (i-v) an interference pattern is formed (v and (c)) rather than the classical sum of travelling through the two individual slits (b). decoherence aims to predict how through quantum interaction (via entanglement) with an external environment an electron's motion can transition the observed probability distribution from quantum interference $|\phi_1 + \phi_2|^2$ (c) to more classical statistical behavior $|\phi_1|^2 + |\phi_2|^2$ (b).

To take the double-slit experiment further, Richard Feynman expanded the thought experiment to include a detector which monitors which slit the electron passes through [4]. With this addition, the distribution of electrons that is recorded at the detector is no longer an interference pattern, but the sum of two smooth Gaussian-like distributions. It seems that this detector disrupted the electron's behavior such that it has

transitioned from quantum to classical mechanics. The theory of decoherence sets out to describe why and how this transition occurs.

Taking an emergent point of view, it seems reasonable to claim that all of classical theory must be a subset of quantum theory, and is just a special case (as Galilean classical relativity can be thought of as the special case of Einsteinian special relativity in the limit of $v/c \ll 1$). Similarly, for matter if you take the de Broglie Wavelength of an object $\lambda_{dB} = h/(mv)$ in the limit of large mass, then the wavelength would resultantly be exceedingly small. Richard Feynman eloquently illustrates this concept in a lecture contained in the book Six Easy Pieces [5], where one imagines exchanging the electrons in the double-slit experiment with bullets (see Figure 1.2). He argued that if were one to fire these coherent bullets through the double slit, then the far field histogram pattern one might hope to observe would be in principle that of an interference pattern, but because of the very small wavelength of these bullets, the distance between of such interference fringes would be so small that in practice no detector could hope to resolve them. Thus, the resulting pattern would be a smeared distribution. It is in this way it is argued that the Copenhagen interpretation remains consistent.

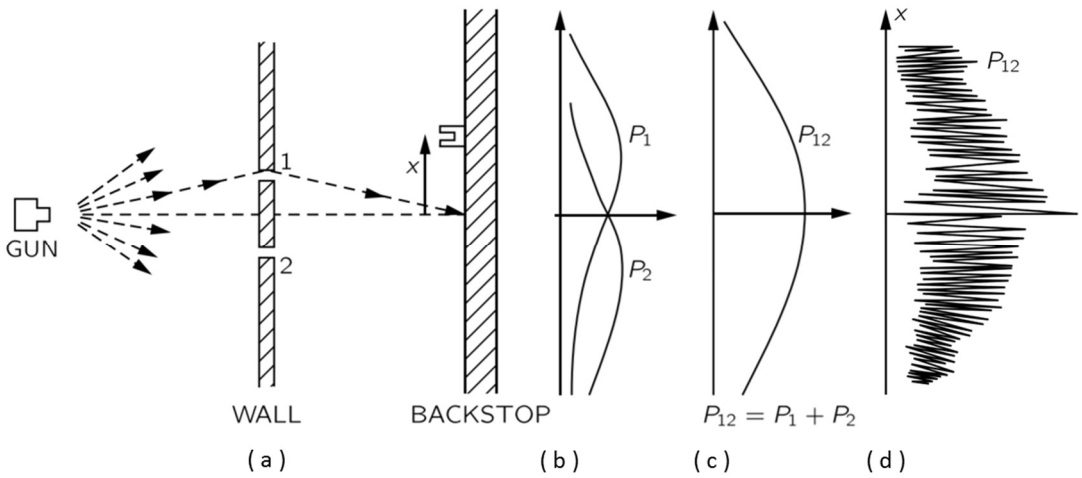

Figure 1.2: Double Slit Experiment With Classical Bullets (from Feynman's Six Easy Pieces [5]). According to the Copenhagen interpretation of quantum mechanics, because the deBroglie wavelength of a high momentum object (like a classical bullet) is so small, any interference pattern produced in a macroscopic double-slit like experiment would result in a fringe periodicity too small to be resolved (d) and as a result the final observed distribution will be smeared out (c).

Is this a sufficient story however? Does this imply that if the momentum of the object is sufficiently small, as in a slow electron, then it will always exhibit quantum behavior such as the superposition principle? Are there no other factors to consider except for momentum?

The Decoherence Program starts with ordinary quantum theory and has found that a quantum system coupled (i.e. entangled) by even a small amount with an environment can result in a large transfer of the wave's phase information from the system to the environment. This loss results in the loss of coherence between different orthogonal states of the wave function, which is precisely what allows them to interfere in the first place. Thus if such an interaction is permitted to occur comparable to the timescale called the decoherence time (also known as the decorrelation time) [6], then interference is not observed; and the system reduces to classical behavior. For the case of the electron and

the double slit, this theory specifies how the statistical distribution of the electron can be changed from the interference pattern (figure 1.1 c) to the addition of two smooth distributions (figure 1.1 b). Symbolically, this means;

$$\left|\phi_1+\phi_2\right|^2 \xrightarrow{\text{decoherence}} \left|\phi_1\right|^2+\left|\phi_2\right|^2. \tag{1.1}$$

One ought to ask how long it takes for a wave packet to reduce to a single position state. This is particularly of interest in the classical world of macroscopic parameters, such as the prior double slit bullet example. To take a similar scenario, Wojciech H. Zurek [7] considered a free particle of mass 1 g at room temperature that is coherently split into a superposition of two distinct positions that is separated by a distance $\Delta x = 1\ cm$. The state is considered weakly coupled to an environment (or bath) of quantum harmonic oscillators. For such a case, the decorrelation timescale θ of this spatial superposition can be compared to the timescale τ of energy dissipation from the particle to the environment by $\theta = \tau\left[\left(\hbar/\sqrt{4mkT}\right)/\Delta x\right]^2$.

Substituting in the aforementioned macroscopic parameters yields a ratio of these two timescales of $\theta/\tau \sim 10^{-40}$. For reference, this means that even if one could ideally build an experimentally isolated system such that the timescale in which energy dissipates from the particle to the outside environment is that of the age of the universe ($\tau \sim 5\times10^{17}\,s$), then the spatial superposition of our 1 g particle would still very quickly decay on the timescale of $\theta \sim 5\times10^{-23}\,s$. This is, for comparison, of the order of the typical strong nuclear interaction [8]. This provides some explanation for why we don't see the such bizarre quantum phenomena in the ordinary world, and why we need not

worry if Schrödinger's torturous experiments can put our feline friends in a superposition limbo.

The goal of the decoherence program is to not only explain why macroscopic objects do not exist in quantum superposition within the framework of quantum theory, but also how microscopic objects such as electrons can lose their quantum behavior. Can environmental interactions described by decoherence make predictions about the loss of "quantumness" of the electron (by, for example, controlling the strength of this environmental interaction)?

1.2 Summary of Chapters

The following chapters feature three projects related to the coherent electron beam interaction with conducting and semiconducting surfaces. Chapter 2 provides an extensive introduction to the theoretical background and tools this work is grounded upon. This includes well established concepts such as image charge potentials which we use to understand our data, as well as competing decoherence models we sought out to directly test. Chapter 3 details the development and characterization of an electron beam switch which is controlled by the field produced by a surface charge distribution produced by a low-power optical laser. Chapter 4 involves the experimental test of decoherence theories by bringing gold and *n*-doped silicon surfaces near an electron beam undergoing diffraction through a nanofabricated grating. Chapter 5 presents a theoretical investigation of the differences between decoherence and dephasing in matter wave propagation (such as electron waves). It is also shown how it is possible to distinguish between dephasing and decoherence processes that may distort the

interference pattern in the far field. Finally, Chapter 6 gives concluding remarks regarding this work and an outlook on what opportunities this work leads to.

1.2.1 Electron-Optical Switch

A beam of 4 *keV* electrons passes by a metallic surface, which is illuminated by a low-power continuous laser beam (typically 10 *mW* and 658 *nm*) [9,10]. These electrons experience a force that deflects the beam's direction by 550 μrad when the electrons are approx. 10 μm from the surface. This "electron switch" has a response time of approximately 6 μs. The deflection of the electron beam is shown to decrease as the beam's distance from the wall increases, giving an observed electron deflection as far as 200 μm from the surface. This switching mechanism is shown to be robust, as it is demonstrated for various optical wavelengths and surfaces. This type of electronic-free electron manipulation has potential use in electron beam microscopy (EBM) and electron beam lithography (EBL).

1.2.2 Decoherence Experiment

A controlled decoherence environment is studied experimentally by free electrons interacting with semi-conducting and metallic plates. The results are compared with physical models applied to decoherence theory to describe the quantum-classical transition. The experiment is consistent with decoherence theory and rules out established Coulomb interaction models in favor of a plasmonic excitation model. In contrast to previous decoherence experiments the present experiment is sensitive to the onset of decoherence.

1.2.3 Dephasing vs Decoherence

In this theoretical work, we study the loss of contrast in double-slit electron-diffraction. We show how the spatial autocorrelation spectrum of the far field intensity distribution can be used to distinguish between a loss of contrast caused by dephasing or decoherence processes. This establishes a measure of time-reversibility that does not require the determination of coherence terms of the density matrix (correlations between spatial states). This contrasts with entropy, another measure of time-reversibility, that does require the coherence terms. This spatial autocorrelation technique is promising, taking into consideration the need to diminish the detrimental experimental effect of loss of contrast, identifying what kind of processes or environments cause irreversible damage to interference and which can be reconstructed, and for fundamental studies regarding the transition from the classical to the quantum regime.

1.3 Chapter 1 Bibliography

[1] R. Bach, D. Pope, S.-H. Liou, and H. Batelaan, New J. Phys. **15**, 033018 (2013).
[2] S. Weinberg, Camb. Core (2015).
[3] M. Schlosshauer, Rev. Mod. Phys. **76**, 1267 (2005).
[4] R. P. Feynman, R. B. Leighton, and M. Sands, in *Feynman Lect. Phys.* (Addison-Wesley, Reading, MA, 1965), p. Chapter 1.
[5] R. P. Feynman, R. B. Leighton, and M. Sands, *Six Easy Pieces: Essentials of Physics Explained by Its Most Brilliant Teacher*, 4, illustrated, reprint ed. (Basic Books, 2011, 1985).
[6] W. H. Zurek, Rev. Mod. Phys. **75**, 715 (2003).
[7] W. H. Zurek, in *Front. Nonequilibrium Stat. Phys.*, edited by G. T. Moore and M. O. Scully (Springer US, 1986), pp. 145–149.
[8] D. Griffiths, in *Introd. Elem. Part.*, 2nd ed. (Wiley-VCH, Weinheim, 2010), pp. 33–34.
[9] W. C.-W. Huang, R. Bach, P. Beierle, and H. Batelaan, J. Phys. Appl. Phys. **47**, 085102 (2014).
[10] P. Beierle, W. Huang, R. Bach, M. Becker, D. Ruffner, and H. Batelaan, in (APS Division of Atomic, Molecular and Optical Physics Meeting, 2014).

CHAPTER 2

THEORY

2.1 Introduction

Outlined in this chapter are the models used to test the experimental and theoretical work completed in Chapters 3-5. The Chapter also provides some details about the underlying theories and concepts these models make use of. This begins in Section 2 with classical electron propagation both in free space and over a conducting surface, including effects born from classical image charge and classical surface charge induced by light (the latter pertaining to Chapter 3). Section 3 introduces quantum propagation of electrons, in particular the electron's wave propagation and the interference which results. In Section 4 the concepts of wave coherence of light and electrons are summarized. Section 5 is an introduction to the theory of decoherence, which aims to bridge the gap between classical and quantum mechanics by attempting to arrive at classical results entirely within a quantum framework. Section 6 provides various physical models which have been developed by others to predict decoherence effects due to a free electron propagating over a conducting surface (which we test in the experiment in Chapter 4). Finally, Section 7 gives an introductory comparison between decoherence and dephasing, and examines the concepts and measures (i.e. spatial autocorrelation and entropy) associated with these two phenomena. This is the subject of the theoretical work of Chapter 5.

2.2 Classical Electron Propagation over a Conducting Surface

Classical motion of non-relativistic electrons which propagate in free space as well as interacting in an electric field can be well approximated using Newtonian mechanics and classical electrodynamics [1]. Importantly, in classical mechanics the motion of an electron is well defined in terms of definite position $\vec{x}(t)$ and definite velocity $\vec{v}(t)$ at any given time. When it comes to electrons interacting with a surface, the term "free" is used to indicate that the electron approaches a surface (perhaps from a different source) and then moves away from the surface until it experiences a negligible or no force due to its field. This contrasts with electrons that may collide with the surface and remain bound to it, or electrons that are emitted from the surface (which are the subject of phenomena such as thermionic electron emission and inverse photoemission). Also not considered are electrons which make contact and rebound from the surface (contact collisions).

The source of the electromagnetic interactions that are taken into consideration from the surface include those surface charge distributions which most simply model the change in momentum the electron experiences perpendicular to the plane of the surface in the later experiments. These surface charge distributions we consider, attributed to maintaining zero electric field inside the conductor, include 1) image charge and 2) the charges which produce the ponderomotive potential induced by laser light effects.

2.2.1 Classical Image Charge

In the classical case, the electric force a point charge experiences in vacuum at a distance *d* away from a boundary separating vacuum from a conducting surface is a

convenient electrostatics problem to solve, and the symmetry of the problem simplifies it further. Consider that in static case, one would have to solve for an electron with charge -*e* located at $\vec{r}$ experiencing a potential $\Phi(\vec{r})$ due to a charge distribution with charge density $\rho(\vec{r}')$ [2],

$$\vec{F}(\vec{r}) = -e\vec{\nabla}\Phi(\vec{r}) = -\frac{|e|}{4\pi\varepsilon_0}\vec{\nabla}\int\frac{\rho(\vec{r}')}{|\vec{r}-\vec{r}'|}d^3r' . \tag{2.1}$$

Because of assumed rotational symmetry about the perpendicular line crossing through the electron, as well as translational symmetries in the *x* & *z* directions and the reflection symmetries in the *x*-*y* and *z*-*y* planes (see figure 1), and ignoring fringe fields of the finite surface (assume *y* is much smaller than the size of the surface) as well as skin-depth effects, this simplifies to

$$\vec{F}(y) = -\frac{e^2}{4\pi\varepsilon_0}\frac{\hat{y}}{(2y)^2} . \tag{2.2}$$

Note that this form is that of an electron experiencing a coulomb force due to a positive charge +|e| at a distance 2*y* away from the electron, thus this surface charge has is called an "image" or "mirror" charge.

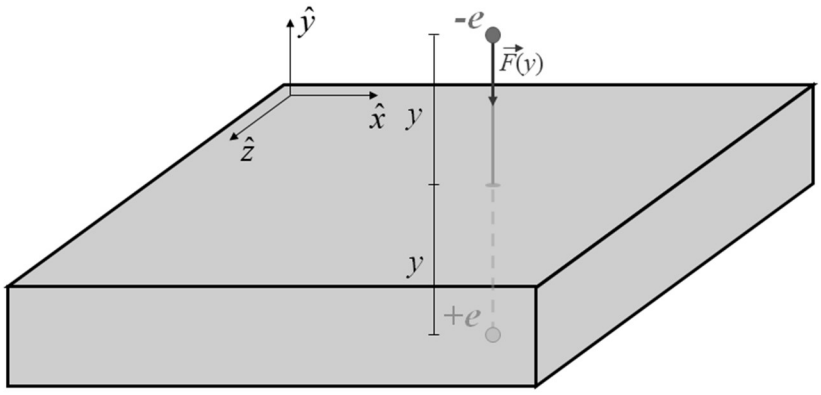

Figure 2.1 Image Charge Configuration

However, there is a slight reduction of this force for the case of the non-perfect conductor with a static dielectric constant $\varepsilon = \varepsilon_0 \varepsilon_r$. In this case, the image charge's strength reduces to

$$q' = +|e|\frac{(\varepsilon - \varepsilon_0)}{(\varepsilon + \varepsilon_0)}. \tag{2.3}$$

If one takes as an example silicon whose relative dielectric constant $\varepsilon_r \approx 11.7$, then $q' = .843|e|$. Thus, the force is modified to become

$$\vec{F}(y) = -\frac{eq'}{4\pi\varepsilon_0}\frac{\hat{y}}{(2y)^2} = -\frac{e^2}{4\pi\varepsilon_0}\frac{(\varepsilon - \varepsilon_0)}{(\varepsilon + \varepsilon_0)}\frac{\hat{y}}{(2y)^2}\ . \tag{2.4}$$

See Chapter 4.5 for details on how this is implemented in modelling the deflection the electron experiences.

2.2.2 Deflection from Induced Light Potential

A surface charge distribution model was also produced by Wayne Huang to predict the force in the developed electron switch described in Chapter 3 [3]. When a laser light is incident perpendicular to a surface, a thin surface layer ($\delta \approx 1\ nm$) of electrons can be redistributed (again as a result of maintaining zero electric field inside the material). This lateral force on these surface electrons is

$$\vec{F}_p(x,z) = -\frac{e^2\lambda^2}{8\pi^2 m_e c^3 \varepsilon_0}\vec{\nabla} I(x,z), \tag{2.5}$$

where λ is the wavelength of the laser, and $I(x,z)$ is the intensity distribution of the focused laser on the surface. Assuming a that this force induces a dipole moment distribution, and thus produces a surface charge distribution,

$$\sigma(x,z) = \left(-\vec{\nabla}\cdot\vec{P}\right)\delta = -\frac{1}{\alpha}\frac{n_0 e^3 \lambda^2 \delta}{8\pi^2 m_e c^3 \varepsilon_0}\nabla^2 I(x,z), \tag{2.6}$$

where α is a fitting parameter in the linear dipole approximation and n_0 is the free electron density of the material. The electron travelling over the surface then experiences a force depending upon its *x*-*z* position on the surface as

$$\vec{F}(x,z) \approx \frac{e\sigma(x,z)}{2}\hat{y}. \tag{2.7}$$

Note that the approximation is used that one is near the surface such that the propagating electron experiences the local surface charge (thus the distance dependence was not worked out). Also, note that the force changes as a function of the path the electron travels along the surface (so much so that the deflection direction can change due to the Laplacian of the laser's intensity spatial distribution on the surface $\nabla^2 I(x,z)$ changing sign). This is in direct contrast with the modelling of image charge, where it is assumed that the force of the electron over the surface is the same everywhere over the surface and that the magnitude of the force depends on the distance to the surface in the *y* direction.

2.3 Quantum Electron Propagation in Free Space

While there are many situations where it is appropriate to approximate the motion of an electron using classical physics, electrons in the right contexts also demonstrate

properties that are not associated with classical particle motion. This includes phenomena such as wave interference (via the superposition principle), quantized states such as quantized angular momentum states when bound in a hydrogen atom, and electron degeneracy arising from fermionic statistics [4].

As an initial framework to describe quantum electron propagation in free space as it is ordinarily introduced, we begin with the emphasis of some of the postulates of quantum mechanics. For a more comprehensive formulation of all of these postulates, see Cohen-Tannoudjii et al [5]. The first postulate is stated as,

First Postulate: At a fixed time t_0*, the state of a physical system is defined by specifying a ket* $|\psi(t_0)\rangle$ *belonging to the state space* $\mathscr{E}$.

The physical system in this case (and throughout this thesis) is the free election. The state space $\mathscr{E}$ is a complex Hilbert space where the included states are unit vectors (i.e. it is a type of complex vector space). Importantly, because the state is described in terms of a complex vector, it follows that a linear combination of states within $\mathscr{E}$ is itself a state within $\mathscr{E}$,

$$|\Psi\rangle = \sum_i \alpha_i |\psi_i\rangle . \tag{2.8}$$

It is this feature of quantum states which is known as the superposition principle.

The sixth postulate according to Cohen-Tannoudjii et al. is the foundation of the propagation, or time evolution of the quantum state,

Sixth Postulate: The time evolution of the state vector $|\psi(t)\rangle$ *is governed by the Schrodinger equation:*

$$i\hbar\frac{\partial}{\partial t}\left|\psi\left(t\right)\right\rangle = H\left(t\right)\left|\psi\left(t\right)\right\rangle \tag{2.9}$$

where $H\left(t\right)$ *[the Hamiltonian] is the observable associated with the total energy of the system.*

The combination of the unitary evolution of the quantum state described by the Schrodinger equation and the superposition principle is what leads to the wave properties of the quantum state of the electron.

Consider now an electron wave that is split into two states $\left|\psi_1\left(x,t\right)\right\rangle$ and $\left|\psi_2\left(x,t\right)\right\rangle$ (see Figure 2) where *x* corresponds to the coordinate position in the transverse direction of propagation. After wave propagation of these separate states, assume that the evolution was prepared such that it leads to recombination at a detection screen (at the detection screen, there is at least some spatial overlap of these two states). One will observe the electron landing at a specific position *x* on the detector determined by the probability distribution in terms of the wavefunction in position representation,

$$P\left(x\right) = I\left(x\right) \propto \left|\psi_1\left(x\right)+\psi_2\left(x\right)\right|^2 = \left|\psi_1\left(x\right)\right|^2 + \left|\psi_2\left(x\right)\right|^2 + 2\,\mathrm{Re}\left(\psi_1\left(x\right)\psi_2^*\left(x\right)\right) \tag{2.10}$$

where the first two terms correspond to the probabilities of the two separate wave functions alone, and the third mixed term provides the interference. At the position *x* when $2\,\mathrm{Re}\left(\psi_1\left(x\right)\psi_2^*\left(x\right)\right) = -\left(\left|\psi_1\left(x\right)\right|^2 + \left|\psi_2\left(x\right)\right|^2\right)$, total destructive interference takes place and the probability of the electron landing at that position is zero. At the position *x* when $2\,\mathrm{Re}\left(\psi_1\left(x\right)\psi_2^*\left(x\right)\right) = +\left(\left|\psi_1\left(x\right)\right|^2 + \left|\psi_2\left(x\right)\right|^2\right)$, total constructive interference takes

place and the probability of the electron landing at that position is maximized for that position.

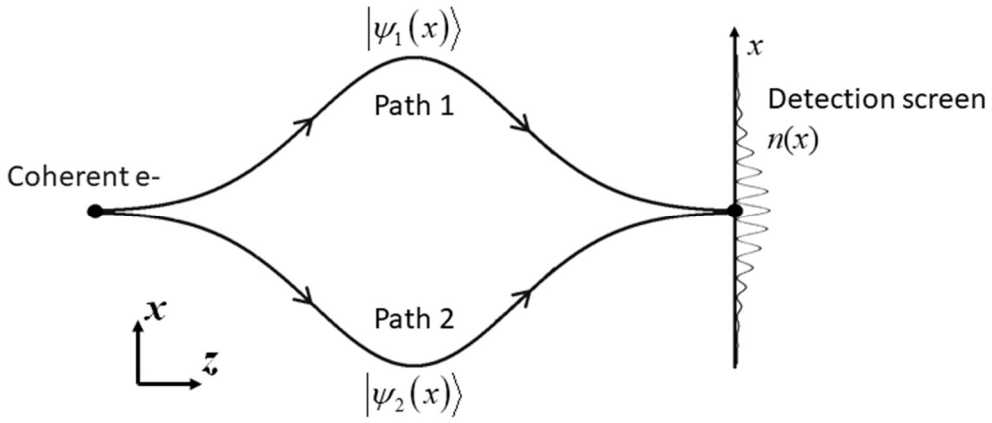

Figure 2.2 Coherent Interference between Two States.

2.4 Coherence of Matter Waves

In the previous section, it was shown how matter wave propagation leads to quantum interference. If the state of the electron can be described in this way, then it is said that the electron state (or the wave function) is coherent. Here, by coherent we informally define coherence as the electron state to undergo "full" interference (total constructive interference, or total destructive interference is possible). This does not necessarily need to be the case, and the following sections described the consequences of the case when the wave function is partially coherent.

2.4.1 Partial Coherence and Density Matrix Formalism

To describe a partially coherent (or mixed) quantum state, the density matrix (also known as a density operator [6]) formalism is a convenient mathematical tool to do so, as the previously described convention is insufficient to do so. Starting with the simple example of a fully coherent state as in equation 2.8, the density matrix of this state can be written as

$$\rho = \left|\Psi\right\rangle\left\langle\Psi\right|. \tag{2.11}$$

If the state is constructed from a wave function (e.g. equation 2.8), the state is said to be pure and $\rho^2 = \rho$. If a quantum state is pure, then it is also fully coherent.

A mixed state can be constructed by statistically adding pure states together as a convex sum,

$$\rho^{mixed} = \sum_k p_k \rho_k^{pure} = \sum_k p_k \left|\Psi\right\rangle_k \left\langle\Psi\right|_k, \tag{2.12}$$

with the convex condition $\sum_k p_k = 1$. Such a statistical mixture may evolve in a unitary manner according to the von Neumann equation (which can be deduced from the Schrodinger equation and vice versa [6]):

$$\left[\hat{H}, \rho\right] = i\hbar \frac{\partial}{\partial t}\rho. \tag{2.13}$$

If the density matrix is written in the position representation (in the *x*-direction) then the final spatial probability distribution can be found by taking the (main) diagonal of the density matrix;

$$P(x) = \left\langle x\right|\rho\left|x\right\rangle = diag\left[\rho(x)\right]. \tag{2.14}$$

2.4.2 Relationship between Coherence and Observed Interference

The nature of coherence can be best understood by its relation to observed interference. Starting again as in section 2.3 with an initial electron wave split into two states, $|\psi_1\rangle$ and $|\psi_2\rangle$, and evolves along separate paths as shown in Figure 2. This time however, the evolution somehow corrupts the two states (examples, such as density matrix state reduction, will be described in sections 2.5-2.7). For the two state case, analogous to equation 2.10, the observed intensity distribution observed in the far field is proportional to [7],

$$I = |\psi_1|^2 + |\psi_2|^2 + 2V\,\mathrm{Re}\left(\psi_1\psi_2^* + \mathrm{H.O.T.}\right), \tag{2.15}$$

where H.O.T. corresponds to higher order terms, and V corresponds to the visibility of the interference pattern:

$$V = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}}. \tag{2.16}$$

$I_{\max}$ is the maximum envelope of the intensity distribution and $I_{\min}$ corresponds to the minimum envelope that develops as a result of loss of contrast in interference (see Figure 3). Thus, if $I_{\min} = 0$, then $V = 1$, and the final state is fully coherent. If $I_{\min} = I_{\max}$, then $V = 0$ and the state may be fully incoherent. If $0 < I_{\min} < I_{\max}$, then $0 < V < 1$ and the final state may be partially coherent. Thus, visibility is often used as a measure of coherence.

An example diffraction pattern that models such a case is double slit diffraction, with a far field intensity distribution,

$$I = I_0\left(\frac{\sin(\gamma)}{\gamma}\right)^2 \frac{\left(1+V\cos(2\alpha)\right)}{2}, \tag{2.17}$$

where I_0 is the central intensity value for V=1 and x=0, $\gamma = \frac{\pi c}{\lambda}\sin(\theta)$ and $\alpha = \frac{\pi a}{\lambda}\sin(\theta)$. c is the width of the slit, a is the distance between slits, λ is the de-Broglie wavelength, and θ is the angle with respect to the normal of the double slit plane. In the small angle approximation at a distance z away from the surface (see Figure 3);

$$I(x) = \frac{I_0}{2}\left(\sin\left(\frac{\pi c x}{\lambda z}\right)\Big/\frac{\pi c x}{\lambda z}\right)^2\left(1+V\cos\left(\frac{2\pi a x}{\lambda z}\right)\right), \tag{2.18}$$

In the limit $V \to 1$, I approaches the ordinary double-slit Fraunhofer equation:

$$I = I_0\left(\frac{\sin(\gamma)}{\gamma}\right)^2 \cos^2(\alpha) \tag{2.19}$$

and in the limit $V \to 0$, it approaches the single slit Fraunhofer equation:

$$I = \frac{I_0}{2}\left(\frac{\sin(\gamma)}{\gamma}\right)^2. \tag{2.20}$$

Thus, a 2-path experiment (such as a double-slit experiment or a 2-path interferometer), visibility is a useful measure of the loss of coherence in a system as measured at the detector.

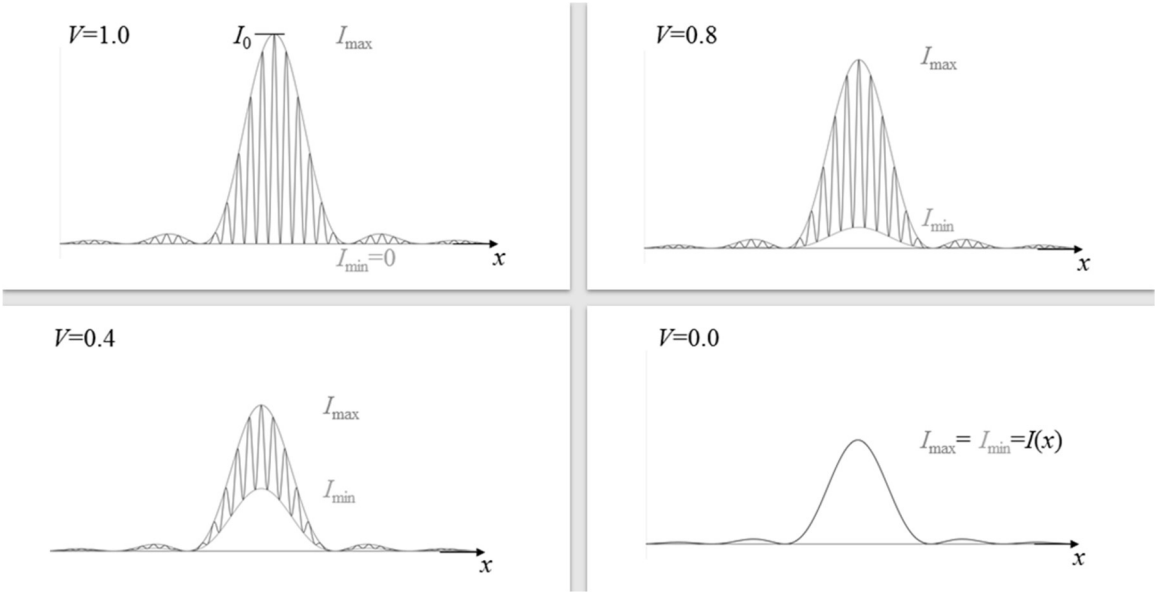

Figure 2.3 Far Field Double-slit Interference Pattern for Various Visibility Values. If V=1, then the interfering waves are fully coherent. See equation (2.17). If 0<V<1, then interfering waves are partially coherent with respect to each other. If $V=0$, then there is no interference between interfering waves (bottom left).

2.4.3 Coherence Length for a Collimated Matter Wave

For the case of a matter wave (such as atoms, electrons, etc) that is collimated (using for example collimation slits, see Figure 4), the coherence length can be computed starting with the Heisenberg uncertainty relation:

$$\Delta x \Delta p_x \sim h\,, \tag{2.21}$$

where we take the uncertainty in position Δx to be the transverse coherence length, and Δp_x corresponds to uncertainty in the beam's transverse momentum. Taking into consideration the geometry of beam collimation in the small angle limit:

$$\frac{\Delta p_x}{p_z} = \tan\left(\theta_{coll}\right) \approx \theta_{coll}\,, \tag{2.22}$$

where θ_{coll} correspond to the beam's divergence angle. This can be substituted into the uncertainty relation:

$$\Delta x \cdot p_z \cdot \theta_{coll} = h\,. \tag{2.23}$$

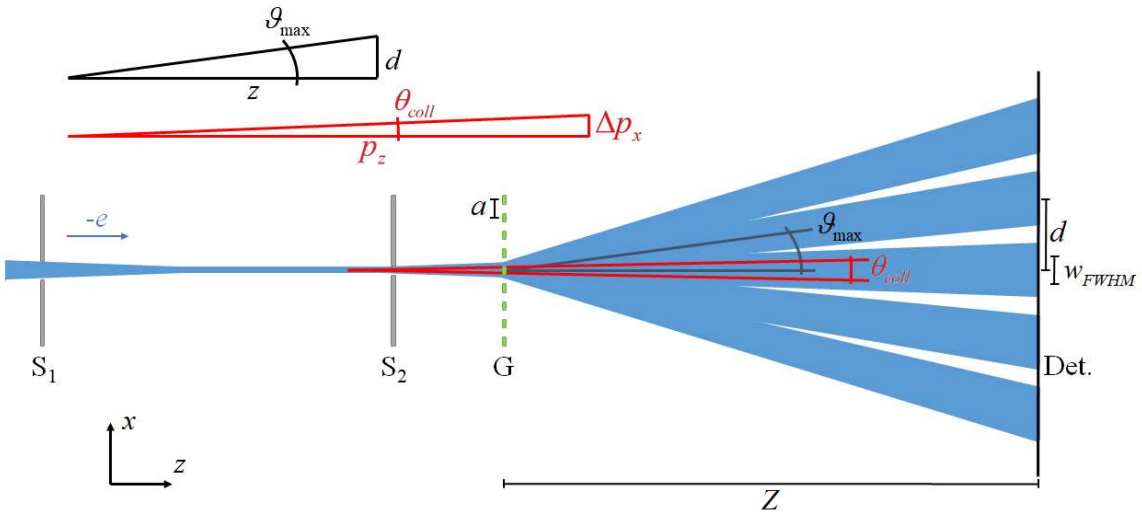

Figure 2.4: Schematic of Collimated Diffraction. The divergence of the electron beam θ_{coll} depends on the geometry of two collimation slits (S_1 and S_2). This ultimately determines the initial width w_{FWHM} of the diffraction peaks. The diffraction angle ϑ_{max} depends on nanofabricated diffraction Grating's (G) periodicity a along with the electron's wavelength. This determines the periodicity d of the observed diffraction at the detector through the diffraction equation (equation 2.27). These parameters are what influence the measure of the initial transverse coherence length (equation 2.30).

Using the definition of the De Broglie wavelength:

$$\lambda_{dB} = \frac{h}{p}, \tag{2.24}$$

the transverse coherence length becomes, in agreement with [8],

$$\Delta x \equiv L_{coh} \approx \frac{\lambda_{dB}}{\theta_{coll}}\,. \tag{2.25}$$

To calculate this transverse coherence length in terms of measurable quantities, a diffraction grating can be introduced after the beam has been collimated. For grating diffraction, the well-known diffraction equation is invoked,

$$a \cdot \sin\left(\vartheta_{\max}\right) = n\lambda\,, \tag{2.26}$$

where a is the periodicity of the grating, n corresponds to the n^{th} order diffraction peak, and $\vartheta_{\max}$ corresponds to the angle between the incident beam and the direction which the far field diffraction peaks propagate. This can be used to determine the distance d between the diffraction peak maxima in the far field in the small diffraction angle approximation;

$$\sin\left(\vartheta_{\max}\right) \approx \vartheta_{\max} \approx \tan\left(\vartheta_{\max}\right) = \frac{d}{z}\,, \tag{2.27}$$

where z is the distance between the grating and the plane in which diffraction peaks that are detected.

Equations (2.26) and (2.27) can be combined for a new expression of the De Broglie wavelength:

$$\lambda_{dB} = \frac{ad}{z}\ . \tag{2.28}$$

Similarly, the collimation angle can be determined in the small angle approximation by the width of the diffraction peaks w_{FWHM} at a distance z away from the grating;

$$\theta_{coll} \approx \tan\left(\theta_{coll}\right) \approx \frac{w_{FWHM}}{z}\,. \tag{2.29}$$

Finally, the calculation of transverse coherence length of the electron at the grating can be determined entirely in terms of measurable quantities observed at the detector by substituting in equations (2.28) and (2.29) into (2.25):

$$L_{coh} = \frac{ad}{w_{FWHM}}. \tag{2.30}$$

Thus, a loss of coherence is associated with a widening of the width of the diffraction peaks w_{FWHM} rather than a loss of contrast. Notice that this measure is independent of the distance *z* between the grating and the detector (provided that the detector is in the Fraunhofer diffraction region), and that only the ratio of the peak-to-peak distance to the width of the peaks matters. We can therefore measure coherence in terms of the transverse coherence length of the diffracted beam as observed at the detector.

2.4.4 Comparison of Coherence Measures (Transverse Coherence Length and Visibility)

To illustrate the effects of irreducible background on the measurement of visibility, let's first take the simple case of a detected interference pattern that is in a spatial range $0 < x < L$ perfectly sinusoidal with no irreducible background. Such an intensity distribution can be modeled as:

$$I_{clean}(x) = \frac{I_{total}}{L}\left(1 + V\cos\left(n\pi x\right)\right), \tag{2.31}$$

where I_{total} is the total integrated intensity in $0 < x < L$ of the particles of interest, $0 < V < 1$ describes the visibility associated with dephasing or decoherence one is trying to measure, and *n* controls the observed periodicity of the intensity pattern. In such a case with a maximum intensity of $\max\left[I_{clean}(x)\right] = \frac{I_{total}}{L}(1+V)$, and a minimum intensity of $\min\left[I_{clean}(x)\right] = \frac{I_{total}}{L}(1-V)$, then using equation (2.16) the computed visibility becomes by design

$$V_{clean} = \frac{\frac{I_{total}}{L}(1+V) - \frac{I_{total}}{L}(1-V)}{\frac{I_{total}}{L}(1+V) + \frac{I_{total}}{L}(1-V)} = V\,. \tag{2.32}$$

Now we introduce to this intensity a constant *irreducible background* with total intensity *Ld*, then the interference intensity distribution is modified to become

$$I_{clean+bckd}(x) = \frac{I_{total}}{L}\left(1 + V\cos(n\pi x)\right) + d\,. \tag{2.33}$$

With a new maximum intensity of $\max\left[I_{clean+bckd}(x)\right] = \frac{I_{total}}{L}(1+V) + d$ and a minimum intensity of $\min\left[I_{clean+bckd}(x)\right] = \frac{I_{total}}{L}(1-V) + d$, once more using equation (2.16) the visibility in this case is

$$V_{clean+bckd} = \frac{I_{total}V}{I_{total} + Ld}\,. \tag{2.34}$$

Note that in the limit $d \to 0$; $V_{clean+bckd} \to V_{clean}$. With a combined total intensity of $I_{total} + Ld$, the fraction of the intensity that is background is

$$f = \frac{Ld}{I_{total} + Ld};\ f \in [0,1);\ \tag{2.35}$$

thus the total background in terms of its fraction to the combined total intensity is

$$Ld = \frac{I_{total} f}{1-f}. \tag{2.36}$$

Finally, substituting this result into equation (2.34) the total visibility as a function of percent irreducible background is

$$V_{clean+bckd}(f) = V_{clean}\left(\frac{1}{1+\frac{f}{1-f}}\right) = V_{clean}(1-f), \tag{2.37}$$

which is plotted in Figure 5. Note how this differs from when calculating the transverse coherence length, in this case subtracting the irreducible background does not change the calculation of the coherence length, because $L_{coh}(f) = const.$

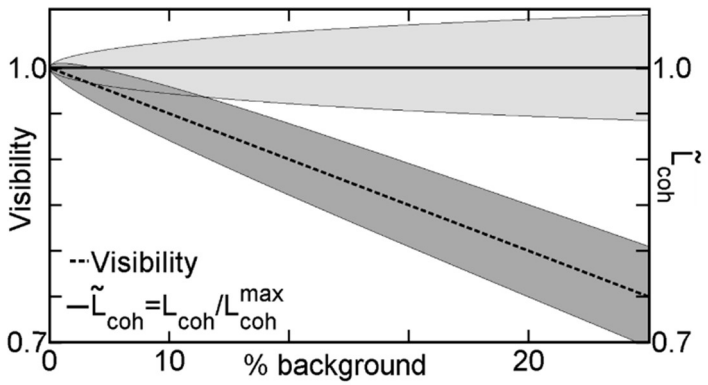

Figure 2.5. Comparison between visibility (V) and normalized transverse coherence length ($\tilde{L}_{coh} = L_{coh}/L_{coh}^{\max}$). For interferometry, the visibility is used to place a bound on decoherence. For diffraction, the decoherence measure is coherence length. The advantage of using diffraction rather than interferometry is that the decoherence measure is not background dependent. In other words, the linear drop of visibility in an interferometer (dashed line) due to a weak background signal masks the decoherence. This makes diffraction well-suited to search for weak decoherence. The shaded areas correspond to uncertainty due to the statistical error introduced by the background.

This illuminates the advantage of using diffractometry (as described in Chapter 4) over interferometry, which lies in their respective decoherence measures, L_{coh} and V

(Figure. 5). The background signal can be subtracted for diffraction without distorting the measured value of L_{coh}. This is not the case when measuring visibility in an interferometer. The visibility V drops off linearly due to a weak background signal, which can mask decoherence. For a weak decoherer that scatters the incident beam and introduces background, diffractometry is thus well suited.

2.5 Decoherence Theory

As introduced in Chapter 1, Decoherence Theory [9,10] sets out to describe and predict how classical behavior emerges out of quantum mechanics when a quantum object or system becomes unable to sufficiently isolate itself from interacting through entanglement with its external environment. Despite Decoherence Theory being an extension of the rules of ordinary quantum mechanics, this explanation is a break from the orthodoxy of the Copenhagen interpretation of quantum mechanics, where the observation of quantum effects of any object is only limited by a the devising of a suitable classical measuring device that can carry out the task with the suitable precision [11].

2.5.1 Zurek's Formalism

The process of decoherence according to Zurek can be summarized in the following way [12]: starting with an initial state $|\psi_S\rangle$ of the "system" (in this case the electron) which is in a superposition of states $|\sigma_i\rangle$ which can be written in the form $|\psi_S\rangle = \sum_i \alpha_i |\sigma_i\rangle$. Separate from this is an external environment in an initial state $|\varepsilon_0\rangle$.

The system then interacts with the environment via entanglement, resulting in a new entangled state $\left|\Psi_{SE}(t)\right\rangle$. This process can be illustrated by:

$$\left|\Psi_{SE}(0)\right\rangle = \left|\psi_S\right\rangle \otimes \left|\varepsilon_0\right\rangle = \left(\sum_i \alpha_i \left|\sigma_i\right\rangle\right) \otimes \left|\varepsilon_0\right\rangle \xrightarrow[\text{entanglement}]{\text{interaction}} \left|\Psi_{SE}(t)\right\rangle = \sum_i \alpha_i \left|\sigma_i\right\rangle \otimes \left|\varepsilon_i\right\rangle . \quad (2.38)$$

The Hamiltonian of this interaction, $H_{s\varepsilon}$, has the important feature that in relationship to the states of the system, the commutation relationship $\left[H_{s\varepsilon}, \left|\sigma_i\right\rangle\left\langle\sigma_i\right|\right] = 0$ is obeyed. What this means is that there exists a set of states $\left|\sigma_i\right\rangle$, known as pointer states, which remain unchanged under entanglement.

The combined state is itself a pure state. Because we are interested only in the system and not the combined state, a partial trace is performed over the environment, resulting in a diagonalized density matrix,

$$\rho_s(t) = Tr_E \left|\Psi_{SE}(t)\right\rangle\left\langle\Psi_{SE}(t)\right| = \sum_i \left|\alpha_i\right|^2 \left|\sigma_i\right\rangle\left\langle\sigma_i\right| . \quad (2.39)$$

which is a diagonalized matrix with no remaining coherences (the off-diagonal elements are zero). Importantly, the set of states that end up on the diagonal of the density matrix is always the same [12], (hence the pointer states being "preferred" states). The process in which these pointer states are selected by the environment is known as "environment-induced superselection" or "einselection" [9]. If the system was to start in one of these preferred states, the interaction of the environment would do nothing to it, and the system would remain unperturbed. These stable pointer states that appear on the diagonal of $\rho_s(t)$ after a decoherence time with probabilities $\left|\alpha_i\right|^2$ (see equation 2.14), are

effectively classical states, as the coherences between these $|\sigma_i\rangle$ states become insignificant (or in the extreme case completely uncorrelated) [12].

2.5.2 Density Matrix and Coherence

An important example of a decoherence process (applied to models such as the one described in section 2.6.1) is a particle interacting with an environment of harmonic oscillators [11,13], i.e. a quantum scalar field φ [14]. Excitations of this scalar field scatter off the particle, carrying with it information about the particle's position *x*. This therefore leads to the localization of the particle in position space [15].

The system-environment interaction takes the form via the Hamiltonian [13],

$$H_{s\varepsilon} = x\sum_i c_i q_i \,, \tag{2.40}$$

where q_i corresponds to the position of the i^{th} quantum oscillator with coupling strength c_i. After tracing over the environment this results in a new evolution equation of the particles density matrix (also known as a master equation) which has an exact solution [16]. In the high temperature limit of the harmonic bath (thermal fluctuations of the field rather than zero-point vacuum fluctuations), the master equation becomes [9]:

$$\dot{\rho}_s = -\frac{i}{\hbar}\left[H,\rho_s\right] - \gamma\left(x-x'\right)\left(\frac{\partial}{\partial x}-\frac{\partial}{\partial x'}\right)\rho_s - \frac{2m\gamma k_B T}{\hbar^2}\left(x-x'\right)^2\rho_s \,. \tag{2.41}$$

The first term alone (when the relaxation rate $\gamma = 0$) corresponds to the Von Neumann equation (equation 2.13). the second term is responsible for relaxation (or dampening) with its rate proportion to the "viscosity" of the particle in the harmonic bath . The third term is responsible for random fluctuations associated with quantum Brownian motion [9].

Zurek defines the diagonal as those elements which $x-x'=0$ and $(\Delta x)^2 \approx (x'-x)^2$ as elements the square of the separation between off-diagonal elements [11].

In the macroscopic/classical limit $\hbar$ is small compared to other combined terms with units of action (e.g. $\hbar << \sqrt{2m\gamma k_B T\left\langle (x-x')^2 \right\rangle}$) the high temperature particle evolution equation becomes dominated by the third term and the equation can be further approximated to [9],

$$\dot{\rho}_s = -\gamma \frac{(x-x')^2}{\lambda_T{}^2}\rho_s, \tag{2.42}$$

where the thermal De Broglie wavelength here is defined as $\lambda_T \equiv \hbar/\sqrt{2mk_bT}$. This linear differential equation can be solved as it is in the simple form $\frac{\partial y}{\partial t} = -cy$ and thus has a solution of

$$\rho_s(x,x',t) = \rho_s(x,x',0)e^{-\gamma\left(\frac{x-x'}{\lambda_T}\right)^2 t} = \rho_s^{initial} e^{-t/\tau_{dec}}. \tag{2.43}$$

From this general solution, we can conclude that the coherence terms in the density matrix decay exponentially with a decoherence timescale $\tau_{dec} = \gamma^{-1}(\lambda_T/\Delta x)^2$. After a sufficiently long time *t* compared to the decoherence time τ_{dec} , the density matrix will be approximately diagonalized and the particle behaves classically.

An alternative derivation by Breuer and Petruccione [15] utilized an underlying master equation based on a general Markov process,

$$\dot{\rho}_s(t) = -i\left[H_s, \rho_s(t)\right] - \Lambda\left[\vec{x}, \left[\vec{x}, \rho_S(t)\right]\right], \tag{2.44}$$

where the Hamiltonian of the particle is $H_s = \frac{\vec{p}^2}{2m}$ with a coordinate $\vec{x}$. Equation 2.44 also has a similar solution to equation 2.41 in the "recoilless limit" (where damping effects are neglected). This means that the decay of the off-diagonals coherences occurs on an time scale much shorter than the dampening of the diagonal elements. Additionally, it is assumed that the free evolution corresponding to the Hamiltonian (for example the broadening of the state due to beam divergence) [15]. This leads to the solution

$$\rho_s\left(\vec{x},\vec{x}',t\right) \approx \rho_s\left(\vec{x},\vec{x}',0\right)e^{-\Lambda\left(\vec{x}-\vec{x}'\right)^2 t}, \tag{2.45}$$

where Breuer and Petruccione calls Λ the "decoherence rate" [15]. Nevertheless, the similarity between this Markovian decoherence solution and the decay of the density matrix due to quantum Brownian motion in equation 2.43 when $\Lambda = \gamma/\lambda_T^2$.

Oftentimes when the off-diagonal terms decay exponentially (as in equation 2.43 and equation 2.45) the absolute value of the terms in the exponent are combined to form the decoherence factor Γ. For example, in the case of equation 2.43, the decoherence factor is $\Gamma = t/\tau_{dec}$.

2.6 Mechanisms of Decoherence Due to Conducting Surfaces

The following subsections provide examples of application of this decoherence program by inserting particular physical models where an electron may decohere due to its interaction with a conducting surface acting as an environment. Each physical model presumes its own decoherence timescales τ_{dec}. These physical models are tested experimentally as described in Chapter 4. In the experiment, the distance *y* between the electron and the surface is not constant in time as it propagates over the surface (because

of the image charge force). Therefore, because of this the strength of the interaction changes over its propagation (i.e. τ_{dec} is a function of y), the decoherence evolution of the density matrix from the beginning of the surface at time T_i to the end of the surface at time T_f is modified to take the form

$$\rho_s\left(x,x',T_f\right)=\rho_s\left(x,x',T_i\right)e^{-\Gamma}=\rho_s\left(x,x',T_i\right)e^{-\int_{T_i}^{T_f}\frac{t}{\tau_{dec}}dt}. \tag{2.46}$$

Additionally, the decoherence factor Γ of these decoherence processes do not universally take the form $e^{-\Gamma}=e^{\alpha(\Delta x)^2}$. To take the example of Scheel and Buhmann (section 2.6.2 [17]) the decoherence factor is modified due to the assumption of an image charge travelling under both paths in their two path setup. However, for small $\Delta x<<1$ the first order term in the expansion of these decoherence factors is consistently $\Gamma^{(1)}\approx a_1\left(\Delta x\right)^2$. This points to the importance of investigating decoherence effects by varying the parameter Δx at large values as an alternative to distinguishing between different effects. See Appendix C for a description of how this evolution of the density due to decoherence is implemented numerically as it relates to the decoherence experiment.

2.6.1 Free Electron-Surface Decoherence Models

The original decoherence model that focused on electron-surface decoherence was conceived by Anglin and Zurek [18,19]. The physical system is a classical image charge on the surface of the conductor that follows the free electron as it travels parallel to the surface (see figure 6). Joule heating, which the image charge experiences while

traversing the surface, causes dissipation with a relaxation time τ_{relax}. Back-action on the free electron leads to decoherence with a corresponding time τ_{dec} (called decorrelation time in [13]). The decoherence time is taken to be proportional to the relaxation time according to [13],

$$\tau_{dec} = \left(\frac{\lambda_{th}}{\Delta x}\right)^2 \tau_{relax}, \tag{2.47}$$

with the thermal de Broglie wavelength of $\lambda_{th} = \hbar/\sqrt{mk_bT}$ in accordance to Sonnentag [20]. Anglin and Zurek proposes that power is dissipated due to the Ohmic resistance that the image charge experiences while it travels over the surface. For an image charge with velocity v (which is approximated to be equal to the velocity of the free electron that is at a distance *y* away from the interface of the surface), and a surface resistivity of ρ. According to Boyer and Chapman et al. this dissipated power is found to be [21,22]

$$P_{Joule} = \frac{e^2 \rho v^2}{16\pi y^3}. \tag{2.48}$$

This power loss is responsible for the relaxation (or dampening) time. Equating this power dissipation to the power associated with the change in kinetic energy of the image charge:

$$P = \frac{d}{dt}\left(\frac{1}{2}mv^2\right) = mv\frac{dv}{dt}. \tag{2.49}$$

Taking the definition of relaxation time according to Zeh et. al. [10]:

$$\tau_{relax} \equiv v\Big/\left|\frac{dv}{dt}\right|. \tag{2.50}$$

This can now be substituted into Eq. 2.48 and then be equated to loss of power due to image charge to arrive at the relaxation time due to Ohmic dissipation:

$$P = mv\dot{v} = \frac{mv^2}{\tau_{relax}} = \frac{e^2 \rho v^2}{16\pi y^3} \rightarrow \tau_{relax} = \frac{16\pi m y^3}{e^2 \rho}. \tag{2.51}$$

Thus, substituting this into equation 2.47, the resulting decoherence time scale is

$$\tau_{dec}^{Zurek} = \frac{4h^2}{\pi e^2 k_B T \rho} \frac{y^3}{(\Delta x)^2}. \tag{2.52}$$

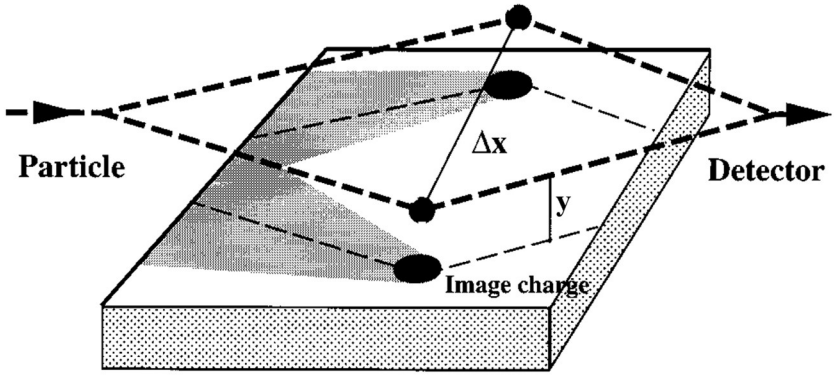

Figure 2.6. Original Electron-Surface Decoherence Model from Anglin and Zurek (image from [18]). Two electron paths that are separated by a distance Δx pass over a surface at height y and subsequently recombine at the detector. The resulting loss of contrast in this model is attributed to the Joule heating that the classical image charge experiences as transverses the surface.

The decoherence model by Scheel and Buhmann [17] is also based on the electron's interaction with its image charge, but it considers a full macroscopic quantum electrodynamic treatment. This considers the surface's linear dielectric response. Taking the low frequency limit where the Drude approximation $\varepsilon(\omega) \approx 1 + i/(\varepsilon_0 \rho \omega)$ holds for both gold [17] and doped silicon [23,24], the decoherence time scale is

$$\tau_{dec}^{Buhmann} = \frac{\pi\varepsilon_0\hbar^2}{e^2 k_B T \rho}\left[\frac{1}{2y} - \frac{1}{\sqrt{(2y)^2 + (\Delta x)^2}}\right]^{-1}. \tag{2.53}$$

In the limit $\Delta x << y$ this is equivalent to Equation 2.52.

Machnikowski's fully quantum many-body electron gas model implies that the primary decoherence mechanism is due to the dissipative effects of image charge formation rather than Ohmic resistivity effects [7]. It is notably dependent on the Fermi wave-vector for metals (k_{Fermi}). This decoherence time scale is

$$\tau_{dec}^{Machnikowski} = \frac{32\varepsilon_0 h^2 k_{Fermi}}{\pi e^2 m k_B T}\left(\frac{y}{\Delta x}\right)^2. \tag{2.54}$$

Howie's model [24] is based on event probability e^{-P} rather than energy dissipation, where such events correspond to aloof scattering with long wavelength plasmons and "similar excitations" up to a cutoff frequency 0.6x10^{12} Hz. The stated expression for this probability is

$$P^{Howie} = \left[\frac{e^2 L\omega_m^2}{4\pi^2\hbar\sigma v^2}\right]\int_{\frac{y}{4\Delta x}}^{\infty}\frac{\exp(-s)}{s}ds. \tag{2.55}$$

The exponential integral is approximated by [25],

$$-Ei(-\eta) = \int_{\eta}^{\infty}\frac{\exp(-s)}{s}ds \cong \left(A^{-7.7} + B\right)^{-0.13}, \tag{2.56}$$

where $\eta = y/4\Delta x$, $A = \log\left[(0.56146/\eta + 0.65)(1+\eta)\right]$, and $B = \eta^4 e^{7.7\eta}(2+\eta)^{3.7}$. Note that in the original article [24], η is mistakenly written as a factor of 16 different than the determined value (A. Howie, private communication). It should be mentioned that this theory has been further elaborated to be explained in terms of the electron's loss of

energy by emission of photons associated with the material's optical excitations (in a similar way the material may also undergo photon emission) [26,27].

There has been much interest in the potential to measure the effects of decoherence due to vacuum field fluctuations in such an experiment as a biprism interferometer [28–33]. It has been shown that, absent the surface, the decoherence factor Γ scales with $\sim(\Delta x)^2/(T^2c^2)$ where T is the total time of flight of the electron [30,32] and *c* is the speed of light. This decoherence effect is said to be intrinsic to the electron propogating in free space; and if a surface is brought near the interferometer, decoherence due to vacuum field fluctuations may be either enhanced by up to a factor of 2 or suppressed (called "recoherence") depending on if the plane of the separated paths is perpendicular or parallel to the surface respectively (see Figure 2.7) [32,33].

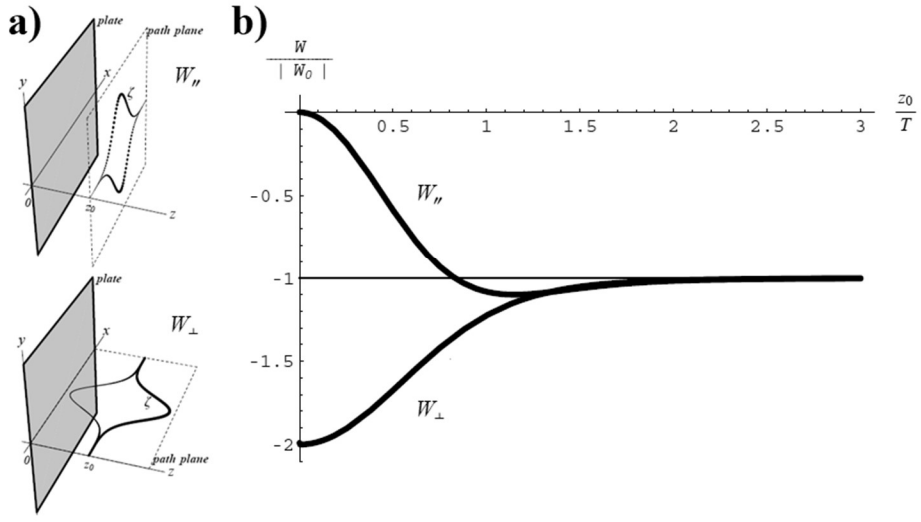

Figure 2.7: Surface Dependence on Decoherence due to Vacuum Field Fluctuations. a) Electron interferomer/separated path configuration either parallel (top) or perpendicular (bottom) to the surface. Here z_0 corresponds to the height of the initial state with respect to to the surface (elsewhere defined as y as in figure 6 et al), and T is the total time of flight over the surface between separation and recombination of states. W corresponds to the negative of the decoherence factor $-\Gamma$. If no surface is present, $W = W_0 \approx -(\Delta x)^2/(T^2c^2)$. b) The height

dependence of the decoherence factor with respect to the surface. As the interferometer is brought near the surface, the decoherence factor decreases toward zero in the parallel case. In in the perpendicular case, the decoherence factor tends toward twice the vacuum field decoherence when no surface is present. Image taken from Hsiang and Lee [32].

2.7 Decoherence vs Dephasing

While the two terms are often used interchangeably, decoherence and dephasing are two distinct phenomena that result in the loss of observed interference (a loss of contrast). However, they are physically quite different, decoherence corresponds to a irreversible process while dephasing corresponds to a reversible process. Zurek differentiates dephasing and decoherence in the following way [9]: There are processes which lead to measures which lead to "ignorance" of phase information, but these processes may not lead to an imprint (i.e. a transfer of information) of the state of the system on the environment (this is what we call dephasing). Dephasing due to "Classical noise" is when a "classical perturbation" (for example random phase noise in the potential) leads to unitary (reversible) evolution, but the evolution remains "unknown". This leads to the modification of the system/apparatus combined wave function such that [9],

$$\left|\Psi_{SA}\right\rangle = \left(\sum_j \alpha_j \left|A_j\right\rangle\right) \rightarrow \sum_j \alpha_j \exp\left(i\phi_j^{(n)}\right)\left|s_j\right\rangle\left|A_j\right\rangle = \left|\Psi_{SA}^{(n)}\right\rangle. \tag{2.57}$$

Here we use $\left|\Psi_{SA}^{(n)}\right\rangle$ to corresponds to different resulting realizations of random phase noise, a member of an ensemble of *n* final states. If the process is acting on the system (or the particle), the dephasing Hamiltonian takes the form

$$H_d^{(n)} = \sum_j \dot{\phi}_j^{(n)}(t)\left|s_j\right\rangle\left\langle s_j\right|. \tag{2.58}$$

Note that a similar Hamiltonian exists if dephasing occurs on the state of the apparatus. An important characteristic of $H_d^{(n)}$ is that it does not change the "nature" or the "degree of the system/apparatus correlations" [9]. It does not transfer information about the state of either the system of the apparatus onto the environment.

Note that each final state $\left|\Psi_{SA}^{(n)}\right\rangle$ is a pure state. Additionally, if the noise information $\phi_j^{(n)}$ is known or is acquirable, these pre-dephased state can be reconstructed. But when this is not the case (e.g., if different *n* corresponds to time-evolving iterations too fast to obtain the evolution of the phase noise), then the final system/apparatus state is written by the density matrix averaged over the ensemble of noise realizations:

$$\begin{aligned}\bar{\rho}_{SA} = \left\langle \left|\Psi_{SA}\right| \right\rangle \left\langle \left|\Psi_{SA}\right| \right\rangle = \sum_j \left|\alpha_j\right|^2 \left|s_j\right\rangle \left\langle s_j\right| \left|A_j\right\rangle \left\langle A_j\right| + \\ + \sum_{j,k} \sum_n e^{i\left[\phi_j^{(n)} - \phi_k^{(n)}\right]} \alpha_j \alpha_k \left|s_j\right\rangle \left\langle s_k\right| \left|A_j\right\rangle \left\langle A_k\right|,\end{aligned} \tag{2.59}$$

where here $\langle\ \rangle$ denotes an ensemble average. This ensemble averaging does result in the reduction of the off-diagonal terms of the density matrix and results in a loss of contrast, hence the similar observed result when decoherence occurs. An important conclusion here is that dephasing is a loss of phase coherence *between members of the ensemble* rather than between pointer states in decoherence, and this loss of coherence is due to differences in the noise in phases each member experiences.

2.7.1 Correlation Function and Entropy

As outlined in Chapter 5; the questions which we arise at are what other physical characteristics differentiate decoherence vs dephasing. More specifically is it possible to

distinguish between the two processes with limited information given only an observed interference intensity pattern (without reverting to directly obtaining the wave's phase information of the individual members of the ensemble as in tomography and entropy measurements). The tool utilized here is the second order correlation function (also known as the degree of second order coherence) for a single source is written as [34]:

$$g^{(2)}\left(x_1,t_1;x_2,t_2\right)=\frac{\left\langle \psi^*\left(x_1,t_1\right)\psi\left(x_1,t_1\right)\psi^*\left(x_2,t_2\right)\psi\left(x_2,t_2\right)\right\rangle}{\left\langle \left|\psi\left(x_1,t_1\right)\right|^2\right\rangle\left\langle \left|\psi\left(x_2,t_2\right)\right|^2\right\rangle}. \tag{2.60}$$

Taking the case of the wave function at time *t* at the detecting plane, considering symmetric points about the origin ($x_1=x$ and $x_2=-x$), and interpreting the ensemble average to be a either a time average or an average over an ensemble of different phase patterns, this becomes:

$$g^{(2)}\left(x;-x\right)=\frac{\left\langle \psi^*\left(x\right)\psi\left(x\right)\psi^*\left(-x\right)\psi\left(-x\right)\right\rangle}{\left\langle \left|\psi\left(x\right)\right|^2\right\rangle\left\langle \left|\psi\left(-x\right)\right|^2\right\rangle}=\frac{\left\langle I\left(x\right)I\left(-x\right)\right\rangle}{\left\langle I\left(x\right)\right\rangle\left\langle I\left(-x\right)\right\rangle}. \tag{2.61}$$

It is demonstrated from this final form of the second order correlation function that this measure, since it is in terms of the intensity distribution (after dephasing or decoherence) of each member final state, without any phase information of the wave functions.

Aside from the second order correlation function, entropy can be used to differentiate between dephasing and decoherence (although, in contrast to the former, the phase information or coherence terms of the density matrix does have to be known). In terms of a density matrix ρ (either pure or mixed) the entropy (particularly Von Neumann entropy) at a given time can be written as

$$S=-Tr\left(\rho\ln\left(\rho\right)\right), \tag{2.62}$$

where *Tr* (…) denotes the full trace (sum of diagonal elements). This is typically more easily computed by rearranging this in terms of the density matrix's spectral decomposition (in terms of the eigenvalues λ_i of ρ):

$$S = -\sum_i \lambda_i \ln(\lambda_i). \tag{2.63}$$

It will be demonstrated in Chapter 5 how an increase in entropy, $\Delta S > 0$, of a quantum system after undergoing a process indicates that it is a decohering process (as it is related to nonunitary, irreversible evolution), and a process with unchanging entropy, $\Delta S = 0$, may be associated with dephasing (as dephasing is still unitary and thus reversible).

2.8 Chapter 2 Bibliography

[1] D. J. Griffiths, *Introduction to Electrodynamics, 3rd Edition* (Pearson, 1999).
[2] J. D. Jackson, *Classical Electrodynamics*, 2nd ed. (Wiley, 1975).
[3] W. C.-W. Huang, R. Bach, P. Beierle, and H. Batelaan, J. Phys. Appl. Phys. **47**, 085102 (2014).
[4] C. Cohen-Tannoudji, B. Diu, and F. Laloe, *Quantum Mechanics* (Wiley, Singapore, 2005).
[5] C. Cohen-Tannoudji, B. Diu, and F. Laloe, in *Quantum Mech.* (Wiley-VCH, Singapore, 2005), pp. 211–266.
[6] C. Cohen-Tannoudji, B. Diu, and F. Laloe, in *Quantum Mech.* (Wiley, 1991).
[7] P. Machnikowski, Phys. Rev. B **73**, 155109 (2006).
[8] A. D. Cronin, J. Schmiedmayer, and D. E. Pritchard, Rev. Mod. Phys. **81**, 1051 (2009).
[9] W. H. Zurek, Rev. Mod. Phys. **75**, 715 (2003).
[10] D. Giulini, E. Joos, C. Kiefer, J. Kupsch, I.-O. Stamatescu, and H. D. Zeh, *Decoherence and the Appearance of a Classical World in Quantum Theory* (Springer-Verlag, Berlin Heidelberg, 1996).
[11] W. H. Zurek, Phys. Today (2008).
[12] W. H. Zurek, (2017).
[13] W. H. Zurek, in *Front. Nonequilibrium Stat. Phys.*, edited by G. T. Moore and M. O. Scully (Springer US, 1986), pp. 145–149.
[14] W. G. Unruh and W. H. Zurek, Phys. Rev. D **40**, 1071 (1989).

[15] H.-P. Breuer and F. Petruccione, in *Theory Open Quantum Syst.* (Oxford University Press, 2002), pp. 232–242.
[16] B. L. Hu, J. P. Paz, and Y. Zhang, Phys. Rev. D **45**, 2843 (1992).
[17] S. Scheel and S. Y. Buhmann, Phys. Rev. A **85**, 030101 (2012).
[18] J. R. Anglin, J. P. Paz, and W. H. Zurek, Phys. Rev. A **55**, 4041 (1997).
[19] J. R. Anglin and W. H. Zurek, in *Dark Matter Cosmol. Quantum Meas. Exp. Gravit.* (Editions Fronti'eres, Gif-sur-Yvette, France, Les Arcs, Savoie, France, 1996), pp. 263–270.
[20] Peter Sonnentag, Ein Experiment zur kontrollierten Dekohärenz in einem Elektronen-Biprisma-Interferometer, Ph.D. thesis, University of Tübingen, 2006.
[21] T. H. Boyer, Phys. Rev. A **9**, 68 (1974).
[22] M. S. Chapman, T. D. Hammond, A. Lenef, J. Schmiedmayer, R. A. Rubenstein, E. Smith, and D. E. Pritchard, Phys. Rev. Lett. **75**, 3783 (1995).
[23] M. van Exter and D. Grischkowsky, Appl. Phys. Lett. **56**, 1694 (1990).
[24] A. Howie, Ultramicroscopy **111**, 761 (2011).
[25] P. H. Giao, Ground Water **41**, 387 (2003).
[26] F. J. García de Abajo, Phys. Rev. Lett. **102**, 237401 (2009).
[27] F. J. García de Abajo, Rev. Mod. Phys. **82**, 209 (2010).
[28] J. Schwinger, Phys. Rev. **152**, 1219 (1966).
[29] L. H. Ford, Phys. Rev. D **47**, 5571 (1993).
[30] H.-P. Breuer and F. Petruccione, Phys. Rev. A **63**, 032102 (2001).
[31] F. D. Mazzitelli, J. P. Paz, and A. Villanueva, Phys. Rev. A **68**, 062106 (2003).
[32] J.-T. Hsiang and D.-S. Lee, Phys. Rev. D **73**, 065022 (2006).
[33] J.-T. Hsiang and L. H. Ford, Int. J. Mod. Phys. A **24**, 1705 (2009).
[34] K. K. Sharma, *Optics, Principles and Applications* (Elsevier Inc., Oxford, UK, 2006).

CHAPTER 3

OPTICAL ELECTRON SWITCH

3.1 Introduction

This Chapter details a free electron beam switch which was developed involving interaction with a laser-induced surface charge [1]. The original motivation of this experiment was the creation of an ultra-fast electron switch (within the femtosecond regime) [2]. Such a fast switch has important uses for the fast detection of electrons in experiments such as an electron Stern-Gerlach experiment [3], freefall experiments for electrons [4], plasmonic physics [5], the detection of ultra-fast physics [6,7], and the development of an electron dispersion compensator [8].

The concept behind this fast electron switch makes use of the short travel time-scales of the electron over a nanostructure. Specifically, the idea was to design a rectification switch using a nanofabricated diffraction grating combined with a laser (see Figure 3.1). The electric field of the optical laser induces electric dipoles on the top surface/edge of the 100 *nm* grating, and these dipoles would then in turn oscillate in phase with the field of the laser. Then, as the electron passes over these dipoles, it would feel a Coulomb force toward (or away, depending on the initial timing of arrival) from the surface as it passes over the beginning of the grating bar (a single solid portion of the grating). Matching the velocity and thus time of flight over the grating bar with the period of the oscillating laser field/dipole field over half the field's period ensures that the electron experiences a force in the same direction for the duration of its travel (thus the force becomes accumulative). Furthermore, the speed of this switching mechanism then simply depends on the speed of the electron and the width of the grating bars. As an

example, for a 3.98 *keV* electron and a 100 *nm* grating bar, the deflection switching mechanism can be as fast as $t_{switch} = a_{grating} / v_{flight} = (100\ nm) / \left(3.74 \times 10^7\ {}^{m}/_{s}\right) = 2.67\ fs$.

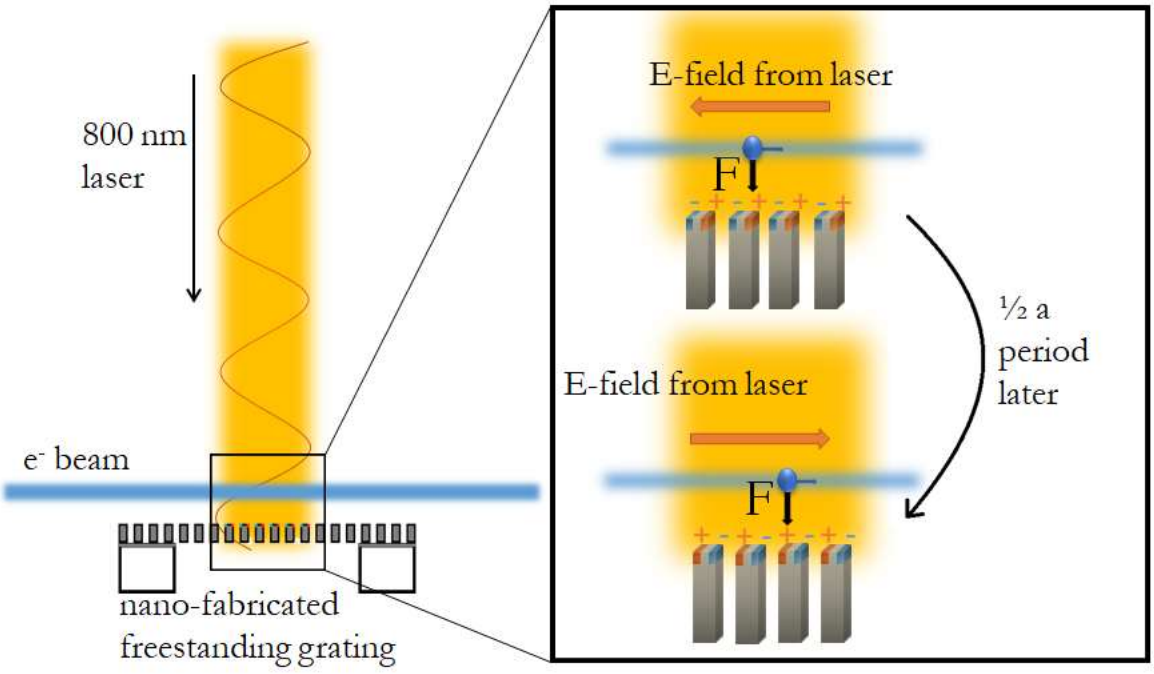

Figure 3.1: Fast Electron Rectification Switch Concept. If we can match the laser frequency with a 3.98 keV electron's travel time over a 100 *nm* periodic grating bar we can access a steering process that will be as fast as 2.67 *fs* [9,10].

However, upon attempting to observe deflections of the electron beam due to such a rectification effect, a much larger deflection angle had occurred [1]. It is this effect (which cannot be explained by the above rectification mechanism) that was measured and characterized.

3.2 Setup and Procedure

A 3.98 keV electron beam is collimated by two slits that have widths of 5 μm and 2 μm in the y-direction and are separated by 24 cm in the z-direction. This produces a y-direction beam divergence of $\approx (5\ \mu m + 2\ \mu m) / 24\ cm = 29\ \mu radians$ (see figure 3.2). 6

cm after the second slit, the electron beam passes over a surface or grating in the *x*-*z* plane (primarily SiN, Au coated) with a beam width of 10 μm. This is while a laser (658-800 nm) is focused with a cylindrical lens on the wall near the e-beam's path. This cylindrical focusing is utilized to maximize the intensity of the laser along the electron's path over the surface. By turning the laser on and off via a mechanical chopper or an acousto-optic modulator (AOM), the resulting angle in which the electron propagates can be changed (from a straight-travelling beam to a deflected beam).

To measure the angle in which the beam is deflected, a 5 μm detection slit 24 cm after the surface samples a portion of the electron beam's profile in the *y*-direction. By scanning the beam in the *y*-direction with deflection plates (10 cm before the detection slit), an electron distribution is acquired using a multichannel scaler (MCS) software and counting electronics connected to a multichannel plate (MCP). By moving the detection slit by a known distance in the *x*-direction (as measured by the linear feedthrough's micrometer) and comparing how far in time the center of the histogram travelled, the width and position of the electron beam at the detection screen can be calibrated.

The lasers used were continuous-wave diode lasers with powers of 1 mW, 10 mW and 5 mW with respective wavelengths of 532 nm, 685 nm and 800 nm. The cylindrical lens produced a laser focus FWHM of 280 μm in the y-direction and retained the spot's length of 1 mm in the z-direction along the beam path. The idea behind not focusing the beam in the z direction is to maximize the time in which the beam experiences the charge distribution to maximize the force. With these wavelengths, deflections were observed (though with different magnitudes) indicating that the observation is robust in its generality.

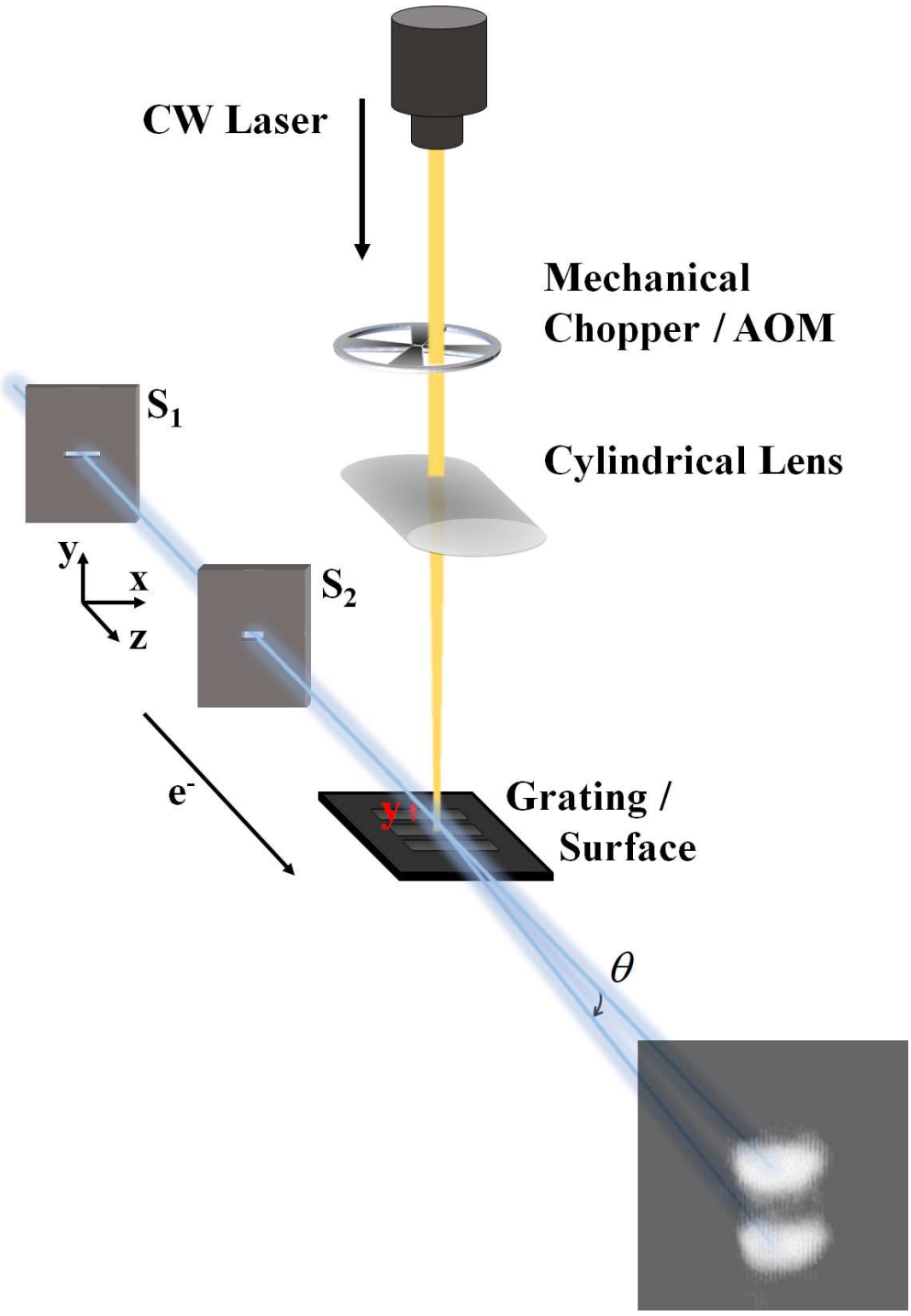

Figure 3.2: Experimental Setup for Surface Optical-Electron Switch. An electron beam is collimated by slits S_1 and S_2 and passes over a Grating/Surface. As the surface is illuminated by a laser beam, it is deflected by an angle θ when the surface is illuminated by a laser beam. For the electrons to switch from one angle/spatial position to another, a mechanical chopper (for low

frequency) or acousto-optic modulator (AOM, for high frequency) turned on and off the laser beam.

3.3 Measurements and Results

To observe the switching speed, the edge of the mount holding the detection slit was used to block the area where the deflected beam would hit the MCP detector (see Figure 3.3). Using the MCS it was found that the switching speed (from on to off) was approximately 6 μs.

Using the MCS spatial calibration as a guide, the deflection angle was also measured as a function of chopping frequency. For low frequencies, this was achieved with a mechanical chopping wheel. Because this chopping frequency was limited to 2000 Hz and a considerable chopping angle was still observed, the mechanical chopper was replaced with the acousto-optic modulator as the switching mechanism for the laser. Angular deflection was observed up to a frequency of 2 MHz.

Interestingly, the deflection angle dropped roughly linearly with chopping frequency but becomes constant at approximately 100 Hz; then it drops linearly once more starting at approximately 300 kHz. Also, note that the deflection angle between using the mechanical chopper and the AOM during constant frequency region (a difference of a factor of ≈ 2) is due to the loss of intensity of the laser beam as it passed through the AOM.

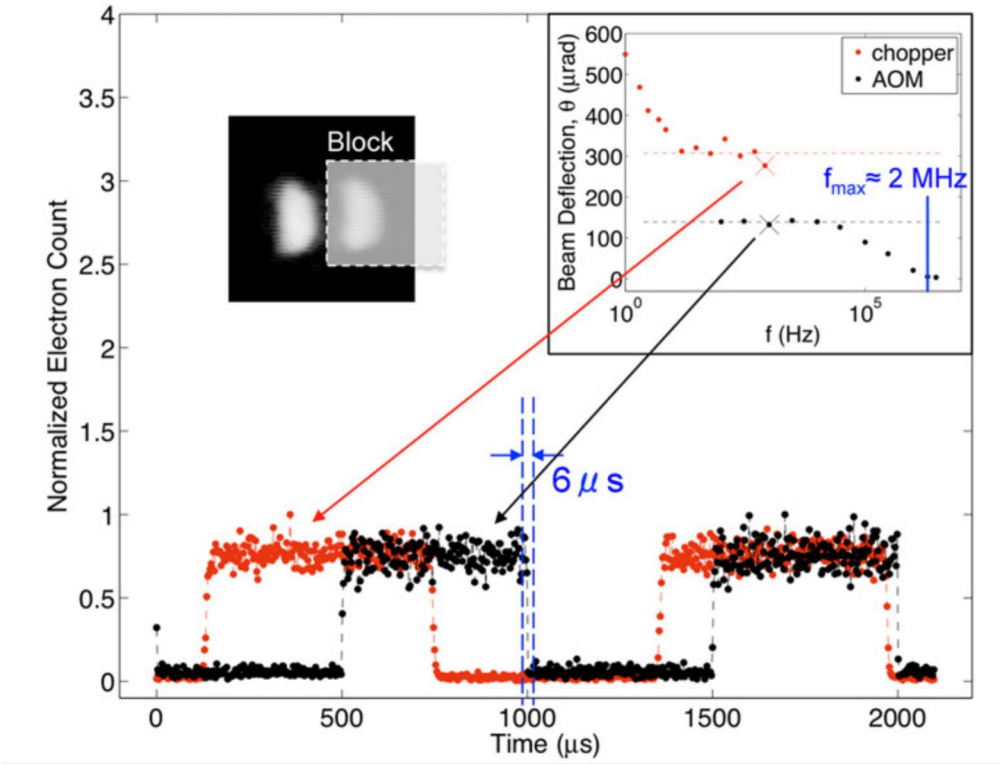

Figure 3.3 Electron-Optical Switch Speed. Plotted is the electron counts as a function of time as the laser is switched on and off. The chopping data for the mechanical chopper (red) at 818 Hz and the AOM (black) at 1000 Hz are shown. Top-left: A time accumulated electron image from the CCD camera shows the initial (left) and deflected electron beam (right). A semi-transparent rectangle "Block" represents the edge of the detection slit mount used to switch the electron beam "on" and "off". Top-right: Plotted is the electron beam deflection as a function of the chopping frequency (using the AOM or mechanical chopper). The maximum chopping frequency is approximately 2 MHz. The red data points represent the data collected with the chopper and the black data points with the AOM.

A distance dependence measurement of this effect was also taken. Illustrative of how strong this effect is with such low power light, the electron beam can be steered or deflected when it is as far as 200 μm away from the surface (farther than which the deflection amplitude is no longer larger than the divergence angle, thus the deflected beam becomes indiscernible from the original beam), see Figure 3.4. Granted, the power of these lasers was not necessarily maximized, so the beam may potentially be deflected even farther from these surfaces.

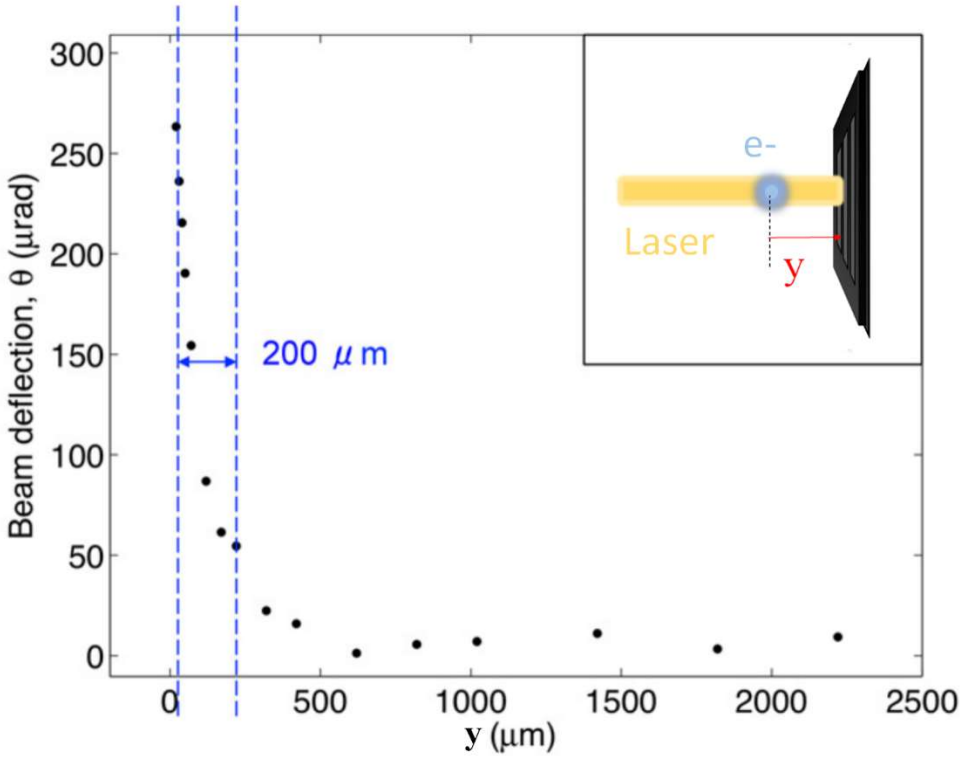

Figure 3.4 Beam Deflection as a Function of Distance Between the Electron Beam and the Surface. the electron beam can be deflected when it is as far as 200 μm away from the surface

One of the most remarkable aspects of this observed deflection mechanism is how the deflection angle changes as the laser beam laterally sweeps perpendicular in the x-direction to the beam's path (see Figure 3.5). While the electron deflects towards the surface as the center of the laser beam is directly over the surface, as the laser beam is moved either to either side of the electron's path the deflection angle amplitude decreases to zero and then further *inverts* in the opposite direction such that the beam deflects away from the surface, and of course further movement of the laser from the beam's path leads to a settling out of no deflection. From this deflection distribution as a function of laser position in the x-direction and the relative width of the laser width of 280 μm (which is comparable to the distance between the crossovers from attractive deflection to repulsive

deflection) is indicative of a formation of the charge distribution being dependent on the gradient of the intensity of the laser (which is what inspired the ponderomotive model for explaining this laser induced deflection phenomenon).

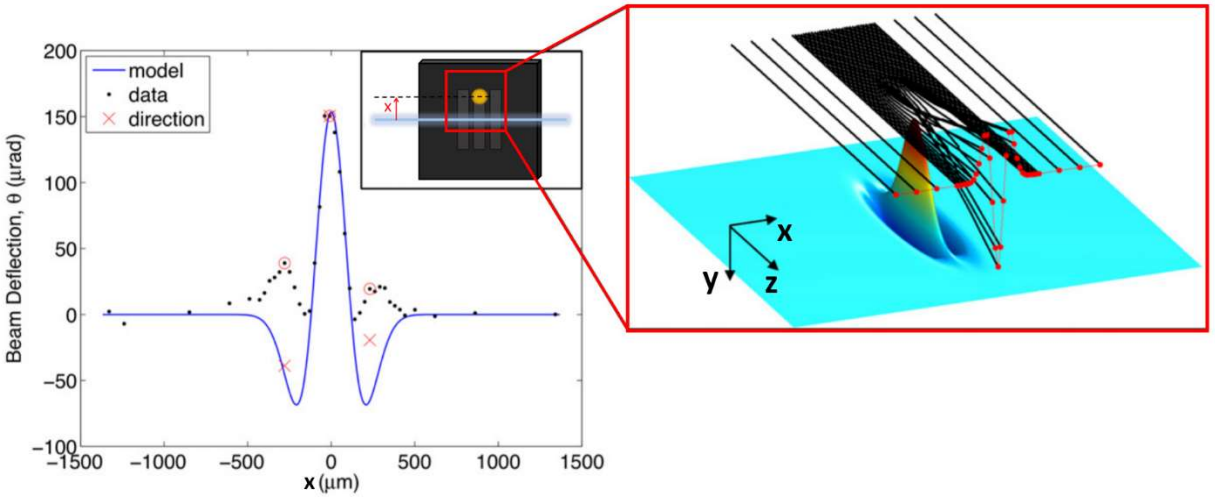

Figure 3.5: Change in Beam Deflection Angle as a Function of the Lateral Position (x) of the Laser Focus. Left: The measured deflection magnitude is measured as a function of the distance between the electron's path and the center of the laser spot. The direction of the deflection is measured at three positions (red circles). The value of these angles including the sign of the direction are indicated (red crosses). Reversals of deflection direction is attempted to be explained by a heuristic model (blue line) of a light-induced surface-charge redistribution based on a ponderomotive force. Right: Diagram of the electron trajectories (black lines) and surface-charge density on the surface (blue to red surface plot) is shown. The interaction between the electron beam and the surface charge distribution is such that it is attractive in the middle of the laser spot and repulsive at the sides of the laser spot (roughly beyond the width of the laser spot, as the FWHM of the laser is focused on the surface at 280 μm).

In addition to a range of wavelengths, a range of available surface types were also investigated, including gold-palladium coated silicon nitride grating, bare silicon nitride grating, solid aluminum, and solid gallium arsenide (the last of which is unpublished). The general trend seen is that the surface-charge redistribution is driven by the intensity gradient of the laser due to a ponderomotive potential (as seen in Figure 3.5) and this what was found with metal coated SiN observation (Figure 3.5), but this is not the case for uncoated SiN (Figure 3.6) or aluminum. Combining this with issues with the

amplitude of the deflection being limited to only a fit factor casts considerable doubt on the ponderomotive force model proposed to explain this deflection model outlined in [1].

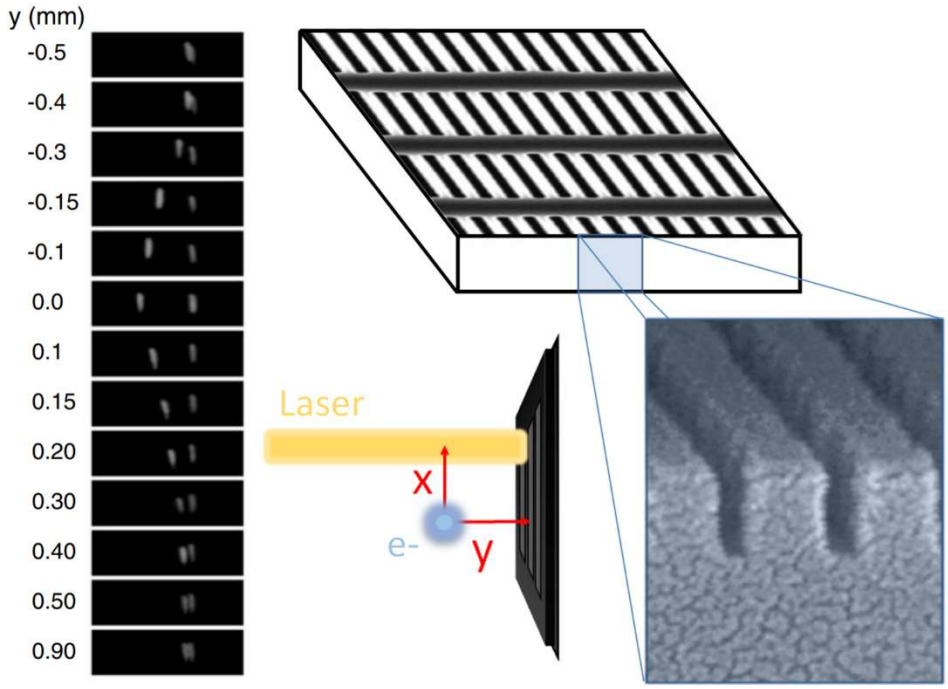

Figure 3.6: "Uncoated" Deflection Measurement. Electron beam deflection is measured as a function of the laser beam position x. This measurement is similar to that shown in figure 3.5, except the deflection is measured at a location on the surface where the coating was not visible, which we call: 'uncoated' or bare Silicon Nitride Grating. The deflection images are shown (left column). For all images the laser beam was chopped on and off, while the electron image was recorded continuously. Note that only deflection in one direction was observed in contrast to that reported in figure 3.5 for the coated SiN surface (for a more detailed description see text). An electron microscope image of the SiN surface is shown (top right). A higher magnification image of the edge view of a similar grating (bottom right) is reported earlier [11].

3.5 Conclusion and Outlook

Even without a complete understanding of the physical mechanism of the laser-induced surface charge redistribution which causes the deflection of the free electron, the setup still shows utility in applications such as electron beam lithography and microscopy where external electronics need to be kept isolated [12]. Additionally, further studies in understanding the mechanism of this laser-induced potential will be needed if one were to

investigate coherence effects after travelling over such an environment. Potential pathways of such an investigation include starting with more geometrically simpler surface (for example, a flat surface instead of a nanofabricated grating), using a range of materials which have been well characterized, and using a more comprehensive laser intensity dependence and wavelength dependence study.

3.6 Chapter 3 Bibliography

[1] W. C.-W. Huang, R. Bach, P. Beierle, and H. Batelaan, J. Phys. Appl. Phys. **47**, 085102 (2014).
[2] P. Beierle, W. Huang, R. Bach, M. Becker, D. Ruffner, and H. Batelaan, in (2014).
[3] H. Batelaan, T. J. Gay, and J. J. Schwendiman, Phys. Rev. Lett. **79**, 4517 (1997).
[4] M. Becker, A. Caprez, and H. Batelaan, Atoms **3**, 320 (2015).
[5] M. Becker, W. C.-W. Huang, H. Batelaan, E. J. Smythe, and F. Capasso, Ann. Phys. **525**, L6 (2013).
[6] J. C. Williamson, J. Cao, H. Ihee, Ahmed H. Zewail, H. Frey, J. Cao, H. Ihee, H. Frey, and A. H. Zewail, Nature **386**, 159 (1997).
[7] S. A. Hilbert, A. Neukirch, C. J. G. J. Uiterwaal, and H. Batelaan, J. Phys. B At. Mol. Opt. Phys. **42**, 141001 (2009).
[8] P. Hansen, C. Baumgarten, H. Batelaan, and M. Centurion, Appl. Phys. Lett. **101**, 083501 (2012).
[9] T. Plettner, R. L. Byer, E. Colby, B. Cowan, C. M. S. Sears, J. E. Spencer, and R. H. Siemann, Phys. Rev. Spec. Top. - Accel. Beams **8**, 121301 (2005).
[10] J. Breuer and P. Hommelhoff, Phys. Rev. Lett. **111**, 134803 (2013).
[11] T. A. Savas, M. L. Schattenburg, J. M. Carter, and H. I. Smith, J. Vac. Sci. Technol. B Microelectron. Nanometer Struct. Process. Meas. Phenom. **14**, 4167 (1996).
[12] MAPPER Lithography, www.mapperlithography.com/

CHAPTER 4

DECOHERENCE EXPERIMENT

4.1 Introduction

The continuous divide between quantum and classical physics can be described by decoherence theory. Decoherence is an irreversible process in which a quantum state entangles with an environment in such a way that it loses its wave interference properties [1,2]. For most experiments, maintaining a system's quantum coherence is desirable, and great efforts are made to isolate the system from its environment [3–6]. Additionally, it has been suggested that some sources of decoherence may be ubiquitous, such as those originating from vacuum field fluctuations or gravitation [7–12], and that decoherence in general is a critical element to resolving the quantum measurement problem [13]. Thus, experimentally sorting out various sources of decoherence and determining which dominate is desirable for both technical applications and fundamental studies, including the decoherence program [13].

There have been experiments in which the transition between the quantum and classical domain has been controlled through both the "distance" between states [14–16] and the strength of the interaction with the environment [16–20]. Most of these experiments involve various wave-matter interferometric techniques.

Here we will describe an decoherence setup that realizes Zurek's original thought experiment of diffracting charges through a grating and controlling the spatial quantum coherence with a conducting surface [21]. We have measured the effect of a gold and silicon surface and found upper bounds on the loss of contrast due to decoherence. These results refute current decoherence models premised on image charge [22–24]. We also

identify viable decoherence models based on dielectric excitation theory from effects including surface plasmons [25,26]. In addition, we propose a pathway to measure decoherence due to electromagnetic vacuum field fluctuations.

Sonnentag and Hasselbach previously used an electron biprism interferometer setup with separated arms passing over a semi-conducting surface before recombination [16]. In contrast, we used electron diffraction from a nano-grating and measured the effect of a conducting surface. Diffraction is well suited for measuring small losses in coherence, which is particularly useful in detecting weak decoherence channels. Sonnentag's and Hasselbach's measurements on doped n-type silicon reveals a decoherence strength that is a factor of $\approx 10^2$ too weak as compared to Zurek's image charge model. This is confirmed by our findings.

Their determination of the physical mechanism nevertheless supported image charge models [16,23] as the analysis ignored the strength of decoherence and was limited to a best fit of the functional form, as was done in a similar experiment by Röder and Lubk [27]. The implicit assumption is that a metallic surface (as used in the theory) behaves similarly as a silicon surface. The image charge models were thus considered valid. Our measurements which now also include the conductor gold as well as silicon, refute this conclusion and identify Howie's model [25,26] as viable.

4.2 Experimental Setup

A 1.67 keV electron beam (Kimball EGG-3101 electron gun) is sent through two collimation slits separated by 25 cm with a geometrical beam divergence of 61 μrad in the *x*-direction and 120 μrad in the *y*-direction (see Figure 4.1). This collimation gives a transverse coherence length of the electron beam of approximately 600 nm as determined

by diffraction images. This makes it possible to diffract the electrons from a 100 nm periodic nanofabricated grating [28,29]. The diffracted electron distribution is magnified 24 cm downstream by an electrostatic quadrupole lens, detected by a multichannel plate detector, backed by a phosphorous screen (Beam Imaging Solutions BOS-18), and imaged by a CCD camera. A LabVIEW image acquisition program [30] accumulates a two-dimensional streaming image from the camera. The vacuum chamber in which this experiment takes place is held at a pressure of $\approx 4 \times 10^{-5}$ Pa and is protected from external magnetic fields by two layers of mu-metal magnetic shielding.

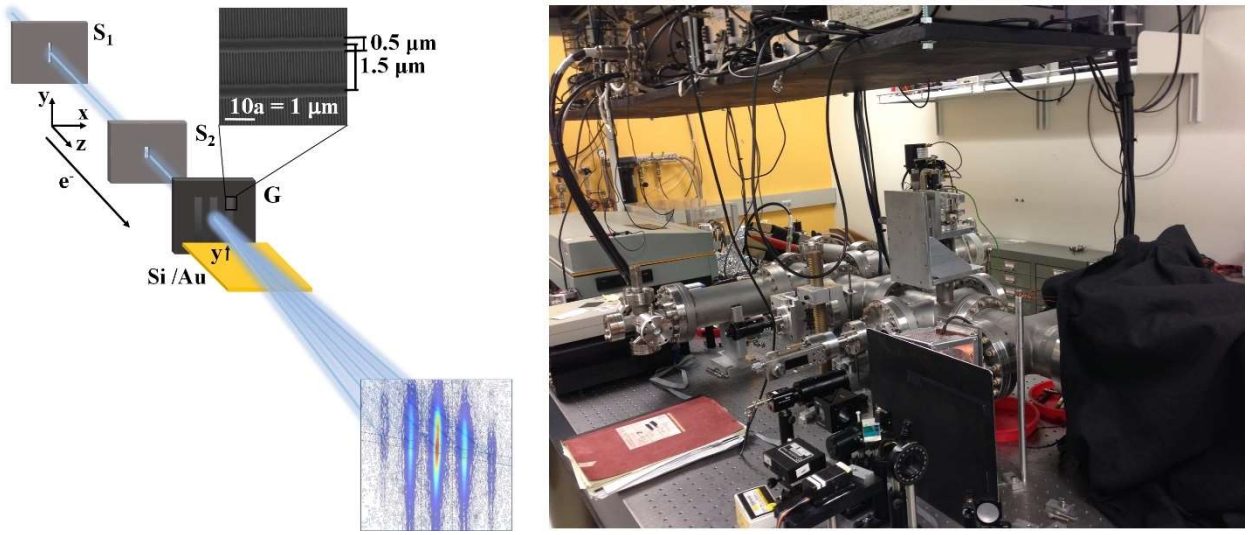

Figure 4.1: Experimental Setup. Left: Diagram of experiment. Electrons are prepared in a spatially coherent state by collimation with two slits (S_1 and S_2), then diffracted through a nanofabricated diffraction grating (G) before passing over either a doped silicon (Si) or gold (Au) surface, which acts as the decohering "environment". Left: Schematic of Experimental setup with contour plot of data. Right: Image of the "Linear Electron eXperiment" (LEX) utilized for this Decoherence Experiment (as well as the optical electron switch experiment in Chapter 3).

A 1 cm^2 surface is then brought in from below the diffracted beam 3 mm after the grating such that the surface in the *x-y* plane is perpendicular to the diffraction grating bars. The surface height is adjusted to cut into the beam so that 1/3 of the intensity of the

original beam reaches the detector. The surface is supported by a mechanical feedthrough whose angular pitch with respect to the beamline can be adjusted with a precision of approx. 0.2 mrad. This pitch of the surface is adjusted to maximize the electron beam's deflection due to image charge attraction. The Si surface was cleaned using a version of the industry-standard RCA cleaning method (without the oxide strip) [31], to remove dust or other contaminants (see Appendix D for more details).

4.3 Analysis

When an electron passes over a decohering surface, it interacts with the surface so that the interference pattern in the far field has lower visibility, and further decreases the closer the electron passes over the surface. Previously, the decoherence was measured in terms of the visibility of the interference pattern [14,16,18,27], i.e. $V_{is} = \left(I_{\max} - I_{\min}\right)/\left(I_{\max} + I_{\min}\right)$.

However, to measure smaller changes in contrast and reduce the uncertainty in measurement due to background counts, we measure coherence in terms of the transverse coherence length of the diffracted beam as observed at the detector [32]:

$$L_{coh} \approx \frac{\lambda_{dB}}{\theta_{coll}} \approx \frac{ad}{w_{FWHM}}. \tag{4.1}$$

Here a is the periodicity of the grating, d is the distance between neighboring diffraction peaks at the detection screen, and w_{FWHM} is the FWHM of a diffraction peak. Thus, here we associate a loss of coherence with an increase of the width of the diffraction peaks w_{FWHM} rather than a loss of visibility.

Two dimensional images of the electron interference pattern are recorded. Lineouts of the diffractograms are extracted so that each x-direction horizontal lineout corresponds to a 4.8 μm range in the y-direction on the detector where electron detection events occurred. The lineouts are taken at a slant with the x-direction to compensate for the image skew. These diffraction lineouts are fitted by the expression:

$$I(x)=\left(\frac{\sin(\alpha(x-x_0))}{\alpha(x-x_0)}\right)^2\times[G(x-x_1)+\sum_{n=1}G(x-x_1+nd)+G(x-x_1-nd)]+A_{bck}e^{-(x-x_2)^2/2c_3^2}, \tag{4.2}$$

where the first term corresponds to the single slit envelope and the diffraction peaks, and each individual peak,

$$G(x)=a_1e^{-x^2/2c_1^2}+(1-a_1)e^{-x^2/2c_2^2}, \tag{4.3}$$

is approximated in terms of two Gaussians with overlapping means, which also very well fits the shape of the beam without diffraction. The full width half max width w_{FWHM} of all diffraction peaks is constrained to be the same for a single diffraction pattern, as is the peak to peak distance d taken to be constant. More details on this fitting method can be found in Appendix A. From this fit w_{FWHM} and d are extracted to compute the coherence length L_{coh} for a given distribution according to equation 4.1 (Figure 4.2). For more details about coherence length in this context, see section 2.4.3 for more details.

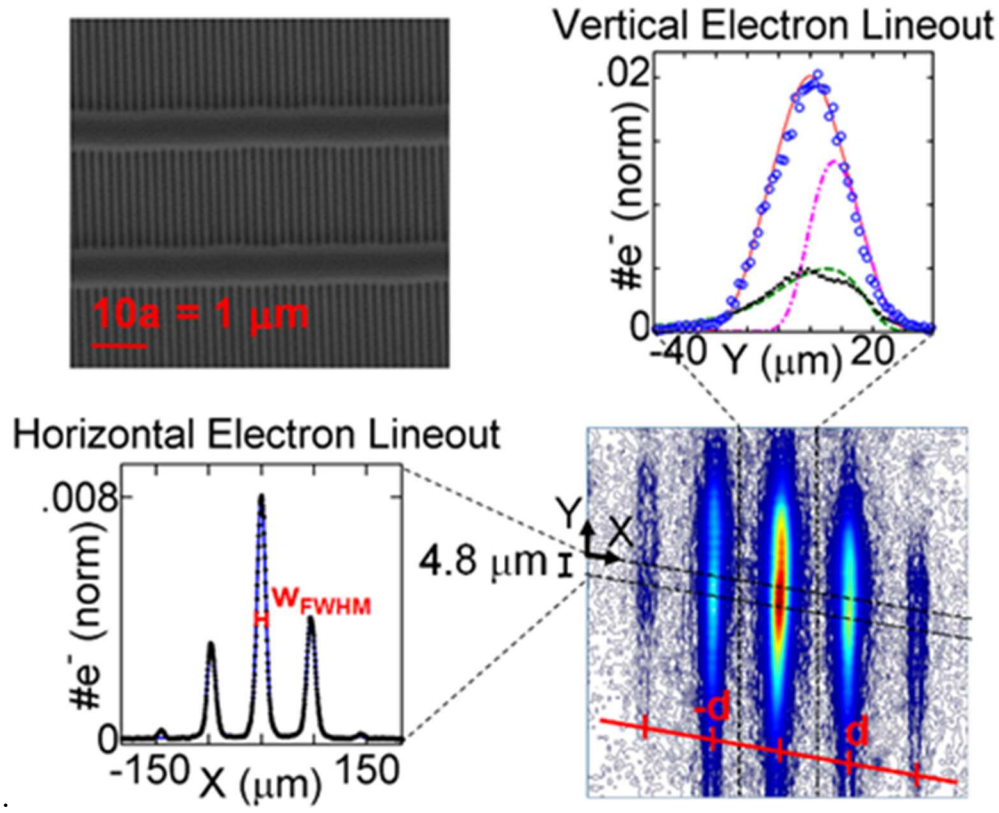

Figure 4.2: Analysis of Diffractogram. Bottom Right: Contour of accumulated CCD image of a diffractogram when the surface cuts the intensity by 1/3 of the original intensity. Top Right: vertical line-out (black solid dots) of zeroth order diffraction peak affected by image charge (see Appendix C for more details). Bottom Left: Horizontal Electron line-out of 4.8 μm of the Diffraction Pattern (Black solid dots). These lineouts are fitted by equation 4.2, and then assigned a coherence length according to $L_{coh} = ad/w_{FWHM}$. Top Left: Nanofabricated diffraction grating with periodicity a = 100 nm.

The advantage of using diffractometry over interferometry lies in their respective decoherence measures, L_{coh} and V_{is}. The background signal can be subtracted for diffraction without distorting the measured value of L_{coh} . This is not the case when measuring visibility in an interferometer. The visibility V_{is} drops off linearly due to a weak background signal, which can mask decoherence. For a weak decohering environment that scatters the incident beam and introduces background, diffractometry is thus well suited. For more details on this, see section 2.4.4.

4.4 Comparison between Experimental Results and Physical Models

As derived in section 2.5.2, decoherence over the surface modifies the density matrix of the electron according to [1,33]:

$$\rho_{final} = \rho_{initial} e^{-\int_{t_i}^{t_f} dt/\tau_{dec}} = \rho_{initial} e^{-\Gamma}, \tag{4.4}$$

where Γ is the decoherence factor. The decoherence time scale τ_{dec} is not only model-dependent, but also depends on the distance between paths Δx and the height above the surface y. For all electron-surface decoherence models described in section 2.6, the predicted final diffraction pattern is obtained by propagating the final density matrix ρ_{final} to the detection screen (see Appendix C for details). From this, the change in transverse coherence length is then obtained from the calculated far-field diffraction pattern using Equation 1.

Plotted in Figure 4.3 is a comparison of the coherence length as a function of vertical position Y at the detector for the case of two different n-type phosphorous doped silicon samples with resistivities 1-20 Ωcm and 1-10 Ωcm (data points). Our results agree with Hasselbach's experimental findings, who used a 1.5 Ωcm n-type doped silicon sample of 1 *cm* length using the same beam energy of 1.67 keV. The observed loss of contrast can be visualized in the diffractogram's diffraction peak broadening (Figure 4.3 top right), based on the histogram data collected from the CCD camera (see Appendix D for more details). The surface acting as a lens (e.g. a charge distribution on the surface on the lens) is ruled out as an explanation of the change in coherence length. This is because a simple lens model cannot simultaneously explain both the widths of the diffraction peaks and the peak periodicity values observed (see Appendix D for more details).

The experimental data is also compared to Zurek's model of classical image charge/Ohmic dissipation, Scheel and Buhmann's macroscopic quantum electrodynamic model and Howie's dielectric excitation theory model. The uncertainty associated with the theoretical curves (shaded regions I, II and III) in Figure 4.3 corresponds to the range of Si resistivity 1-20 Ωcm. The shaded region for Hasselbach's experimental fit (IV) corresponds to the published experimental uncertainty [16].

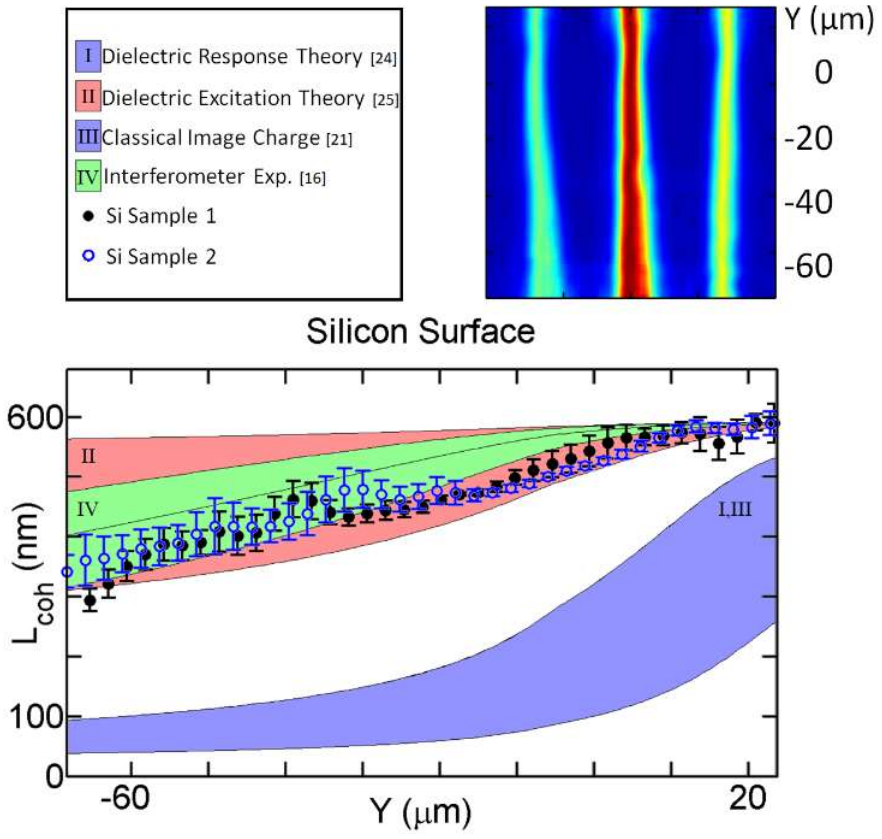

Figure 4.3 Transverse Coherence Length for a Silicon Surface. The diffraction pattern shows a loss of contrast as the diffraction peaks broaden for electrons that passed closer to the surface (top right). Our experimental findings (dotted) show agreement with Hasselbach's experimental fit (IV, green), and is consistent with modelling based on dielectric excitation theory (II, red). The data does not agree with models based on Ohmic dissipation due to classical image charge and macroscopic quantum electrodynamic theory using dielectric response (I and III, blue).

The observed loss of contrast in doped-silicon rules out Zurek's and Scheel's decoherence models. This is in contrast to the claim made earlier that Zurek's model is in adequate agreement with experiment [16] . Even if dephasing is present in our experiment, the observed loss of contrast is much smaller than predicted by the models and therefore the conclusion remains valid. Howie's dielectric excitation model agrees with our findings.

This experiment was also carried out for the case of a gold surface, and plotted in Figure 4.4 is the transverse coherence length as a function of height. For a metal with a resistivity of 2.2×10^{-6} Ωcm [34], no reduction in contrast is measured for an electron passing close to the gold surface. This is consistent with Zurek's and Scheel & Buhman's models. Machnikowski's quantum many body model based on image charge formation significantly overestimates the loss of coherence, despite being developed for high conductivity metals such as gold. Hence, Machnikowski's model can also be ruled out as a viable decoherence mechanism.

The general lack of height-dependence of the loss of contrast can be visualized in the diffraction peak's width of the diffraction pattern remaining approximately constant (Figure 4.4 top right). This height independence of the coherence length for the case of the gold surface contrasts with that of doped silicon. This may be connected to the much smaller resistivity of gold than doped silicon. No theoretical model is currently able to explain both results.

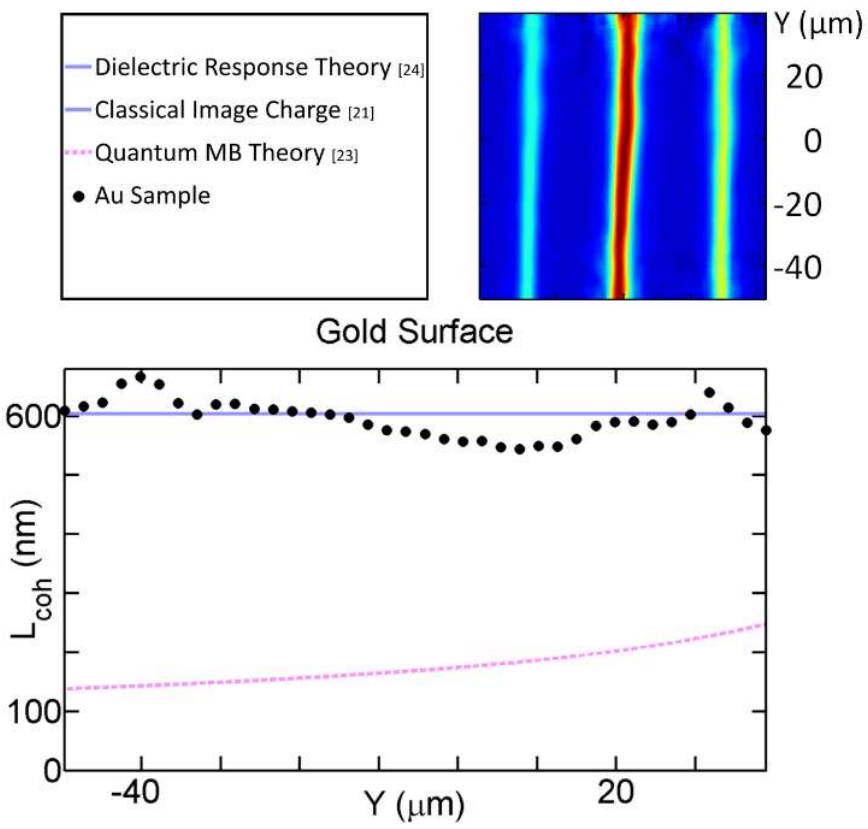

Figure 4.4: Transverse Coherence Length for a Gold Surface. For an initial coherence width of ~600 nm, no loss of coherence is observed. Here we can see that decoherence due to image charge formation in a quantum many-body electron gas model can also be ruled out.

4.5 Outlook and Conclusion

This diffractometer setup opens the door to more sensitive measurements of weak decoherence results. Consider that our modest experimental setup is limited by an initial coherence width (~600 nm) and that the decoherence factor in many cases scales as $(\Delta x)^2$. Given that it is now possible for transmission electron microscopes (TEM) to reach coherence lengths as large as 100 microns [35], we can thus improve our sensitivity by about 10^4. This opens the pathway to study decoherence surface effects due to plasmon excitation [25,26,36], optical bandgap excitation, superconductive transitions, spin

dependent transport effects [37–39], coherent thermal near-fields [40–42] blackbody-like near-fields [43,44], etc.

There has been much interest in the potential to measure the effects of decoherence due to vacuum field fluctuations in electron interference [7,8,10,45–47]. It has been shown that, absent of the surface, the decoherence factor scales with $\Gamma \approx (\Delta x)^2 / (T_{flight}^2 c^2)$ where T_{flight} is the total time of flight of the electron [8,47]. Given that Δx is generally between 100 nm – 100 μm and T_{flight} is roughly between 1 ns – 100 ns, this corresponds to a transverse velocity of $v_T \equiv \Delta x / T_{flight} \approx 10^5$ m/s and a decoherence factor of $\approx 10^7$, which is not currently feasible to observe. To observe such decoherence, the transverse velocity has to be increased by changing the experimental configuration (for example as in an quantum electron microscope [48,49]).

In conclusion, we have confirmed the loss of contrast in an electron diffraction pattern due to the introduction of a doped silicon surface with a strength consistent with Sonnentag and Hasselbach's biprism interferometer experiment. Our diffractometer setup is simpler in terms of its components and is particularly advantageous in observing weak decoherence effects. Thus, we have shown a new pathway to observe weak decoherence channels, including vacuum field decoherence. Additionally, for the case of a gold surface we have placed an upper bound on the loss of contrast that can be attributed to decoherence. Combining our silicon and gold decoherence results, it has been confirmed that the observed effect is strongly material dependent. We have ruled out a range of decoherence models due image charge based on classical theory [22], quantum many body theory [23], and dielectric theory [24]. For the materials and electron beam parameter range studied, our work remains consistent with decoherence effects due to

dielectric excitation theory from effects including surface plasmons [25,26]. These findings are consistent with the general decoherence program [1,2,13].

4.6 Chapter 4 Bibliography

[1] W. H. Zurek, Rev. Mod. Phys. **75**, 715 (2003).

[2] D. Giulini, E. Joos, C. Kiefer, J. Kupsch, I.-O. Stamatescu, and H. D. Zeh, *Decoherence and the Appearance of a Classical World in Quantum Theory* (Springer-Verlag, Berlin Heidelberg, 1996).

[3] S. Kuhr, S. Gleyzes, C. Guerlin, J. Bernu, U. B. Hoff, S. Deléglise, S. Osnaghi, M. Brune, J.-M. Raimond, S. Haroche, E. Jacques, P. Bosland, and B. Visentin, Appl. Phys. Lett. **90**, 164101 (2007).

[4] H. J. Lewandowski, J. M. McGuirk, D. M. Harber, and E. A. Cornell, Phys. Rev. Lett. **91**, 240404 (2003).

[5] T. Pellizzari, S. A. Gardiner, J. I. Cirac, and P. Zoller, Phys. Rev. Lett. **75**, 3788 (1995).

[6] W. G. Unruh, Phys. Rev. A **51**, 992 (1995).

[7] L. H. Ford, Phys. Rev. D **47**, 5571 (1993).

[8] H.-P. Breuer and F. Petruccione, Phys. Rev. A **63**, 032102 (2001).

[9] Y. Levinson, J. Phys. Math. Gen. **37**, 3003 (2004).

[10] J.-T. Hsiang and L. H. Ford, Int. J. Mod. Phys. A **24**, 1705 (2009).

[11] R. Penrose, Gen. Relativ. Gravit. **28**, 581 (1996).

[12] I. Pikovski, M. Zych, F. Costa, and Č. Brukner, Nat. Phys. **advance online publication**, (2015).

[13] M. Schlosshauer, Rev. Mod. Phys. **76**, 1267 (2005).

[14] M. Brune, E. Hagley, J. Dreyer, X. Maître, A. Maali, C. Wunderlich, J. M. Raimond, and S. Haroche, Phys. Rev. Lett. **77**, 4887 (1996).

[15] C. J. Myatt, B. E. King, Q. A. Turchette, C. A. Sackett, D. Kielpinski, C. A. Sackett, C. J. Myatt, D. J. Wineland, C. Monroe, B. E. King, W. M. Itano, Q. A. Turchette, W. M. Itano, C. Monroe, and D. J. Wineland, Nature **403**, 269 (2000).

[16] P. Sonnentag and F. Hasselbach, Phys. Rev. Lett. **98**, 200402 (2007).

[17] J. R. Friedman, V. Patel, W. Chen, S. K. Tolpygo, and J. E. Lukens, Nature **406**, 43 (2000).

[18] K. Hornberger, S. Uttenthaler, B. Brezger, L. Hackermüller, M. Arndt, and A. Zeilinger, Phys. Rev. Lett. **90**, 160401 (2003).

[19] L. Hackermüller, K. Hornberger, B. Brezger, A. Zeilinger, and M. Arndt, Nature **427**, 711 (2004).

[20] D. Akoury, K. Kreidi, T. Jahnke, T. Weber, A. Staudte, M. Schöffler, N. Neumann, J. Titze, L. P. H. Schmidt, A. Czasch, O. Jagutzki, R. A. C. Fraga, R. E. Grisenti, R. D. Muiño, N. A. Cherepkov, S. K. Semenov, P. Ranitovic, C. L. Cocke, T. Osipov, H. Adaniya, J. C. Thompson, M. H. Prior, A. Belkacem, A. L. Landers, H. Schmidt-Böcking, and R. Dörner, Science **318**, 949 (2007).

[21] J. R. Anglin, J. P. Paz, and W. H. Zurek, Phys. Rev. A **55**, 4041 (1997).

[22] J. R. Anglin and W. H. Zurek, in *Dark Matter Cosmol. Quantum Meas. Exp. Gravit.* (Editions Fronti'eres, Gif-sur-Yvette, France, Les Arcs, Savoie, France, 1996), pp. 263–270.
[23] P. Machnikowski, Phys. Rev. B **73**, 155109 (2006).
[24] S. Scheel and S. Y. Buhmann, Phys. Rev. A **85**, 030101 (2012).
[25] A. Howie, Ultramicroscopy **111**, 761 (2011).
[26] A. Howie, J. Phys. Conf. Ser. **522**, 012001 (2014).
[27] F. Röder and A. Lubk, Ultramicroscopy **146**, 103 (2014).
[28] G. Gronniger, B. Barwick, H. Batelaan, T. Savas, D. Pritchard, and A. Cronin, Appl. Phys. Lett. **87**, 124104 (2005).
[29] B. Barwick, G. Gronniger, L. Yuan, S.-H. Liou, and H. Batelaan, J. Appl. Phys. **100**, 074322 (2006).
[30] R. Bach, D. Pope, S.-H. Liou, and H. Batelaan, New J. Phys. **15**, 033018 (2013).
[31] M. Itano, F. W. Kern, M. Miyashita, and T. Ohmi, IEEE Trans. Semicond. Manuf. **6**, 258 (1993).
[32] A. D. Cronin, J. Schmiedmayer, and D. E. Pritchard, Rev. Mod. Phys. **81**, 1051 (2009).
[33] H.-P. Breuer and F. Petruccione, in *Theory Open Quantum Syst.* (Oxford University Press, 2002), pp. 232–242.
[34] C. Kittel, *Introduction to Solid State Physics*, 7th ed. (John Wiley & Sons, Inc., Canada, 1996).
[35] G. Guzzinati, A. Béché, H. Lourenço-Martins, J. Martin, M. Kociak, and J. Verbeeck, Nat. Commun. **8**, ncomms14999 (2017).
[36] F. J. García de Abajo, Phys. Rev. Lett. **102**, 237401 (2009).
[37] Y. Otani, M. Shiraishi, A. Oiwa, E. Saitoh, and S. Murakami, Nat. Phys. **advance online publication**, (2017).
[38] B. Dlubak, M.-B. Martin, C. Deranlot, B. Servet, S. Xavier, R. Mattana, M. Sprinkle, C. Berger, W. A. De Heer, F. Petroff, A. Anane, P. Seneor, and A. Fert, Nat. Phys. **8**, 557 (2012).
[39] G. Shi and E. Kioupakis, Nano Lett. **15**, 6926 (2015).
[40] K. Joulain, J.-P. Mulet, F. Marquier, R. Carminati, and J.-J. Greffet, Surf. Sci. Rep. **57**, 59 (2005).
[41] J.-J. Greffet and C. Henkel, Contemp. Phys. **48**, 183 (2007).
[42] A. I. Volokitin and B. N. J. Persson, Rev. Mod. Phys. **79**, 1291 (2007).
[43] S.-A. Biehs, M. Tschikin, and P. Ben-Abdallah, Phys. Rev. Lett. **109**, 104301 (2012).
[44] S.-A. Biehs, S. Lang, A. Y. Petrov, M. Eich, and P. Ben-Abdallah, Phys. Rev. Lett. **115**, 174301 (2015).
[45] J. Schwinger, Phys. Rev. **152**, 1219 (1966).
[46] F. D. Mazzitelli, J. P. Paz, and A. Villanueva, Phys. Rev. A **68**, 062106 (2003).
[47] J.-T. Hsiang and D.-S. Lee, Phys. Rev. D **73**, 065022 (2006).
[48] W. P. Putnam and M. F. Yanik, Phys. Rev. A **80**, 040902 (2009).
[49] P. Kruit, R. G. Hobbs, C.-S. Kim, Y. Yang, V. R. Manfrinato, J. Hammer, S. Thomas, P. Weber, B. Klopfer, C. Kohstall, T. Juffmann, M. A. Kasevich, P. Hommelhoff, and K. K. Berggren, Ultramicroscopy **164**, 31 (2016).

CHAPTER 5

DEPHASING VS DECOHERENCE

5.1 Introduction

The nature of the loss of quantum interference can be generally separated into two distinct processes: dephasing and decoherence. The effect of these two processes on interference can be seen most clearly in the combined phenomenological expression of the detected probability distribution of two matter-wave states of equal shape [1,2],

$$I\left(x,t_f\right)=\left|\psi_1\left(x,t_f\right)\right|^2+\left|\psi_2\left(x,t_f\right)\right|^2+ 2\,\mathrm{Re}\left(e^{-\Omega\left(x,t_f\right)}e^{-i\Theta\left(x,t_f\right)}\psi_1^*\left(x,t_f\right)\psi_2\left(x,t_f\right)+\mathrm{H.O.T.}\right). \tag{5.1}$$

The third term, which is responsible for the interference, can be modulated by both the dephasing term $e^{-i\Theta\left(x,t_f\right)}$ (which represents the net distortion from dephasing) and the decoherence term $e^{-\Omega\left(x,t_f\right)}$ (which represents the net distortion from decoherence) where both Θ and Ω are real numbers. Notice the similarity between equation 5.1 and equation 2.15. Ignoring the higher order corrections, it turns out that the visibility (a measure of the loss of contrast) in this case is $V=e^{-\Omega\left(x,t_f\right)}\,\mathrm{Re}\left(e^{-i\Theta\left(x,t_f\right)}\right)$. Thus, we can see that both processes can contribute to loss of visibility.

There is however an important physical difference between the two processes. Decoherence, which involves entanglement with its environment as described in section 2.5.1, is time-irreversible. Dephasing on the other hand, which involves phase modulation of the wavefunction from an external field, is time-reversible. Unfortunately, visibility, transverse coherence length, and similar measures do not adequately distinguish between

decoherence and dephasing. This makes it hard to identify sources of contrast loss by looking at the intensity distribution alone, and thus take appropriate measures to reduce such a loss of contrast (if the goal is to observe high-contrast diffraction or interference).

As this relates to the work presented in Chapter 4, a separate issue is revealed. When probing the transition from quantum to classical mechanics, or testing if one is introducing a decoherence environment to a system, it is not immediately evident that the observed loss of contrast is due to decoherence rather than dephasing using the measurement methods that are traditionally utilized. This opens the concerning question of whether or not the transition between quantum and classical transition has actually been observed, or if only upper limits to the loss of coherence have been observed.

One straightforward way to determine if a process is a decohering one is to evaluate the Von Neuman entropy before and after the process:

$$S = -Tr\left(\rho \ln \rho\right) \tag{5.2}$$

When S remains constant in time, the process is time-reversible; as opposed to when S increases in time, the process is time-irreversible [3]. Calculating S however requires the determination of the off-diagonal elements (also known as the coherence terms) of the density matrix ρ. These terms cannot be determined by the intensity distribution alone, and require phase retrieval techniques such as holography or quantum state tomography [4].

Instead of relying on such methods, we have proposed using the method based on reptetitive ensemble measurements of the propability distribution using a spatial correlation function of these measurements. Without knowing the details of the dephaser (defined as a process which causes dephasing), we show that, with repetitive

measurement, this method can in principle extract sufficient information to recover the loss of contrast in the diffraction pattern. Conversely, we show that the same process fails to recover the loss of contrast in the diffraction pattern for the case of a decoherer (defined as a process which causes decoherence).

5.2 Path Integral Simulation with Dephasing and Decoherence

The following is a description of how, via the path integral formalism [5,6], a discrete numerical simulation is used to produce a far field diffraction pattern after the electron undergoes either dephasing or decoherence in the near field of a diffraction grating (particularly right before the grating). The final density matrix in both cases is recorded. This is initially pursued to demonstrate that we can create a decoherer and a dephaser that both produce a similar loss of contrast. But by calculating the entropy before and after the simulation, it is shown that the dephaser does not change the entropy of the system while the decoherer increases the entropy in the system.

The parameters used are inspired by the decoherence experiment's setup outlined in Chapter 4. Figure 5.1 is a rough sketch of the grid planes used in this computation. These grid points are the positions considered in a finite sum approximation of the path integrals. Only straight-lined paths are considered, and an azimuthal equal velocity approximation is used for the propagator [6].

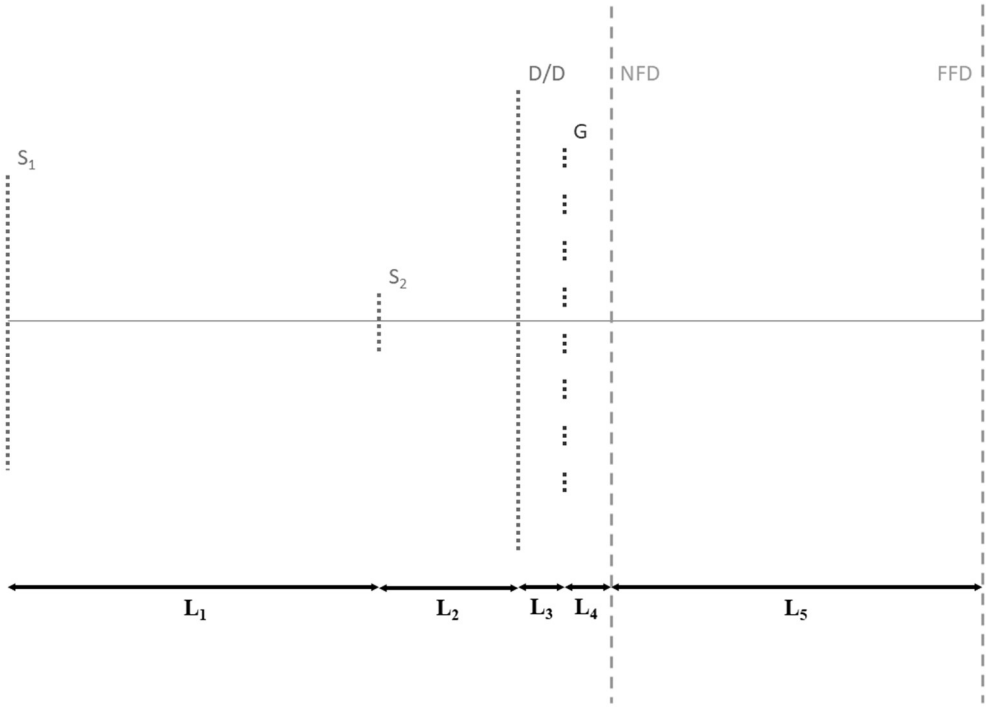

Figure 5.1: Diagram of Dephaser/Decoherer Path Integral Setup 1. A matter wave (with the mass of an electron and an energy of 1.67 *keV*) propogates incoherently from the first collimation slit (S_1) to a second slit, (S_2) to a plane acting as a dephaser or decoherer (D/D). It then propagates through a grating (G) and finally to the near field (NFD) and the Far Field (FFD).

Beginning with the dephaser case, the wavefunction propagates from one individual grid point x_k at the plane representing slit 1 to the plane for slit 2 with grid points x_j to produce the wave function ψ_2 at slit 2:

$$\psi_2^k\left(x_j\right)=\sum_j e^{-i2\pi\sqrt{\left(x_j-x_k\right)^2+L_1^2}\,/\,\lambda_{dB}}\,, \tag{5.3}$$

where $\lambda_{dB}=h/mv$ is the de-Broglie wavelength of the electron.

The wave propagation continues from the second slit from each grid point x_i to the plane representing the dephaser with grid points x_j:

$$\psi_3^k\left(x_j\right)=\sum_j e^{i\theta_n\left(x_j\right)}\sum_i \psi_2^k\left(x_i\right)e^{-i2\pi\sqrt{\left(x_j-x_i\right)^2+L_2^2}\,/\,\lambda_{dB}}\,. \tag{5.4}$$

A phase $e^{i\theta_n(x_j)}$ is applied to the wave function ψ_3 to act as the dephaser, a piecewise step function which consists of *N* blocks of 241 nm size with constant phase angle θ_n uniformly randomly varying from 0 to 2πc (see Figure 5.2). The quantity c is a tunable variable used to tune the dephasing effect; i.e. $0 \leq c \leq 1$ where c=0 means no dephasing and c=1 is maximal dephasing for this step function dephasing pattern.

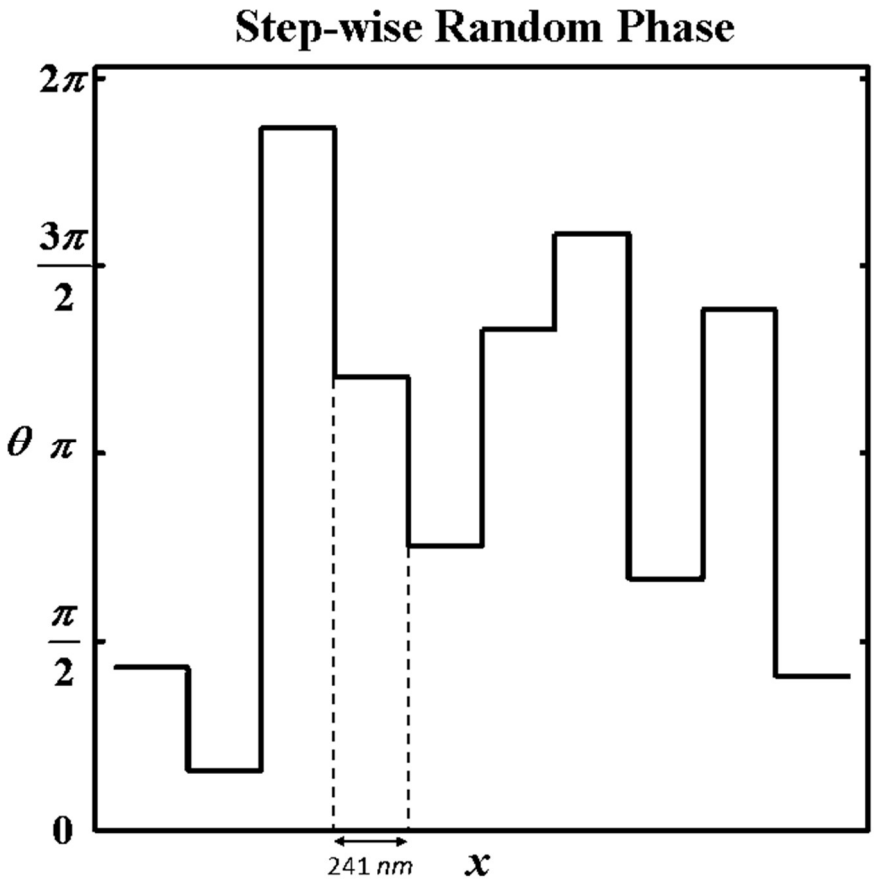

Figure 5.2. Step-wise Defined Random Phase used in Dephaser.

Next, the wave propagates to a grating at a distance of L_3 away, which has a 50/50 transmission ratio and 100 nm periodicity:

$$\psi_4^k\left(x_j\right)=\sum_j H\left(x_j\right)\sum_i \psi_3^k\left(x_i\right)e^{-i2\pi\sqrt{\left(x_j-x_i\right)^2+L_3^2}\,/\,\lambda_{dB}}\,,$$

(5.5)

where $H\left(x_j\right)$ represents the grating function (equals 1 for transmission and equals 0 for no transmission). Then the propagation continues to the near field of the grating, and then to the far field "detector":

$$\begin{aligned}\psi_5^k\left(x_j\right)&=\sum_j\sum_i \psi_4^k\left(x_i\right)e^{-i2\pi\sqrt{\left(x_j-x_i\right)^2+L_4^2}\,/\,\lambda_{dB}}\\ \psi_6^k\left(x_j\right)&=\sum_j\sum_i \psi_5^k\left(x_i\right)e^{-i2\pi\sqrt{\left(x_j-x_i\right)^2+L_5^2}\,/\,\lambda_{dB}}.\end{aligned}$$

(5.6)

With such a wave function, initially from the k[th] grid point at the first slit, we construct a density matrix (which is at the moment still coherent):

$$\rho_k=\left|\psi_6^k\right\rangle\left\langle\psi_6^k\right|.$$

(5.7)

We repeat the above series of path integrals for all N_s grid-points at the source and sum the density matrices together (and then normalize) to obtain the final density matrix associated with dephasing:

$$\rho_{final}^{deph}=\sum_k \rho_k=\sum_{k=1}^{N_s}\left|\psi_6^k\right\rangle\left\langle\psi_6^k\right|.$$

(5.8)

For our initial modeling of decoherence instead of dephasing, this path integral system is somewhat modified. The initial propagation from the first to second slit outlined in equation (5.3) remains the same. However, from the second slit to the decoherer, the only paths considered are those whose final position lies within the n^{th} 241 *nm* piecewise step section of the random phase. $e^{i\theta_n\left(x_j\right)}$ are initially integrated over:

$$\psi_3^{k,n}\left(x_j^{(n)}\right)=\sum_{j\in n}e^{i\theta_n\left(x_j^{(n)}\right)}\sum_i\psi_2^k\left(x_i\right)e^{-i2\pi\sqrt{\left(x_j^{(n)}-x_i\right)^2+L_2^2}\Big/\lambda_{dB}}\quad. \tag{5.9}$$

Then from this smaller portion of the dephasing plane, the wave function continues to propagate to the grating:

$$\psi_4^{k,n}\left(x_j\right)=\sum_j H\left(x_j\right)\sum_{i\in n}\psi_3^{k,n}\left(x_i^{(n)}\right)e^{-i2\pi\sqrt{\left(x_j-x_i^{(n)}\right)^2+L_3^2}\Big/\lambda_{dB}}, \tag{5.10}$$

and as before to the near and far field:

$$\begin{aligned}\psi_5^{k,n}\left(x_j\right)&=\sum_j\sum_i\psi_4^{k,n}\left(x_i\right)e^{-i2\pi\sqrt{\left(x_j-x_i\right)^2+L_4^2}\Big/\lambda_{dB}}\\ \psi_6^{k,n}\left(x_j\right)&=\sum_j\sum_i\psi_5^{k,n}\left(x_i\right)e^{-i2\pi\sqrt{\left(x_j-x_i\right)^2+L_5^2}\Big/\lambda_{dB}}.\end{aligned} \tag{5.11}$$

Note as before that while up to this point a fully coherent density matrix has been produced,

$$\rho_{k,n}=\left|\psi_6^{k,n}\right\rangle\left\langle\psi_6^{k,n}\right|, \tag{5.12}$$

to produce the desired final density matrix representative of the entire process, not only does one add incoherently over all initial states at the source, but also over all N_p piecewise segments of the decoherer. Thus:

$$\rho_{final}^{dec}=\sum_n\sum_k\rho_{k,n}=\sum_{n=1}^{N_p}\sum_{k=1}^{N_s}\left|\psi_6^{k,n}\right\rangle\left\langle\psi_6^{k,n}\right|. \tag{5.13}$$

A summary of the parameters used in this path integral simulation to test the results of the effects of this dephaser and decohere can be found in Appendix E.1.

One can then determine the coherence length and time associated with the resulting probability distribution by taking the diagonal of the final density matrix,

$$P_{final}\left(x\right)=diag\left(\rho_{final}\right). \tag{5.14}$$

The von Neumann entropy S (Equation 5.2) of the system can be determined using the final density matrix, most easily by first determining the eigenvalues of the density matrix:

$$S = -\sum_i \lambda_i \ln(\lambda_i). \tag{5.15}$$

The results of doing this can be seen in Figure 5.3, where plotted is the final intensity distribution for the cases of 1) no dephasing/decoherence, 2) with dephasing, and 3) with decoherence (where c= .7). The coherence length in both cases decreases, and they decreased by roughly the same amount (although the spatial noise for the dephasing case is much larger). However, from the second slit to the detector, the change in entropy ΔS=0 for the case of dephasing, whereas the entropy increases by ΔS= 2.7444 for the case of decoherence.

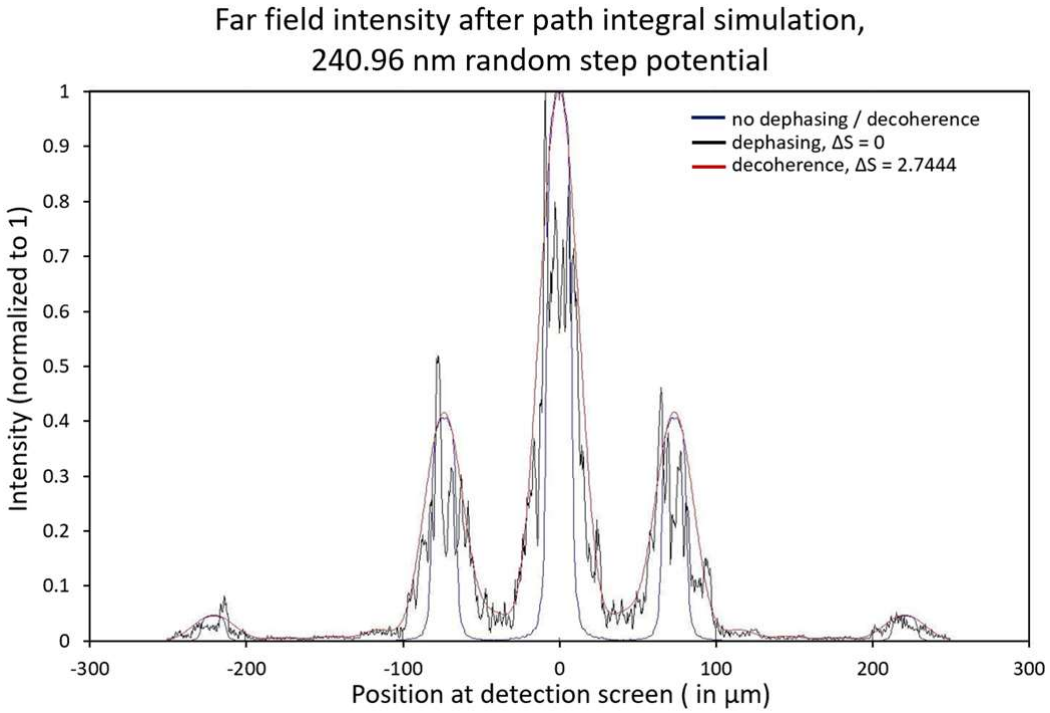

Figure 5.3: Far Field Intensity Pattern After Dephasing/Decoherence Path Integral Simulation 1. Adding dephasing and decoherence in this case widens the diffraction peaks, but in the case of decoherence, the entropy increases.

It can be concluded that these two processes both can alter the interference properties in proximally the same way (and if one were to take a time average, they can become indistinguishable). However, given that entropy changes in one case and remains unchanged in another, the underlying physical mechanisms indeed differ in a very fundamental way which further motivates a means to distinguish what is the cause of the lost of contrast post-diffraction/interference.

5.3 Spatial Correlation Method

Previously, Rui-Feng et al [7] developed an optical experiment that exhibited a loss of contrast. They managed to recover the diffraction pattern after it was severely distorted using a ground glass disk. Here they shined a red 632.8 nm laser through a double slit with a slit separation of d = 1.5 mm and slit width of .5 mm; and imaged the diffraction pattern at a distance z = 20 cm in the Fresnel diffraction region ($z \ll z_0;\ z_0 = d^2/\lambda \approx 3.5$ m) using a CCD camera (see Figure 5.4). They also placed a ground glass disk spinning at .5 Hz very close and right before the double slit. When the ground glass disk was removed, the camera can clearly image the near field pattern with its two main lobes and smaller near field fringes present. When the ground glass disk is present, the image is completely blurred away and no information seems to remain.

As the ground glass disk was spinning, images of the pattern were taken with a 100 μs exposure time for each image. After collecting many blurred images (~1000), they used a second order correlation function on the spatial intensity distribution according to;

$$g^{(2)}(x,-x) = \langle I(x) I(-x) \rangle \Big/ \Big(\langle I(x) \rangle \langle I(-x) \rangle \Big). \tag{5.16}$$

where $I(x)$ represents the intensity measured on the CCD camera at position x in the direction of diffraction (taking x=0 to be the center of the camera) and $\langle ... \rangle$ here representing averaging over time (or averaged over the frames). The result of this is was the production of a second order correlation pattern which closely fits the Fraunhofer pattern, or expected far field diffraction pattern that matched the experimental parameters. The implication of this is that after Fourier transformation of the second order correlation function, the original near field pattern can be restored.

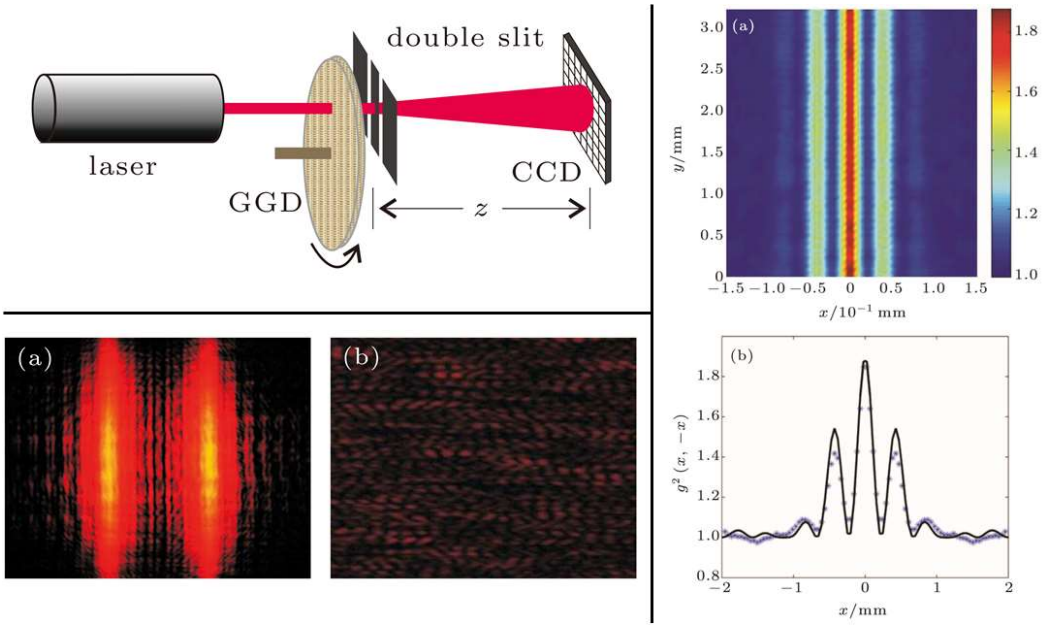

Figure 5.4: Ghost Imaging Experiment by Rui-Feng et al [7]. Top Left: Experimental setup. Laser light is sent through a rotating ground glass disk (GDD) before passing through a double slit and being imaged by a CCD camera in the Fresnel diffraction region. Bottom Left: (a) The Near Field intensity image obtained when no GGD is present shows a coherent diffraction pattern as opposed to (b) a blurred or scrambled pattern when the GDD is present. Right: by calculating the second order correlation function $g^{(2)}(x,-x)$ from a series of blurred images, the far field diffraction pattern is obtained. Thus after a Fourier transformation of $g^{(2)}(x,-x)$, near field information can be restored. See text and [3] for more details. Photos credited to Rui-Feng et al [7].

It is from this work we asked the questions: given that it appears that scrambled information is restored, is this recovery procedure one method to discern if the process which causes loss of contrast (the laser light passing through the ground glass disk) reversible or irreversible, and thus a dephasing or decoherence process? Under what parameters is this diffraction reconstitution possible? Can such a method be extended to matter-wave optics?

To test this, the path integral simulation described in Section 5.2 was modified so diffraction of the electron occurs through a double-slit rather than a grating, resembling the double slit experiment by Bach et al [8] (see Figure 5.5). An electron energy of 1.67 keV is used, but it starts with a fully coherent plane wave which propagates to the double-slit. The dephasing/decoherence takes place at the double-slit plane rather than being a separate plane (i.e. there is no propagation between the dephasing/decohering plane and the double slit). It then propagates to the far field detection plane. A summary of the parameters can be found in Appendix E.2.

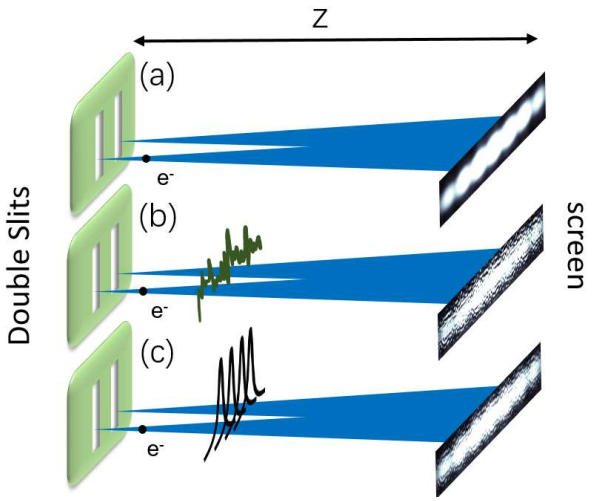

Figure 5.5. Dephasing vs Decoherence for the Case of the Double Slit. Sketch of the double slit setup when (a) no dephasing or decoherence is present, (b) dephasing and (c) decoherence. Contrast is lost for the cases of dephasing and decoherence. For dephasing, a random potential (dark green curve) using a sum of gaussians is used at the double-slit. For decoherence, the electron wave is split into a probabilistic sum of separate incoherent overlapping gaussian waves,

and then propagate to the detector. The diffraction image produced in (a) is attributed to Bach et al [8]. Image produced by Zilin Cheng.

For the dephaser, instead of the phase distribution being in the form of a step function, the spatial dephasing pattern used in $e^{i\theta(x)}$ is a smooth sum of 500 Gaussians:

$$\theta(x) = \sum_i A_i e^{-\left(x - x_i/\sigma_i\sqrt{2}\right)^2}, \tag{5.17}$$

where each A_i and x_i are evenly distributed random numbers within the intervals [0, 2π] and [-250 nm, 250 nm] respectively. σ_i is a normally distributed random number with a mean value of 4 nm and a standard deviation of .5 nm.

For the case of decoherence, the same phase is applied, and as before the wave propagation is cut into finite segments. However, this time instead of being cut into boxed segments, the entire double-slit plane is propagated over, and this time the wave function is truncated by a Gaussian function (similar to a grating or double slit function):

$$\psi^n(x) = \psi_d(x) A_0 e^{-\left(x - x_n/\sigma_0\sqrt{2}\right)^2}, \tag{5.18}$$

where $\psi_d(x)$ is the wave function at the double-slit with the smooth random phase pattern, $A_0 = 2.23\times10^{-5}$ and $\sigma_0 = 100/\sqrt{2}$ nm are constant values of and x_n is the central position of the Gaussian. For the incoherent adding of the path integrals (addition like before), the distance between these Gaussians is 12.5 nm and there are 40 Gaussians total. Plotted in Figure 5.6 are representative images of the random phase pattern and the summed Gaussians at the double slit plane.

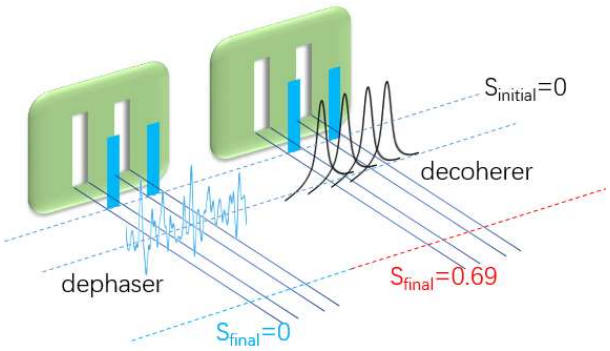

Figure 5.6: Representative Diagram of Dephasing and Decoherence at the Double Slit. the vertical lines show the range of the double slit. Left: the blue curve represents the smooth random phase also applied to the electron wave function. Right: example Gaussian distributions applied to the wave function. Note that here the widths and separations are not drawn to scale. For dephasing, only the phase is applied to the wave function while for decoherence both the random phase and (incoherently added) the Gaussians are applied to the electron wave function at the double slit. For both cases the diffraction pattern was calculated 500 times each with a different set of random numbers. Image Produced by Zilin Cheng

This path integral simulation was carried out 500 times for both the dephasing and decoherence case each with a different smooth random phase applied. Plotted in Figure 5.7 are example intensity distributions as measured in the far field after dephasing/decoherence, and the time average of each. Individually, although the main structure is maintained for both dephasing and decoherence (in that the position and relative amplitudes of the maxima are similar in both cases), a lower contrast for the case of decoherence is a distinguishing difference for a single iteration. For the near field, this is consistent with the work done by P Kazemi et. al [9]. where they were able to see interference minima for the case of dephasing in the near field quantum carpet, but not for the case of decoherence.

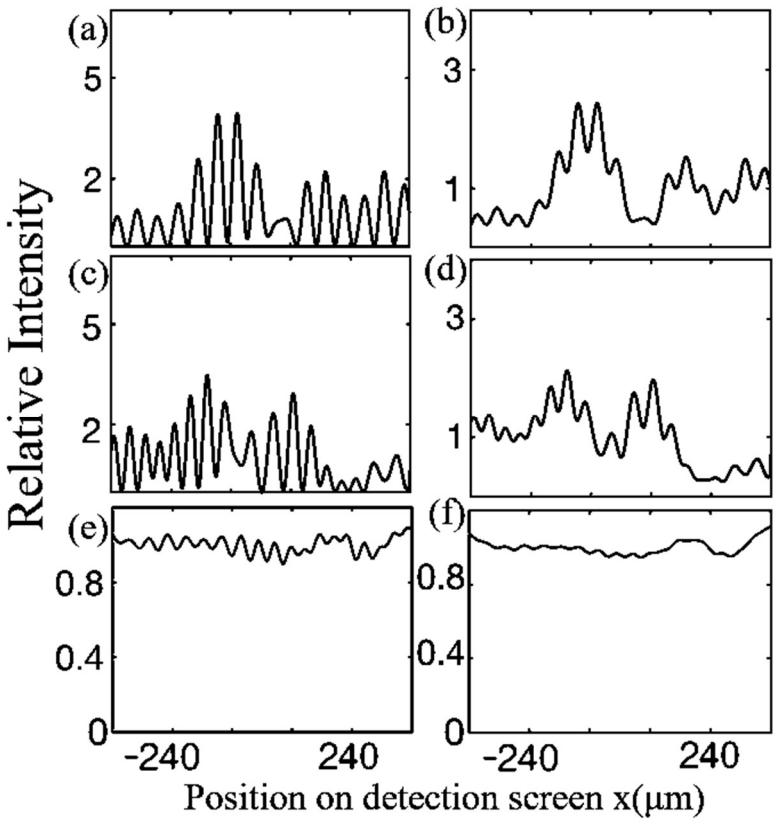

Figure 5.7 Example Intensity Distributions in Far Field after Dephasing/Decoherence for Path Integral Setup 2. (a) and (b) shows the intensity distribution resulting after experiencing the first random phase after dephasing and decoherence, respectively. (c) and (d) shows the intensity distribution after a different smooth random phase. Individually, although the main structure is maintained for both dephasing and decoherence (in that the position and relative amplitudes of the maxima are similar in both cases), a lower contrast for the case of decoherence is a distinguishing difference for a single iteration. (e) and (f) show the intensity distributions averaged over 500 iterations with different random phases. Image produced by Zilin Cheng.

Similar to Rui-Feng et al [7], the deviation of the correlation function (equation 5.16) is applied to the sequence of 500 phase-independent final intensity distributions in the far field for both the dephasing and decoherence simulations:

$$\begin{aligned}&\Delta G^{(2)}\left(x_1,x_2\right)\\&\equiv G^{(2)}\left(x_1,x_2\right)-\left\langle I_1\left(x_1\right)\right\rangle\left\langle I_2\left(x_2\right)\right\rangle\\&\equiv\left|\int dx_1'dx_2'h_1^*\left(x_{1,}x_1'\right)h_2\left(x_{2,}x_2'\right)G^{(1)}\left(x_1',x_2'\right)\right|^2.\end{aligned}\tag{5.19}$$

The result is a recovery of what resembles the far field diffraction pattern for the case of dephasing and what appears to be only a single peak for the case of decoherence.

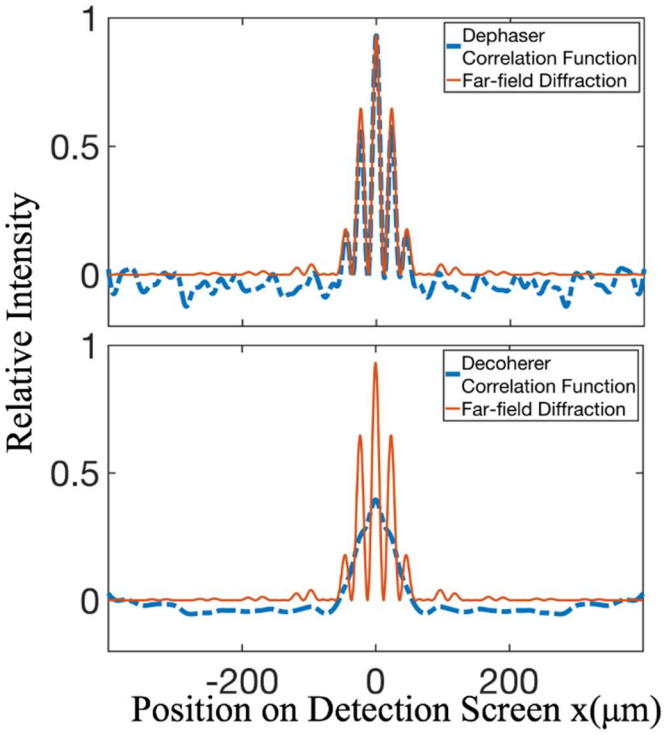

Figure 5.8: Effects of Applying the Correlation Function $g^{(2)}(x,-x)$. When applying the correlation function $g^{(2)}(x,-x)$ to 500 independent iterations of the far field pattern where (top) is the computation after dephasing while (bottom) is the computation after decoherence. The correlation function successfully recovers a far field pattern in the case of dephasing while this is not the case for decoherence. the diffraction pattern fits the dephaser correlation pattern well, as the peak to peak distance of the correlation pattern (22.9 μm) is close to the expected diffraction peak to peak distance (24 μm). Image produced by Zilin Cheng.

Fitting the autocorrelation pattern that came out of the dephasing case with the double slit diffraction pattern function in the small angle approximation (See Figure 5.8),

$$I(x) = 4w^2 \mathrm{sinc}^2\left(\frac{wx}{\lambda L_2}\right)\cos^2\left(\frac{\pi dx}{\lambda L_2}\right), \tag{5.20}$$

where w is the width of the slits (50 nm), d is the distance between slits (150 nm), λ is the electron's wavelength (30.03 pm), and L_2 is the distance from the double slit to the far

field screen (24 cm). Using these values, one would expect that the distance between local maxima (peaks) in the diffraction pattern would be 24 μm, and the best fit to our autocorrelation pattern is a peak to peak distance of 22.9 μm.

The change in entropy from the source to detection for both dephasing and decoherence cases is computed as before (equation 5.15). For dephasing, the change of entropy was $\Delta S = 0$ whereas for decoherence the change in entropy was increased by $\Delta S = 0.69$, as anticipated from the previous path integral results. This reaffirms that the dephasing processes we theoretically realized are time-reversible while the decoherence process is time-irreversible.

5.4 Conclusion and Outlook

In conclusion, we have reaffirmed using a matter wave path integral simulation that the distinguishing feature between dephasing and decoherence processes is that the former is a time-reversible process and the latter is a time-irreversible process. Although the resulting time-averaged intensity patterns can be difficult to distinguish from each other at first glance, by taking a sequence of measurements and performing an autocorrelation computation on the time sequence in the far field, we have found that a far field diffraction pattern showing high visibility interference can indeed be recovered for the dephasing case, but not for the decoherence case. This result agrees with the experimental results by Stibor et. al. [10,11].

Questions arise regarding how to realize this result experimentally. Primarily, how fast does the imaging of individual patterns need to be in order to recover the far field diffraction pattern for the case of dephasing? Although in Stibor's case they

managed to recover interference by taking the positions of nearest and next nearest neighbors of individual electron events, they used a known modulation frequency. Therefore, we hypothesize that so long as the characteristic dephasing time (between modulations or "sequences") is sufficiently slow compared to the time to accumulate a statistically significant far field pattern, a diffraction pattern should be recoverable.

Besides these results having implications for fundamental studies including thermodynamics in quantum mechanics and experiments pertaining to the quantum measurement problem, this process has application as a diagnostic tool for determining the source of loss of contrast in imaging involving diffraction such as transmission electron microscopy (TEM) and improving contrast in long-exposure images that are distorted over time.

5.5 Chapter 5 Bibliography

[1] Y. Levinson, J. Phys. Math. Gen. **37**, 3003 (2004).
[2] P. Machnikowski, Phys. Rev. B **73**, 155109 (2006).
[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2011).
[4] G. Mauro D'Ariano, M. G. A. Paris, and M. F. Sacchi, in *Adv. Imaging Electron Phys.*, edited by P. W. Hawkes (Elsevier, 2003), pp. 205–308.
[5] R. P. Feynman, Rev. Mod. Phys. **20**, 367 (1948).
[6] E. R. Jones, R. A. Bach, and H. Batelaan, Eur. J. Phys. **36**, 065048 (2015).
[7] L. Rui-Feng, Y. Xin-Xing, F. Yi-Zhen, Z. Pei, Z. Yu, G. Hong, and L. Fu-Li, Chin. Phys. B **23**, 054202 (2014).
[8] R. Bach, D. Pope, S.-H. Liou, and H. Batelaan, New J. Phys. **15**, 033018 (2013).
[9] P. Kazemi, S. Chaturvedi, I. Marzoli, R. F. O'Connell, and W. P. Schleich, New J. Phys. **15**, 013052 (2013).
[10] A. Rembold, G. Schütz, W. T. Chang, A. Stefanov, A. Pooch, I. S. Hwang, A. Günther, and A. Stibor, Phys. Rev. A **89**, 033635 (2014).
[11] A. Günther, A. Rembold, G. Schütz, and A. Stibor, Phys. Rev. A **92**, 053607 (2015).

CHAPTER 6

CONCLUSIONS AND OUTLOOK

6.1 Optical-Electron Switch

Despite issues with having a complete understanding of the physical mechanism of the laser-induced surface charge redistribution which causes the deflection of the free electron, the setup still shows utility in applications such as electron beam lithography and microscopy where external electronics need to be kept isolated [1]. Additionally, further study in understanding the mechanism of this laser-induced potential will be needed if one were to study coherence effects after travelling over such an environment. Potential pathways of such an investigation include starting with more geometrically simpler and pristine surfaces, using a range of materials which have been well characterized, and using a more comprehensive laser intensity dependence and wavelength dependence study.

6.2 Decoherence Experiment

This diffractometer setup opens the door to more sensitive measurements of weak decoherence results. Consider that our modest experimental setup is limited by an initial coherence width (~600 nm) and that the decoherence factor in many cases scales as $(\Delta x)^2$. Given that it is now possible for transmission electron microscopes (TEM) to reach coherence lengths as large as 100 microns [2], the sensitivity can thus be improved by about 10^4. The general method of detection present here opens the pathway to study spatially dependent decoherence surface effects due to plasmon excitation [3–5], optical

bandgap excitation, superconductive transitions, spin dependent transport effects [6–8], coherent thermal near-fields [9–11], blackbody-like near-fields [12,13], etc.

In this experiment, we have confirmed the loss of contrast in an electron diffraction pattern due to the introduction of a doped silicon surface with a strength consistent with Sonnentag and Hasselbach's biprism interferometer experiment. Our diffractometer setup is simpler in terms of its components and is particularly advantageous in observing weak decoherence effects. Thus, we have shown a new pathway to observe weak decoherence channels, including vacuum field decoherence. Additionally, for the case of a gold surface we have placed an upper bound on the loss of contrast that can be attributed to decoherence. The silicon and gold decoherence results together confirm that the observed effect is strongly material dependent. We have ruled out a range of decoherence models due image charge based on classical theory [14], quantum many body theory [15], and dielectric theory [16]. For the materials and electron beam parameter range studied, our work remains consistent with decoherence effects due to dielectric excitation theory from effects including surface plasmons [4,5]. These findings are consistent with the general decoherence program [17–19].

If the goal is to study decoherence for its own sake, then there is a need for less complex and well characterized environments. Such an engineered environment reduces the number of possible decoherence channels. Fine-tuning the geometry of the material may be one way of doing this (such as low dimensional materials to reduce certain degrees of freedom and thus restricting the field modes, such as quantum dots, atomic and molecular gases, nano-wires, and low dimensional surfaces), as well as choosing

materials that suppress certain field channels and enhance others (eg thermal polaritons vs field induced polaritons), etc.

Independent of this, there is great need to study these experiments over an independent and wider parameter range. This was attempted to some degree by Sonnentag and Hasselbach (where they also varied Δx as well as y). However, in their case, they managed to confirm the small separation approximation where the decoherence factor varies with small Δx, but did not go into further detail to investigate outside this limit which is where the decoherence theories in Chapter 2 section 6 diverge (see Figure 6.1). Other important parameters besides Δx to serve the same purpose include temperature, the energy of the system, and the momentum exchange between the system and the environment.

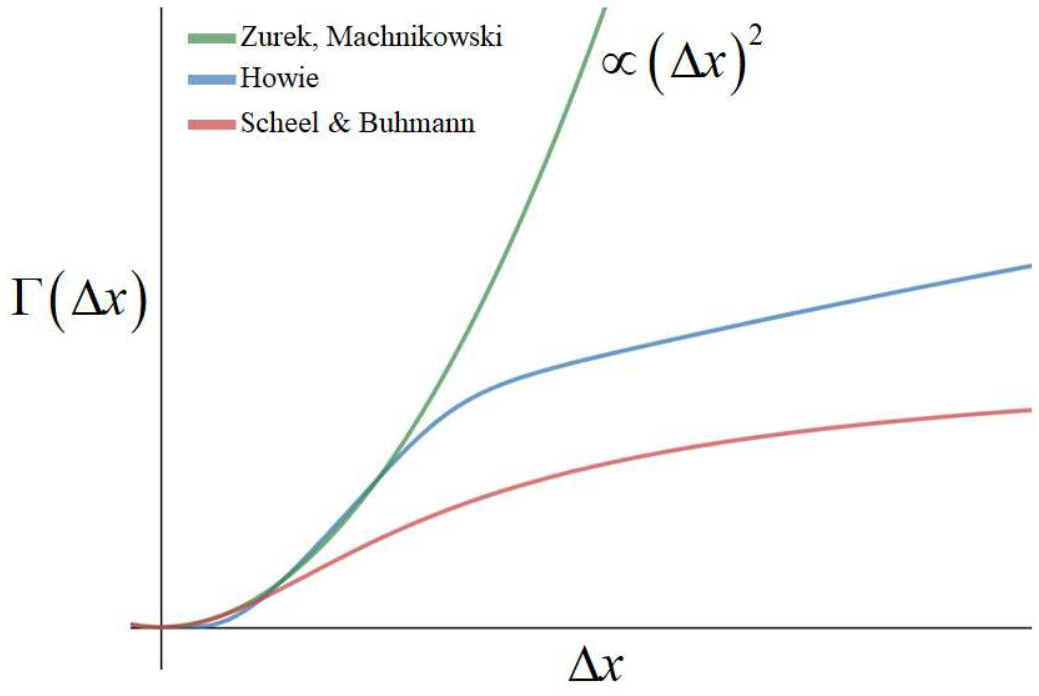

Figure 6.1: Illustration of the Functional form of the Decoherence Factor. For small Δx, these decoherence factor based on physical models described in Chapter 2 are all roughly proportional

to $(\Delta x)^2$, but diverge for larger Δx. Probing the decoherence factor over a wide enough range of Δx where such divergence is evident is an important cross-check of the validity and exclusion of any of these theories along with the check of the strength of the decoherence factor.

These kinds of decoherence experiments can serve as a great navigation tool for a vast amount of mesoscopic studies (including nanoscience, quantum information, biological and organic systems, etc) where it is not clear to what degree quantum or classical effects are taking place, and whether decoherence itself is playing a role. However, should this be a goal, these studies would be best served if the prior mentioned experiments are tailor made to investigate the precise underlying physics in question, once preliminary studies such as the ones investigated here are well understood. This general approach also applies to proposed tests of the fundamental of quantum mechanics (including nanomechanical oscillators [20], quantum optomechanics [21], tests of collapse models [22], and gravitational cat states [23–25]).

6.3 Dephasing vs. Decoherence

In this theoretical work, we have reaffirmed using a matter-wave path-integral simulation that the distinguishing feature between dephasing and decoherence processes is that the former is a time-reversible process and the latter is a time-irreversible process. Although the resulting time-averaged intensity patterns can be difficult to distinguish from each other at first glance, by taking a sequence of measurements and performing an correlation computation on the time sequence in the far field, we have found that a far field diffraction pattern, showing high visibility interference, can indeed be recovered for the dephasing case, but not for the decoherence case. This result agrees with the experimental results by Stibor et. al. [26,27].

The question arises how to realize this result experimentally. Primarily, how fast does the imaging of individual patterns need to be in order to recover the far field diffraction pattern for the case of dephasing? Although in Stibor's case they were able to recover interference by taking the positions of nearest and next nearest neighbors of individual electron events, they used a known modulation frequency. Therefore, we hypothesize that so long as the characteristic dephasing time (between modulations or "sequences") is sufficiently slow compared to the time to accumulate a statistically significant far field pattern, a diffraction pattern should be recoverable.

Besides these results having implications for fundamental studies including thermodynamics in quantum mechanics and experiments pertaining to the quantum measurement problem, this process has applications as a diagnostic tool for determining the source of loss of contrast in imaging involving diffraction such as transmission electron microscopy (TEM) and improving contrast in long-exposure images that are distorted over time.

6.4 Chapter 6 Bibliography

[1] MAPPER Lithography, Www.mapperlithography.com/.
[2] G. Guzzinati, A. Béché, H. Lourenço-Martins, J. Martin, M. Kociak, and J. Verbeeck, Nat. Commun. **8**, ncomms14999 (2017).
[3] F. J. García de Abajo, Phys. Rev. Lett. **102**, 237401 (2009).
[4] A. Howie, Ultramicroscopy **111**, 761 (2011).
[5] A. Howie, J. Phys. Conf. Ser. **522**, 012001 (2014).
[6] Y. Otani, M. Shiraishi, A. Oiwa, E. Saitoh, and S. Murakami, Nat. Phys., (2017).
[7] B. Dlubak, M.-B. Martin, C. Deranlot, B. Servet, S. Xavier, R. Mattana, M. Sprinkle, C. Berger, W. A. De Heer, F. Petroff, A. Anane, P. Seneor, and A. Fert, Nat. Phys. **8**, 557 (2012).
[8] G. Shi and E. Kioupakis, Nano Lett. **15**, 6926 (2015).
[9] K. Joulain, J.-P. Mulet, F. Marquier, R. Carminati, and J.-J. Greffet, Surf. Sci. Rep. **57**, 59 (2005).
[10] J.-J. Greffet and C. Henkel, Contemp. Phys. **48**, 183 (2007).
[11] A. I. Volokitin and B. N. J. Persson, Rev. Mod. Phys. **79**, 1291 (2007).

[12] S.-A. Biehs, M. Tschikin, and P. Ben-Abdallah, Phys. Rev. Lett. **109**, 104301 (2012).
[13] S.-A. Biehs, S. Lang, A. Y. Petrov, M. Eich, and P. Ben-Abdallah, Phys. Rev. Lett. **115**, 174301 (2015).
[14] J. R. Anglin and W. H. Zurek, in *Dark Matter Cosmol. Quantum Meas. Exp. Gravit.* (Editions Fronti'eres, Gif-sur-Yvette, France, Les Arcs, Savoie, France, 1996), pp. 263–270.
[15] P. Machnikowski, Phys. Rev. B **73**, 155109 (2006).
[16] S. Scheel and S. Y. Buhmann, Phys. Rev. A **85**, 030101 (2012).
[17] W. H. Zurek, Rev. Mod. Phys. **75**, 715 (2003).
[18] D. Giulini, E. Joos, C. Kiefer, J. Kupsch, I.-O. Stamatescu, and H. D. Zeh, *Decoherence and the Appearance of a Classical World in Quantum Theory* (Springer-Verlag, Berlin Heidelberg, 1996).
[19] M. Schlosshauer, Rev. Mod. Phys. **76**, 1267 (2005).
[20] J. Chan, T. P. M. Alegre, A. H. Safavi-Naeini, J. T. Hill, A. Krause, S. Gröblacher, M. Aspelmeyer, and O. Painter, Nature **478**, 89 (2011).
[21] M. Aspelmeyer, P. Meystre, and K. Schwab, Phys. Today (20120701).
[22] A. Bassi, K. Lochan, S. Satin, T. P. Singh, and H. Ulbricht, Rev. Mod. Phys. **85**, 471 (2013).
[23] R. Penrose, Gen. Relativ. Gravit. **28**, 581 (1996).
[24] S. L. Adler, J. Phys. Math. Theor. **40**, 755 (2007).
[25] M. Derakhshani, C. Anastopoulos, and B. L. Hu, J. Phys. Conf. Ser. **701**, 012015 (2016).
[26] A. Rembold, G. Schütz, W. T. Chang, A. Stefanov, A. Pooch, I. S. Hwang, A. Günther, and A. Stibor, Phys. Rev. A **89**, 033635 (2014).
[27] A. Günther, A. Rembold, G. Schütz, and A. Stibor, Phys. Rev. A **92**, 053607 (2015).

APPENDIX A

MATLAB CODE FOR DECOHERENCE EXPERIMENT ANALYSIS

A.1 Matlab Code for Decoherence Experiment Analysis

The following Matlab code is used to extract horizontal and vertical lineouts of the CCD integrated images of the MCP's phosphorous screen. See the below flowchart (figure A.1) for a diagrammatical description. After The acquired image data (from a LabView image acquisition program [1]) is uploaded in its .txt format and ordering the accumulated data values into their original pixel/matrix positions. A contour plot of the histogram is generated (figure 4.11 bottom right). At this point, the pixel locations of the maxima corresponding to the individual diffraction peaks are determined. These maxima are best-fitted to a line, which determines the slope of what will be the "tilted" horizontal lineouts. Next, the zeroth order diffraction peak is selected as representative of the vertical distribution of the diffraction pattern. An integrated sum (or line-out) of the entire diffraction peak is produced (creating for example the vertical experimental data points in figure 4.11 bottom left).

From such a vertical lineout (be electron beam near or far away from the surface) a range in the y-direction is selected to perform horizontal lineouts on; namely the domain corresponding to the vertical position along the center of the zeroth order peak which is at minimum 5% of the maximum intensity value of the vertical lineout. Then an integrated tilted horizontal lineout (interpolated from the data with a vertical width of 4.8 μm) is performed, one corresponding to each data point of the vertical lineout. Each horizontal lineout is then fitted to the function of Equation 4.1, with α fixed for all

lineouts determined by the lineout corresponding to the maximum intensity. To begin a fit, an initial guess (manual or otherwise) is made for all parameters except for α. Then after undergoing an r-square fitting routine to determine the best-fit parameters, some of these new parameters are plugged in as new initial guesses (d, x_1, c_1 and c_2) which affect the position of the peaks, the periodicity and their widths. However all other parameters ($\alpha, x_0, x_2, c_3, a_1$ and A_{bck}) are held fixed. This final fit is then saved and the final parameters are then inserted as the next initial guess for the horizontal lineout fit. Finally. After recording the final parameters for all fits, the transverse coherence length is calculated using equation 2.29.

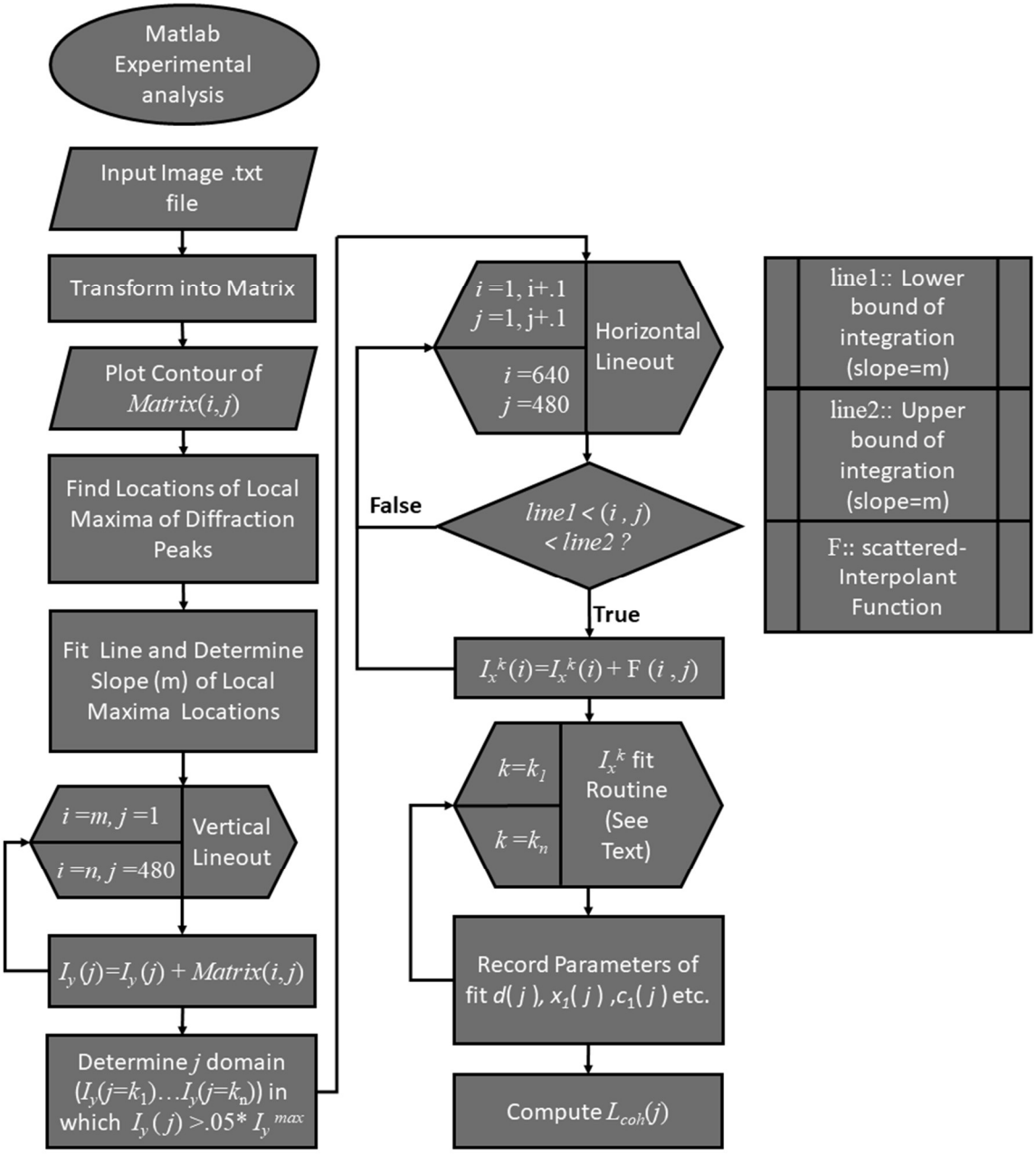

Figure A.1: Flowchart of Experimental Decoherence Image Analysis

```
xslope=[266.5,292.75,320.5];
yslope=[287,286,282.5];

cftool(xslope,yslope)
%%
clc;

i004_nearsurf_Au=zeros(480,640);
for i=1:640
    for j=1:480
        gogo=(j-1)*640+i;
        i004_nearsurf_Au(j,i)=intensity(gogo);
    end
end
%%
contour(i004_nearsurf_Au)
%%
slope=-.084;
%%
i004_nearsurf_Au(173:174,53)=0;
%intmatrix(171:172,52)=0;
%intmatrix(131,216)=0;
%intmatrix(134,216)=0;
contour(log(i004_nearsurf_Au))
%%

%%
%Nice_Contour_Generator

Xscale = [1:640]*((72e-6)/27.1913)-7.73325e-4;
Yscale = 1.2e-6*[1:480]-3.45e-4;
centery=((1.2e-6*(317.5))-3.81e-4)-((1.2e-6*(240))-3.81e-4)
[mxx,myy] = meshgrid(Yscale,Xscale);
contour(myy,mxx,transpose((i004_nearsurf_Au)),1000)
%shading interp;
box on;
%view([0,90])
hold on
set(gca,'xtick',[])
set(gca,'ytick',[])

ylim([1.2e-6*(246)-3.45e-4 1.2e-6*(320)-3.45e-4])
%ylim([(1.2e-6)-3.81e-4 ((1.2e-6)*480)-3.81e-4]+centery)
xlim([-200e-6 200e-6])
hold off


%%
ylineout_004nearsurf_Au=zeros(1,480);
for i=1:480
    for j=277:308
ylineout_004nearsurf_Au(i)=ylineout_004nearsurf_Au(i)+i004_nearsurf_Au(i,j);
    end
end
%%

clc;
figure
plot(1.2e-6*[1:480]-3.45e-4,ylineout_004nearsurf_Au./max(ylineout_004nearsurf_Au),
'.k','markers',15) %246-320
hold on
plot(simulation_domain+6e-5,revised_nearsurf_simulation_Gold,'LineWidth',3)
xlim([1.2e-6*(246)-3.45e-4 1.2e-6*(320)-3.45e-4])
xlabel('Y (in microns)','FontSize',40)
ylabel('#e^{-}','FontSize',40)
```

```
title('Vertical Electron Lineout','FontSize',40)
legend('fit','experimental data')
%
hold off
%%
%creating lineouts in x direction

newdomain=zeros(640*480,2);

newdomain(:,1)=VarName1;
newdomain(:,2)=VarName2;
F = scatteredInterpolant(newdomain,intensity,'natural','nearest')

%%

clc;
xxdom=1:.1:640;
yydom=1:.1:480;
%red lines

for k=246
line1=slope.*(xxdom-xslope(2))+k-2;
line2=slope.*(xxdom-xslope(2))+k+2;

lineout_temp=zeros(1,numel(xxdom));
for i=1:numel(xxdom)
   for j=1:numel(yydom)
       if yydom(j)>=line1(i)
           if yydom(j)<=line2(i)
               lineout_temp(i)=lineout_temp(i)+F(yydom(j),xxdom(i));
           end
       end
   end
end
lineout_im004_Au(k,:)=lineout_temp;


end
beep on
beep

%%
clc;
100*delta/(1.9*(2*sqrt(2*log(2))))
3.1*(2*sqrt(2*log(2)))
%%
clc;
k=246;
%guessing section
%plot(lineout(345,:)./max(lineout(345,:)))
xdata=1:.1:640;
xdata2=1.00003:.10003:.10018*6391;

enamp= 0.95%.99%0.9509;
alpha=0.046%.055%0.0532;
v=289.4%339.1864;
c1=291.2%336.7331;
delta=27.30%25.7624;
%delta=27.4
sigma=2.1%1.9589;
Scl=4 %4.4265 %Scaling factor determined on 11/30/16 from "beam only" fit
a1=0.9%.9861; %Amplitude factor determined on 11/30/16 from "beam only" fit
bckamp=0.05;%0.0113;
bckwidth=0.0005;%0.0005;
bckmid=290.;%341.2022%350.0158;
```

```
130
131
132   %x=[enampout_007_1(k),alphaout_007_1(k),vout_007_1(k),clout_007_1(k),dout_007_1(k)
      ,wout_007_1(k)/(2*sqrt(2*log(2))),bckamplout_007_1(k),bckwidthlout_007_1(k),bckmid
      lout_007_1(k),a1_1(k)]
133   x=[enamp,alpha,v,cl,delta,sigma,bckamp,bckwidth,bckmid,a1];
134   guess=x(1)*((sin(x(2)*(xdata2-x(3)))./(x(2)*(xdata2-x(3)))).^2).*...
135          (
             x(10)*exp(-((xdata2-x(4)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-x(4)).^2)
             /(2*Scl*x(6)^2))...
136
              +x(10)*exp(-((xdata2-x(4)+x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-x(
              4)+x(5)).^2)/(2*Scl*x(6)^2))...
137
              +x(10)*exp(-((xdata2-x(4)-x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-x(
              4)-x(5)).^2)/(2*Scl*x(6)^2))...
138
              +x(10)*exp(-((xdata2-x(4)+2*x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-
              x(4)+2*x(5)).^2)/(2*Scl*x(6)^2))...
139
              +x(10)*exp(-((xdata2-x(4)-2*x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-
              x(4)-2*x(5)).^2)/(2*Scl*x(6)^2))...
140
              +x(10)*exp(-((xdata2-x(4)+3*x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-
              x(4)+3*x(5)).^2)/(2*Scl*x(6)^2))...
141
              +x(10)*exp(-((xdata2-x(4)-3*x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-
              x(4)-3*x(5)).^2)/(2*Scl*x(6)^2))...
142           )+x(7)*exp(-x(8)*(xdata2-x(9)).^2);
143
144
145   plot(xdata2,guess)
146   hold on
147   plot(xdata,lineout_im004_Au(k,:)./max(lineout_im004_Au(k,:)),'r')
148
149   plot(xdata,x(7)*exp(-x(8)*(xdata2-x(9)).^2),'k')
150   xlim([235 355])
151   %xlim([290 320])
152   %xlim([1 640])
153   ylim([0 1.02])
154   hold off
155
156
157
158   %%
159   clc;
160   clock
161   %%
162
163   clc;
164   begintime=clock
165   xxdom=1:.1:640;
166   yydom=1:.1:480;
167   %alpha=alphaout1(321);
168   %k=256;
169   %x=[enampout_007_1(k),alphaout_007_1(k),vout_007_1(k),clout_007_1(k),dout_007_1(k)
      ,wout_007_1(k)/(2*sqrt(2*log(2))),bckamplout_007_1(k),bckwidthlout_007_1(k),bckmid
      lout_007_1(k),a1_1(k)]
170   x=[enamp,alpha,v,cl,delta,sigma,bckamp,bckwidth,bckmid,a1];
171   for k=246:320 %maxima to ~5% of maxima
172   %  k=322-running
173   %line1=slope.*(xxdom-xslope(2))+k-2;
174   %line2=slope.*(xxdom-xslope(2))+k+2;
175
176   %lineout_temp=zeros(1,numel(xxdom));
```

```
%for i=1:numel(xxdom)
%   for j=1:numel(yydom)
%       if yydom(j)>=line1(i)
%           if yydom(j)<=line2(i)
%               lineout_temp(i)=lineout_temp(i)+F(yydom(j),xxdom(i));
%           end
%       end
%   end
%end
%lineout_im004_Au(k,:)=lineout_temp;

ydata=lineout_im004_Au(k,:)./max(lineout_im004_Au(k,:));


%x=[enamp,alpha,v,c1,delta,sigma,bckamp,bckwidth,bckmid,bckamp2,bckwidth2,bckmid2]
;

%lower and upper bounds force background to not fit peaks, or become
%negative
if k<316
lb=[-inf,-inf,-inf,-inf,-inf,-inf,0,0,0,.50];
ub=[inf,inf,inf,inf,inf,inf,inf,0.0007,inf,.95];
end
if k>=316
lb=[-inf,-inf,-inf,-inf,-inf,-inf,0,0,0,.01];
ub=[inf,inf,inf,inf,inf,inf,inf,0.0007,inf,.5];
end

%lb=[-inf,-inf,-inf,-inf,-inf,-inf,0,0,0,.50];
%ub=[inf,inf,inf,inf,inf,inf,inf,0.0007,inf,.95];

f = @(x,xdata2)x(1)*((sin(x(2)*(xdata2-x(3)))./(x(2)*(xdata2-x(3)))).^2).*...
       (
       x(10)*exp(-((xdata2-x(4)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-x(4)).^2)
       /(2*Scl*x(6)^2))...
        +x(10)*exp(-((xdata2-x(4)+x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-x(
        4)+x(5)).^2)/(2*Scl*x(6)^2))...
        +x(10)*exp(-((xdata2-x(4)-x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-x(
        4)-x(5)).^2)/(2*Scl*x(6)^2))...
        +x(10)*exp(-((xdata2-x(4)+2*x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-
        x(4)+2*x(5)).^2)/(2*Scl*x(6)^2))...
        +x(10)*exp(-((xdata2-x(4)-2*x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-
        x(4)-2*x(5)).^2)/(2*Scl*x(6)^2))...
        +x(10)*exp(-((xdata2-x(4)+3*x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-
        x(4)+3*x(5)).^2)/(2*Scl*x(6)^2))...
        +x(10)*exp(-((xdata2-x(4)-3*x(5)).^2)/(2*x(6)^2))+(1-x(10))*exp(-((xdata2-
        x(4)-3*x(5)).^2)/(2*Scl*x(6)^2))...
        )+x(7)*exp(-x(8)*(xdata2-x(9)).^2);

x = lsqcurvefit(f,x,xdata2,ydata,lb,ub);
x2=[x(4),x(5),x(6)];

f2 = @(x2,xdata2)x(1)*((sin(x(2)*(xdata2-x(3)))./(x(2)*(xdata2-x(3)))).^2).*...
       (
       x(10)*exp(-((xdata2-x2(1)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdata2-x2(1)).
       ^2)/(2*Scl*x2(3)^2))...
        +x(10)*exp(-((xdata2-x2(1)+x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdata2
        -x2(1)+x2(2)).^2)/(2*Scl*x2(3)^2))...
```

```
223
          +x(10)*exp(-((xdata2-x2(1)-x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdata2
          -x2(1)-x2(2)).^2)/(2*Scl*x2(3)^2))...
224
          +x(10)*exp(-((xdata2-x2(1)+2*x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdat
          a2-x2(1)+2*x2(2)).^2)/(2*Scl*x2(3)^2))...
225
          +x(10)*exp(-((xdata2-x2(1)-2*x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdat
          a2-x2(1)-2*x2(2)).^2)/(2*Scl*x2(3)^2))...
226
          +x(10)*exp(-((xdata2-x2(1)+3*x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdat
          a2-x2(1)+3*x2(2)).^2)/(2*Scl*x2(3)^2))...
227
          +x(10)*exp(-((xdata2-x2(1)-3*x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdat
          a2-x2(1)-3*x2(2)).^2)/(2*Scl*x2(3)^2))...
228       )+x(7)*exp(-x(8)*(xdata2-x(9)).^2);
229
230   x2 = lsqcurvefit(f2,x2,xdata2,ydata);
231
232   result_004_Au_02(k,:)=x(1)*((sin(x(2)*(xdata2-x(3)))./(x(2)*(xdata2-x(3)))).^2).*.
      ..
233          (
          x(10)*exp(-((xdata2-x2(1)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdata2-x2(1)).
          ^2)/(2*Scl*x2(3)^2))...
234
          +x(10)*exp(-((xdata2-x2(1)+x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdata2
          -x2(1)+x2(2)).^2)/(2*Scl*x2(3)^2))...
235
          +x(10)*exp(-((xdata2-x2(1)-x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdata2
          -x2(1)-x2(2)).^2)/(2*Scl*x2(3)^2))...
236
          +x(10)*exp(-((xdata2-x2(1)+2*x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdat
          a2-x2(1)+2*x2(2)).^2)/(2*Scl*x2(3)^2))...
237
          +x(10)*exp(-((xdata2-x2(1)-2*x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdat
          a2-x2(1)-2*x2(2)).^2)/(2*Scl*x2(3)^2))...
238
          +x(10)*exp(-((xdata2-x2(1)+3*x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdat
          a2-x2(1)+3*x2(2)).^2)/(2*Scl*x2(3)^2))...
239
          +x(10)*exp(-((xdata2-x2(1)-3*x2(2)).^2)/(2*x2(3)^2))+(1-x(10))*exp(-((xdat
          a2-x2(1)-3*x2(2)).^2)/(2*Scl*x2(3)^2))...
240       )+x(7)*exp(-x(8)*(xdata2-x(9)).^2);
241
242   %chisqr calculation
243   %chisqr=0;
244   %norm_result=result_14(k,:)./sum(result_14(k,:));
245   %norm_ydata=ydata./sum(ydata);
246
247   %for i=1:numel(result_14(k,:))
248   %    chisqr=chisqr+((result(k,i)-ydata(i))^2)/result(k,i);
249   %     chisqr=chisqr+((norm_result(i)-norm_ydata(i))^2)/norm_result(i);
250   %end
251
252   %nu=numel(ydata)-(8); %9=free parameters/covariates
253   %reduced_chisqr(k)=chisqr/nu;
254
255   %reset x (initial guess for each k)
256   x(4)=x2(1);
257   x(5)=x2(2);
258   x(6)=x2(3);
259
260   dout_004_Au_2(k)=x(5);
261   wout_004_Au_2(k)=2*sqrt(2*log(2))*x(6);
262
```

```
263   enampout_004_Au_2(k)=x(1);
264   alphaout_004_Au_2(k)=x(2);
265   vout_004_Au_2(k)=x(3);
266   clout_004_Au_2(k)=x(4);
267   bckamplout_004_Au_2(k)=x(7);
268   bckwidthlout_004_Au_2(k)=x(8);
269   bckmidlout_004_Au_2(k)=x(9);
270   a1_004_Au_2(k)=x(10);
271
272   k
273   end
274   beep on
275   beep
276   endtime=clock
277   x(2)
278
279   alpha
280   alpha=x(2)
281
282
283   %%
284   begintime-endtime
285   %%
286
287   %dout_007_1(256:334)*100./wout_007_1(256:334);
288   dout_004_Au_2(256)
289   wout_004_Au_2(256)
290   %%
291   (8.2152-5.2795)/5.2795
292   (25.7624-22.9957)/25.7624
293   %%
294   x(6)
295   wout(362)
296   %%
297   %figure %275-289
298   k=286; %246:320
299
300   x_lineout_data_004=result_004_Au_02(k,:);
301   xfit_004=lineout_im004_Au(k,:)./max(lineout_im004_Au(k,:));
302
303   plot(xdata*((72e-6)/27.1913)-7.73325e-4,x_lineout_data_004,'LineWidth',3)
304   hold on
305   plot(xdata*((72e-6)/27.1913)-7.73325e-4,xfit_004,'.k','markers',15)
306   xlim([-200e-6 200e-6])
307   ylim([0 1.02])
308   xlabel('Y (in microns)','FontSize',40)
309   ylabel('#e^{-}','FontSize',40)
310   title('Horizontal Electron Lineout','FontSize',40)
311   legend('fit','experimental data')
312   hold off
313
314   %%
315   %275-289
316   %290-308
317
318
319   plot(1.2e-6*[246:320]-3.45e-4,lcoh_im_004_Au_2,'.-k','markers',15)%564.125
320   hold on
321   plot(simulation_domain+6e-5,real_coherence_length_zurek_gold,'g','LineWidth',3)
322   plot(simulation_domain+6e-5,real_coherence_length_mach_gold,'LineWidth',3)
323   plot(simulation_domain+6e-5,real_coherence_length_scheel_gold,'r','LineWidth',3)
324   plot(simulation_domain+6e-5,real_coherence_length_mach_gold_dblcheck,'m','LineWidt
      h',3)
325   plot(simulation_domain+6e-5,real_coherence_length_levinson_gold,'k','LineWidth',3)
326
```

```
327
328   xlim([1.2e-6*(246)-3.45e-4 1.2e-6*(320)-3.45e-4])
329   xlabel('Y (in pixels)','FontSize',40)
330   ylabel('L_{t} (in nm)','FontSize',40)
331   title('Gold','FontSize',40)
332   legend('experiment','Zurek','Machnikowski','Scheel & Buhmann','Machnikowski
      dblcheck')
333   ylim([0 680])
334   hold off
335
336   %%
337   mean(lcoh_im_004_Au_2)
338   %%
339   plot(wout_004_Au_2(246:320),'.k')
340   %%
341   enamp= 0.7669%.99%0.9509;
342   alpha=0.048%.055%0.0532;
343   v=337.1342%339.1864;
344   c1=336.9912%336.7331;
345   delta=26.0218%25.7624;
346   %delta=27.4
347   sigma=1.9%1.9589;
348   Scl=4 %4.4265 %Scaling factor determined on 11/30/16 from "beam only" fit
349   a1=0.5698%.9861; %Amplitude factor determined on 11/30/16 from "beam only" fit
350   bckamp=0.2034;%0.0113;
351   bckwidth=0.0005;%0.0005;
352   bckmid=340.5628;%341.2022%350.0158;
353   %%
354   %plot(283:364,alphaout_006_1(283:364),'.-k')
355   %plot(283:364,bckmid1out_006_1(283:364),'.-k')
356   %plot(283:364,bckwidth1out_037_1(283:364),'.-k')
357   plot(246:320,bckamp1out_004_Au_2(246:320),'.-k')
358   plot(246:320,a1_004_Au_2(246:320),'.-k')
359   hold on
360   %xlim([333.9 334.1])
361   hold off
362
363   %%
364   clc;
365   plot(1:86,lcoh_im038_01,'.-k')
366   hold on
367   plot([1:74]+11,lcoh_im034_01,'.-b')
368   hold off
369
370   %%
371
372   plot(1.3e-6*[1:480]-4.173e-4,ylineout_i007_nearsurf_h4p435_1000_36000./max(ylineou
      t_i007_nearsurf_h4p435_1000_36000),'.-b')
373   hold on
374   plot(1.3e-6*[307:335]-4.173e-4,ylineout_i007_nearsurf_h4p435_1000_36000(307:335)./
      max(ylineout_i007_nearsurf_h4p435_1000_36000(307:335)),'.-m')
375   plot(1.3e-6*[300:345]-4.173e-4,lcoh_im038_01/500,'.-r')
376   plot(simulation_domain,1.03*edist_no_surf_no_exclusion./max(edist_no_surf_no_exclu
      sion),'k')
377   plot(simulation_domain,1.03*edist_nosurf_exludedpaths./max(edist_nosurf_exludedpat
      hs),'g')
378   xlim([-.5e-4 .5e-4])
379   ylim([0 1.1])
380   title('Crude Fit, Verticle Electron Distribution of 0th order')
381   xlabel('Position (in meters)')
382   ylabel('Intensity (normalized to max=1)')
383   legend('Experiment','Classical Simulation')
384   hold off
385
386   %%
```

```
387
388   plot(1.3e-6*[307:335]-4.18e-4,ylineout_i007_nearsurf_h4p435_1000_36000(307:335)./m
      ax(ylineout_i007_nearsurf_h4p435_1000_36000(307:335)),'.-b')
389   hold on
390   plot(simulation_domain,1.03*edist_no_surf_no_exclusion./max(edist_no_surf_no_exclu
      sion),'k')
391   xlim([-1e-4 1e-4])
392   ylim([0 1.1])
393   title('Crude Fit, Verticle Electron Distribution of 0th order')
394   xlabel('Position (in meters)')
395   ylabel('Intensity (normalized to max=1)')
396   legend('Experiment','Classical Simulation')
397   hold off
398
399   %%
400
401   mean(dout_004_Au_2(246:320))
402
403
404
```

A.2 Visualization of Loss of Coherence

In order to highlight the loss of contrast in the diffraction pattern, the accumulated image of the MCP detector that was taken by the CCD camera was transformed into the revised images shown in Figure 4.3 and Figure 4.4. Figure A.2 shows the images before and after this process. Line-outs of the image are extracted to obtain diffraction patterns. The line-outs are taken at a slant with the x-direction to compensate for image skew. This skew can be explained by small rotational misalignments between the optical elements in the system, however this does not affect the measured coherence length. In the y-direction a 4.8 μm range on the detector is integrated for each line-out. Each of these line-outs then correspond to an individual horizontal line on the diffractogram.

After the individual line-outs are fitted according to equation 4.2, the background term is subtracted from the line-out to show only the relative broadening. Each diffraction peak is normalized by its maximum intensity value for that order.

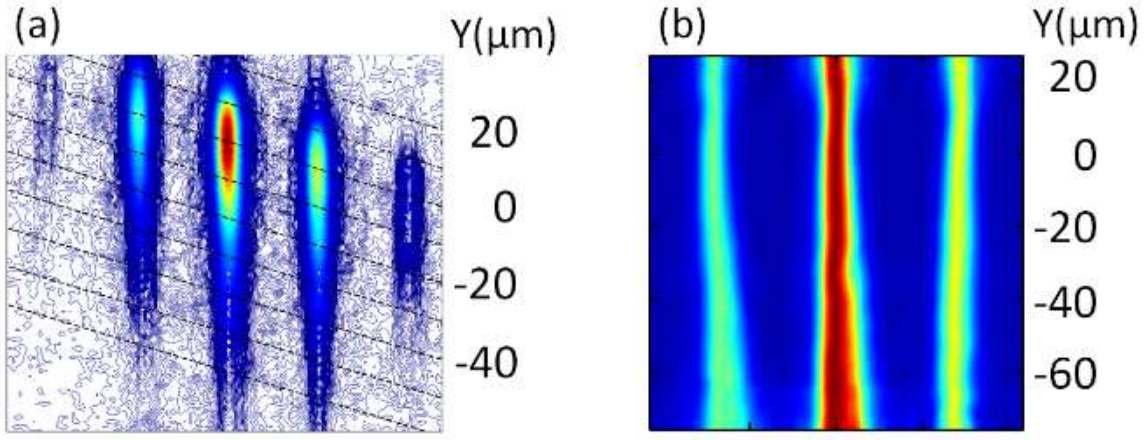

Figure A.2. Visualization of the Loss of Contrast. (a) Contour of data accumulated by CCD camera. (b) Resulting diffractogram based on data.

[1] R. Bach, Electron Matter Interferometry and the Electron Double-Slit Experiment, Appendix B, Dissertation, University of Nebraska-Lincoln, 2014.

APPENDIX B

MATLAB CODE FOR CORERENCE LENGTH CALCULATION I

The following Program is the Program for Computing the transverse coherence length of the electron beam after decoherence has occurred based on the final density matrix of the electron state. This executed by 1) deconvoluting the final density as a sum of pure quantum states, 2) propagating each state to the far field using Fourier transformation, and 3) incoherently summing the resulting probability distributions from each of the pure quantum states. See Appendix C.2 for an extended description of how this is done.

```
clc;
w=2.2745d0*sqrt(2d0);
sigma=w/(2d0*sqrt(2d0*log(2d0)));

b=1/(2d0*(sigma^2));
rho_1=zeros(2001,2001);
near_field_x=zeros(1,2001);
rho_1p5=zeros(2001,2001);
coherence_dist=zeros(1,2001);
initial_coh_dist=zeros(1,2001);
probability=zeros(1,2001);


for i=1:2001
 for j=1:2001
  x=i*10d0/2000d0;
  y=j*10d0/2000d0;
  rho_1(i,j)=exp(-b*((x-5.005)^2+(y-5.005)^2));
 end
end

for i=1:2001
 near_field_x(i)=-5d0+(i-1)*.005d0;
end

%fwhm1p67=16.48; .4628
%fwhm1p67=189.1; %Attempt to obtain density width of 557nm (525nm)
%fwhm1p67=210.2; %Attempt to obtain density of 512nm (lcoh=570nm),new si
fwhm1p67=215.44; %Attempt to obtain density of 524.1nm (lcoh=583.1229nm), old si
%fwhm1p67=220.5; %Attempt to obtain density of 535.7nm (lcoh=595.5127m), gold


beta=fwhm1p67/(2d0*sqrt(log(2d0)));

for i=1:2001
  for j=1:2001
   rho_1p5(i,j)=rho_1(i,j)*exp(-((i-j)/beta)^2);
  end
end


for i=1:2001
 j=2002-i;
 coherence_dist(i)=rho_1p5(i,j);
 initial_coh_dist(i)=rho_1(i,j);
 probability(i)=rho_1(i,i);
end


plot(near_field_x,coherence_dist,'.-k')
hold on
plot(near_field_x,initial_coh_dist,'.-b')
plot(near_field_x,probability,'.-m')
hold off


f = fit(transpose(near_field_x),transpose(coherence_dist),'gauss1')

clc;
coeffvals = coeffvalues(f);
width=2*sqrt(log(2))*coeffvals(3)


%%
%operator which acts as a grating to the wavefunction distribution
```

```
grating_operator=zeros(1,numel(near_field_x));

for i=1:numel(near_field_x)
        if mod(near_field_x(i)*100,20)>= 0
            if mod(near_field_x(i)*100,20)< 10
                grating_operator(i)=1;
            end
        end
end

%%
%send coherent_function_through_grating
clc;
%first normalize rho_1
rho_1=rho_1/trace(rho_1);
for i=1:2001
 wavefunction_1(i)=sqrt(rho_1(i,i))*grating_operator(i);
end
plot(near_field_x,wavefunction_1)


%%
clc;
wavefunction_1extended(1:1000)=0;
wavefunction_1extended(1001:3001)=wavefunction_1;
wavefunction_1extended(3002:4001)=0;
plot(wavefunction_1extended)
%%

farfield_x=[-2000:2000]*.72; %far field domain to obtain a ff peak_periodicity
of 72um

Y=fft(wavefunction_1extended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result_coherent=abs(farfield_pattern)/max(abs(farfield_pattern));
%%
plot(farfield_x,fourier_result_coherent,'.-k')
hold on
%xlim([68 76])
hold off
%%
clc;
% 0th order peak
format long
f =
fit(transpose(farfield_x(1960:2040)),transpose(fourier_result_coherent(1960:2040))
,'gauss1')
coeffvals = coeffvalues(f);
width=2*sqrt(log(2))*coeffvals(3);
.1*72/width % = 2.573496999326959
%%
clc;
% 1st order peak
format long
f =
fit(transpose(farfield_x(2060:2140)),transpose(fourier_result_coherent(2060:2140))
,'gauss1')
coeffvals = coeffvalues(f);
width=2*sqrt(log(2))*coeffvals(3);
.1*72/width % = 2.576545361683533

%%
clc;
2.2745/2.573496999326959%ratio of diagonal width to coherence length
```

```
126   2.2745/2.576545361683533
127   %%
128   clc;
129   1/(2*3.14159265)
130   %%
131   clc;
132
133   72*.1/(384.7587-367.3198)
134
```

```
1
2   plot(near_field_x,coherence_dist,'.-k')
3   hold on
4   plot(near_field_x,initial_coh_dist,'.-b')
5   plot(near_field_x,probability,'.-m')
6   hold off
7   %%
8
9   clc;
10
11  for i=1:2001
12   near_field_x(i)=-5d0+(i-1)*.005d0;
13  end
14
15  w=2.2745d0;
16  %wdblprime=.5;% produces lcoh=.557;
17  %wdblprime=.5120; %produces lcoh = .570
18  wdblprime=.5241; %produces lcoh = 583.1229, old si
19  %wdblprime=.5357; %produces lcoh = 595.5127, gold
20  sigmadbl=wdblprime/(2d0*sqrt(2d0*log(2d0)));
21  delx=.1;
22
23  wprime=sqrt((w*w)-(wdblprime*wdblprime));
24  sigmaprime=wprime/(2d0*sqrt(2d0*log(2d0)));
25
26
27  a0=exp(-(0).^2/(2*sigmaprime*sigmaprime));
28  a1=exp(-(delx).^2/(2*sigmaprime*sigmaprime));
29  a2=exp(-(2*delx).^2/(2*sigmaprime*sigmaprime));
30  a3=exp(-(3*delx).^2/(2*sigmaprime*sigmaprime));
31  a4=exp(-(4*delx).^2/(2*sigmaprime*sigmaprime));
32  a5=exp(-(5*delx).^2/(2*sigmaprime*sigmaprime));
33  a6=exp(-(6*delx).^2/(2*sigmaprime*sigmaprime));
34  a7=exp(-(7*delx).^2/(2*sigmaprime*sigmaprime));
35  a8=exp(-(8*delx).^2/(2*sigmaprime*sigmaprime));
36  a9=exp(-(9*delx).^2/(2*sigmaprime*sigmaprime));
37  a10=exp(-(10*delx).^2/(2*sigmaprime*sigmaprime));
38  a11=exp(-(11*delx).^2/(2*sigmaprime*sigmaprime));
39  a12=exp(-(12*delx).^2/(2*sigmaprime*sigmaprime));
40  a13=exp(-(13*delx).^2/(2*sigmaprime*sigmaprime));
41  a14=exp(-(14*delx).^2/(2*sigmaprime*sigmaprime));
42  a15=exp(-(15*delx).^2/(2*sigmaprime*sigmaprime));
43  a16=exp(-(16*delx).^2/(2*sigmaprime*sigmaprime));
44  a17=exp(-(17*delx).^2/(2*sigmaprime*sigmaprime));
45  a18=exp(-(18*delx).^2/(2*sigmaprime*sigmaprime));
46  a19=exp(-(19*delx).^2/(2*sigmaprime*sigmaprime));
47  a20=exp(-(20*delx).^2/(2*sigmaprime*sigmaprime));
48  a21=exp(-(21*delx).^2/(2*sigmaprime*sigmaprime));
49  a22=exp(-(22*delx).^2/(2*sigmaprime*sigmaprime));
50  a23=exp(-(23*delx).^2/(2*sigmaprime*sigmaprime));
51  a24=exp(-(24*delx).^2/(2*sigmaprime*sigmaprime));
52  a25=exp(-(25*delx).^2/(2*sigmaprime*sigmaprime));
53  a26=exp(-(26*delx).^2/(2*sigmaprime*sigmaprime));
54  a27=exp(-(27*delx).^2/(2*sigmaprime*sigmaprime));
55  a28=exp(-(28*delx).^2/(2*sigmaprime*sigmaprime));
56  a29=exp(-(29*delx).^2/(2*sigmaprime*sigmaprime));
57  a30=exp(-(30*delx).^2/(2*sigmaprime*sigmaprime));
58
59  rho_lp5_prob0=a0*exp(-(near_field_x).^2/(2*sigmadbl*sigmadbl));
60  rho_lp5_prob1a=a1*exp(-(near_field_x-delx).^2/(2*sigmadbl*sigmadbl));
61  rho_lp5_prob1b=a1*exp(-(near_field_x+delx).^2/(2*sigmadbl*sigmadbl));
62  rho_lp5_prob2a=a2*exp(-(near_field_x-2*delx).^2/(2*sigmadbl*sigmadbl));
63  rho_lp5_prob2b=a2*exp(-(near_field_x+2*delx).^2/(2*sigmadbl*sigmadbl));
64  rho_lp5_prob3a=a3*exp(-(near_field_x-3*delx).^2/(2*sigmadbl*sigmadbl));
65  rho_lp5_prob3b=a3*exp(-(near_field_x+3*delx).^2/(2*sigmadbl*sigmadbl));
```

```
rho_1p5_prob4a=a4*exp(-(near_field_x-4*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob4b=a4*exp(-(near_field_x+4*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob5a=a5*exp(-(near_field_x-5*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob5b=a5*exp(-(near_field_x+5*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob6a=a6*exp(-(near_field_x-6*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob6b=a6*exp(-(near_field_x+6*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob7a=a7*exp(-(near_field_x-7*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob7b=a7*exp(-(near_field_x+7*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob8a=a8*exp(-(near_field_x-8*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob8b=a8*exp(-(near_field_x+8*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob9a=a9*exp(-(near_field_x-9*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob9b=a9*exp(-(near_field_x+9*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob10a=a10*exp(-(near_field_x-10*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob10b=a10*exp(-(near_field_x+10*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob11a=a11*exp(-(near_field_x-11*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob11b=a11*exp(-(near_field_x+11*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob12a=a12*exp(-(near_field_x-12*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob12b=a12*exp(-(near_field_x+12*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob13a=a13*exp(-(near_field_x-13*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob13b=a13*exp(-(near_field_x+13*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob14a=a14*exp(-(near_field_x-14*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob14b=a14*exp(-(near_field_x+14*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob15a=a15*exp(-(near_field_x-15*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob15b=a15*exp(-(near_field_x+15*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob16a=a16*exp(-(near_field_x-16*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob16b=a16*exp(-(near_field_x+16*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob17a=a17*exp(-(near_field_x-17*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob17b=a17*exp(-(near_field_x+17*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob18a=a18*exp(-(near_field_x-18*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob18b=a18*exp(-(near_field_x+18*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob19a=a19*exp(-(near_field_x-19*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob19b=a19*exp(-(near_field_x+19*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob20a=a20*exp(-(near_field_x-20*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob20b=a20*exp(-(near_field_x+20*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob21a=a21*exp(-(near_field_x-21*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob21b=a21*exp(-(near_field_x+21*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob22a=a22*exp(-(near_field_x-22*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob22b=a22*exp(-(near_field_x+22*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob23a=a23*exp(-(near_field_x-23*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob23b=a23*exp(-(near_field_x+23*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob24a=a24*exp(-(near_field_x-24*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob24b=a24*exp(-(near_field_x+24*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob25a=a25*exp(-(near_field_x-25*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob25b=a25*exp(-(near_field_x+25*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob26a=a26*exp(-(near_field_x-26*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob26b=a26*exp(-(near_field_x+26*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob27a=a27*exp(-(near_field_x-27*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob27b=a27*exp(-(near_field_x+27*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob28a=a28*exp(-(near_field_x-28*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob28b=a28*exp(-(near_field_x+28*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob29a=a29*exp(-(near_field_x-29*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob29b=a29*exp(-(near_field_x+29*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob30a=a30*exp(-(near_field_x-30*delx).^2/(2*sigmadbl*sigmadbl));
rho_1p5_prob30b=a30*exp(-(near_field_x+30*delx).^2/(2*sigmadbl*sigmadbl));

rho_1p5_total=rho_1p5_prob0+rho_1p5_prob1a+rho_1p5_prob1b+rho_1p5_prob2a+rho_1p5_p
rob2b...

                          +rho_1p5_prob3a+rho_1p5_prob3b+rho_1p5_prob4a+rho_1p5_p
                          rob4b...

                          +rho_1p5_prob5a+rho_1p5_prob5b+rho_1p5_prob6a+rho_1p5_p
                          rob6b...

                          +rho_1p5_prob7a+rho_1p5_prob7b+rho_1p5_prob8a+rho_1p5_p
```

```
                            rob8b...
125
                            +rho_1p5_prob9a+rho_1p5_prob9b+rho_1p5_prob10a+rho_1p5_
                            prob10b...
126
                            +rho_1p5_prob11a+rho_1p5_prob11b+rho_1p5_prob12a+rho_1p
                            5_prob12b...
127
                            +rho_1p5_prob13a+rho_1p5_prob13b+rho_1p5_prob14a+rho_1p
                            5_prob14b...
128
                            +rho_1p5_prob15a+rho_1p5_prob15b+rho_1p5_prob16a+rho_1p
                            5_prob16b...
129
                            +rho_1p5_prob17a+rho_1p5_prob17b+rho_1p5_prob18a+rho_1p
                            5_prob18b...
130
                            +rho_1p5_prob19a+rho_1p5_prob19b+rho_1p5_prob20a+rho_1p
                            5_prob20b...
131
                            +rho_1p5_prob21a+rho_1p5_prob21b+rho_1p5_prob22a+rho_1p
                            5_prob22b...
132
                            +rho_1p5_prob23a+rho_1p5_prob23b+rho_1p5_prob24a+rho_1p
                            5_prob24b...
133
                            +rho_1p5_prob25a+rho_1p5_prob25b+rho_1p5_prob26a+rho_1p
                            5_prob26b...
134
                            +rho_1p5_prob27a+rho_1p5_prob27b+rho_1p5_prob28a+rho_1p
                            5_prob28b...
135
                            +rho_1p5_prob29a+rho_1p5_prob29b+rho_1p5_prob30a+rho_1p
                            5_prob30b;
136
137   plot(near_field_x,probability,'m')
138   hold on
139   plot(near_field_x,rho_1p5_prob2a,'g')
140   plot(near_field_x,rho_1p5_prob15a,'g')
141   %plot(near_field_x,rho_1p5_prob1,'b')
142   plot(near_field_x,rho_1p5_total./max(rho_1p5_total),'k')
143   hold off
144
145   %%
146
147   for i=1:2001
148    wavefun_1p5_0(i)=sqrt(rho_1p5_prob0(i))*grating_operator(i);
149    wavefun_1p5_1a(i)=sqrt(rho_1p5_prob1a(i))*grating_operator(i);
150    wavefun_1p5_1b(i)=sqrt(rho_1p5_prob1b(i))*grating_operator(i);
151    wavefun_1p5_2a(i)=sqrt(rho_1p5_prob2a(i))*grating_operator(i);
152    wavefun_1p5_2b(i)=sqrt(rho_1p5_prob2b(i))*grating_operator(i);
153    wavefun_1p5_3a(i)=sqrt(rho_1p5_prob3a(i))*grating_operator(i);
154    wavefun_1p5_3b(i)=sqrt(rho_1p5_prob3b(i))*grating_operator(i);
155    wavefun_1p5_4a(i)=sqrt(rho_1p5_prob4a(i))*grating_operator(i);
156    wavefun_1p5_4b(i)=sqrt(rho_1p5_prob4b(i))*grating_operator(i);
157    wavefun_1p5_5a(i)=sqrt(rho_1p5_prob5a(i))*grating_operator(i);
158    wavefun_1p5_5b(i)=sqrt(rho_1p5_prob5b(i))*grating_operator(i);
159    wavefun_1p5_6a(i)=sqrt(rho_1p5_prob6a(i))*grating_operator(i);
160    wavefun_1p5_6b(i)=sqrt(rho_1p5_prob6b(i))*grating_operator(i);
161    wavefun_1p5_7a(i)=sqrt(rho_1p5_prob7a(i))*grating_operator(i);
162    wavefun_1p5_7b(i)=sqrt(rho_1p5_prob7b(i))*grating_operator(i);
163    wavefun_1p5_8a(i)=sqrt(rho_1p5_prob8a(i))*grating_operator(i);
164    wavefun_1p5_8b(i)=sqrt(rho_1p5_prob8b(i))*grating_operator(i);
165    wavefun_1p5_9a(i)=sqrt(rho_1p5_prob9a(i))*grating_operator(i);
166    wavefun_1p5_9b(i)=sqrt(rho_1p5_prob9b(i))*grating_operator(i);
```

```
    wavefun_1p5_10a(i)=sqrt(rho_1p5_prob10a(i))*grating_operator(i);
    wavefun_1p5_10b(i)=sqrt(rho_1p5_prob10b(i))*grating_operator(i);
    wavefun_1p5_11a(i)=sqrt(rho_1p5_prob11a(i))*grating_operator(i);
    wavefun_1p5_11b(i)=sqrt(rho_1p5_prob11b(i))*grating_operator(i);
    wavefun_1p5_12a(i)=sqrt(rho_1p5_prob12a(i))*grating_operator(i);
    wavefun_1p5_12b(i)=sqrt(rho_1p5_prob12b(i))*grating_operator(i);
    wavefun_1p5_13a(i)=sqrt(rho_1p5_prob13a(i))*grating_operator(i);
    wavefun_1p5_13b(i)=sqrt(rho_1p5_prob13b(i))*grating_operator(i);
    wavefun_1p5_14a(i)=sqrt(rho_1p5_prob14a(i))*grating_operator(i);
    wavefun_1p5_14b(i)=sqrt(rho_1p5_prob14b(i))*grating_operator(i);
    wavefun_1p5_15a(i)=sqrt(rho_1p5_prob15a(i))*grating_operator(i);
    wavefun_1p5_15b(i)=sqrt(rho_1p5_prob15b(i))*grating_operator(i);
    wavefun_1p5_16a(i)=sqrt(rho_1p5_prob16a(i))*grating_operator(i);
    wavefun_1p5_16b(i)=sqrt(rho_1p5_prob16b(i))*grating_operator(i);
    wavefun_1p5_17a(i)=sqrt(rho_1p5_prob17a(i))*grating_operator(i);
    wavefun_1p5_17b(i)=sqrt(rho_1p5_prob17b(i))*grating_operator(i);
    wavefun_1p5_18a(i)=sqrt(rho_1p5_prob18a(i))*grating_operator(i);
    wavefun_1p5_18b(i)=sqrt(rho_1p5_prob18b(i))*grating_operator(i);
    wavefun_1p5_19a(i)=sqrt(rho_1p5_prob19a(i))*grating_operator(i);
    wavefun_1p5_19b(i)=sqrt(rho_1p5_prob19b(i))*grating_operator(i);
    wavefun_1p5_20a(i)=sqrt(rho_1p5_prob20a(i))*grating_operator(i);
    wavefun_1p5_20b(i)=sqrt(rho_1p5_prob20b(i))*grating_operator(i);
    wavefun_1p5_21a(i)=sqrt(rho_1p5_prob21a(i))*grating_operator(i);
    wavefun_1p5_21b(i)=sqrt(rho_1p5_prob21b(i))*grating_operator(i);
    wavefun_1p5_22a(i)=sqrt(rho_1p5_prob22a(i))*grating_operator(i);
    wavefun_1p5_22b(i)=sqrt(rho_1p5_prob22b(i))*grating_operator(i);
    wavefun_1p5_23a(i)=sqrt(rho_1p5_prob23a(i))*grating_operator(i);
    wavefun_1p5_23b(i)=sqrt(rho_1p5_prob23b(i))*grating_operator(i);
    wavefun_1p5_24a(i)=sqrt(rho_1p5_prob24a(i))*grating_operator(i);
    wavefun_1p5_24b(i)=sqrt(rho_1p5_prob24b(i))*grating_operator(i);
    wavefun_1p5_25a(i)=sqrt(rho_1p5_prob25a(i))*grating_operator(i);
    wavefun_1p5_25b(i)=sqrt(rho_1p5_prob25b(i))*grating_operator(i);
    wavefun_1p5_26a(i)=sqrt(rho_1p5_prob26a(i))*grating_operator(i);
    wavefun_1p5_26b(i)=sqrt(rho_1p5_prob26b(i))*grating_operator(i);
    wavefun_1p5_27a(i)=sqrt(rho_1p5_prob27a(i))*grating_operator(i);
    wavefun_1p5_27b(i)=sqrt(rho_1p5_prob27b(i))*grating_operator(i);
    wavefun_1p5_28a(i)=sqrt(rho_1p5_prob28a(i))*grating_operator(i);
    wavefun_1p5_28b(i)=sqrt(rho_1p5_prob28b(i))*grating_operator(i);
    wavefun_1p5_29a(i)=sqrt(rho_1p5_prob29a(i))*grating_operator(i);
    wavefun_1p5_29b(i)=sqrt(rho_1p5_prob29b(i))*grating_operator(i);
    wavefun_1p5_30a(i)=sqrt(rho_1p5_prob30a(i))*grating_operator(i);
    wavefun_1p5_30b(i)=sqrt(rho_1p5_prob30b(i))*grating_operator(i);

end
%%
wavefun_1p5_0extended(1:1000)=0;
wavefun_1p5_0extended(1001:3001)=wavefun_1p5_0;
wavefun_1p5_0extended(3002:4001)=0;

wavefun_1p5_1aextended(1:1000)=0;
wavefun_1p5_1aextended(1001:3001)=wavefun_1p5_1a;
wavefun_1p5_1aextended(3002:4001)=0;
wavefun_1p5_1bextended(1:1000)=0;
wavefun_1p5_1bextended(1001:3001)=wavefun_1p5_1b;
wavefun_1p5_1bextended(3002:4001)=0;
wavefun_1p5_2aextended(1:1000)=0;
wavefun_1p5_2aextended(1001:3001)=wavefun_1p5_2a;
wavefun_1p5_2aextended(3002:4001)=0;
wavefun_1p5_2bextended(1:1000)=0;
wavefun_1p5_2bextended(1001:3001)=wavefun_1p5_2b;
wavefun_1p5_2bextended(3002:4001)=0;
wavefun_1p5_3aextended(1:1000)=0;
wavefun_1p5_3aextended(1001:3001)=wavefun_1p5_3a;
wavefun_1p5_3aextended(3002:4001)=0;
wavefun_1p5_3bextended(1:1000)=0;
```

```
wavefun_1p5_3bextended(1001:3001)=wavefun_1p5_3b;
wavefun_1p5_3bextended(3002:4001)=0;
wavefun_1p5_4aextended(1:1000)=0;
wavefun_1p5_4aextended(1001:3001)=wavefun_1p5_4a;
wavefun_1p5_4aextended(3002:4001)=0;
wavefun_1p5_4bextended(1:1000)=0;
wavefun_1p5_4bextended(1001:3001)=wavefun_1p5_4b;
wavefun_1p5_4bextended(3002:4001)=0;
wavefun_1p5_5aextended(1:1000)=0;
wavefun_1p5_5aextended(1001:3001)=wavefun_1p5_5a;
wavefun_1p5_5aextended(3002:4001)=0;
wavefun_1p5_5bextended(1:1000)=0;
wavefun_1p5_5bextended(1001:3001)=wavefun_1p5_5b;
wavefun_1p5_5bextended(3002:4001)=0;

wavefun_1p5_6aextended(1:1000)=0;
wavefun_1p5_6aextended(1001:3001)=wavefun_1p5_6a;
wavefun_1p5_6aextended(3002:4001)=0;
wavefun_1p5_6bextended(1:1000)=0;
wavefun_1p5_6bextended(1001:3001)=wavefun_1p5_6b;
wavefun_1p5_6bextended(3002:4001)=0;
wavefun_1p5_7aextended(1:1000)=0;
wavefun_1p5_7aextended(1001:3001)=wavefun_1p5_7a;
wavefun_1p5_7aextended(3002:4001)=0;
wavefun_1p5_7bextended(1:1000)=0;
wavefun_1p5_7bextended(1001:3001)=wavefun_1p5_7b;
wavefun_1p5_7bextended(3002:4001)=0;
wavefun_1p5_8aextended(1:1000)=0;
wavefun_1p5_8aextended(1001:3001)=wavefun_1p5_8a;
wavefun_1p5_8aextended(3002:4001)=0;
wavefun_1p5_8bextended(1:1000)=0;
wavefun_1p5_8bextended(1001:3001)=wavefun_1p5_8b;
wavefun_1p5_8bextended(3002:4001)=0;
wavefun_1p5_9aextended(1:1000)=0;
wavefun_1p5_9aextended(1001:3001)=wavefun_1p5_9a;
wavefun_1p5_9aextended(3002:4001)=0;
wavefun_1p5_9bextended(1:1000)=0;
wavefun_1p5_9bextended(1001:3001)=wavefun_1p5_9b;
wavefun_1p5_9bextended(3002:4001)=0;
wavefun_1p5_10aextended(1:1000)=0;
wavefun_1p5_10aextended(1001:3001)=wavefun_1p5_10a;
wavefun_1p5_10aextended(3002:4001)=0;
wavefun_1p5_10bextended(1:1000)=0;
wavefun_1p5_10bextended(1001:3001)=wavefun_1p5_10b;
wavefun_1p5_10bextended(3002:4001)=0;

wavefun_1p5_11aextended(1:1000)=0;
wavefun_1p5_11aextended(1001:3001)=wavefun_1p5_11a;
wavefun_1p5_11aextended(3002:4001)=0;
wavefun_1p5_11bextended(1:1000)=0;
wavefun_1p5_11bextended(1001:3001)=wavefun_1p5_11b;
wavefun_1p5_11bextended(3002:4001)=0;
wavefun_1p5_12aextended(1:1000)=0;
wavefun_1p5_12aextended(1001:3001)=wavefun_1p5_12a;
wavefun_1p5_12aextended(3002:4001)=0;
wavefun_1p5_12bextended(1:1000)=0;
wavefun_1p5_12bextended(1001:3001)=wavefun_1p5_12b;
wavefun_1p5_12bextended(3002:4001)=0;
wavefun_1p5_13aextended(1:1000)=0;
wavefun_1p5_13aextended(1001:3001)=wavefun_1p5_13a;
wavefun_1p5_13aextended(3002:4001)=0;
wavefun_1p5_13bextended(1:1000)=0;
wavefun_1p5_13bextended(1001:3001)=wavefun_1p5_13b;
wavefun_1p5_13bextended(3002:4001)=0;
wavefun_1p5_14aextended(1:1000)=0;
```

```
297   wavefun_1p5_14aextended(1001:3001)=wavefun_1p5_14a;
298   wavefun_1p5_14aextended(3002:4001)=0;
299   wavefun_1p5_14bextended(1:1000)=0;
300   wavefun_1p5_14bextended(1001:3001)=wavefun_1p5_14b;
301   wavefun_1p5_14bextended(3002:4001)=0;
302   wavefun_1p5_15aextended(1:1000)=0;
303   wavefun_1p5_15aextended(1001:3001)=wavefun_1p5_15a;
304   wavefun_1p5_15aextended(3002:4001)=0;
305   wavefun_1p5_15bextended(1:1000)=0;
306   wavefun_1p5_15bextended(1001:3001)=wavefun_1p5_15b;
307   wavefun_1p5_15bextended(3002:4001)=0;
308
309   wavefun_1p5_16aextended(1:1000)=0;
310   wavefun_1p5_16aextended(1001:3001)=wavefun_1p5_16a;
311   wavefun_1p5_16aextended(3002:4001)=0;
312   wavefun_1p5_16bextended(1:1000)=0;
313   wavefun_1p5_16bextended(1001:3001)=wavefun_1p5_16b;
314   wavefun_1p5_16bextended(3002:4001)=0;
315   wavefun_1p5_17aextended(1:1000)=0;
316   wavefun_1p5_17aextended(1001:3001)=wavefun_1p5_17a;
317   wavefun_1p5_17aextended(3002:4001)=0;
318   wavefun_1p5_17bextended(1:1000)=0;
319   wavefun_1p5_17bextended(1001:3001)=wavefun_1p5_17b;
320   wavefun_1p5_17bextended(3002:4001)=0;
321   wavefun_1p5_18aextended(1:1000)=0;
322   wavefun_1p5_18aextended(1001:3001)=wavefun_1p5_18a;
323   wavefun_1p5_18aextended(3002:4001)=0;
324   wavefun_1p5_18bextended(1:1000)=0;
325   wavefun_1p5_18bextended(1001:3001)=wavefun_1p5_18b;
326   wavefun_1p5_18bextended(3002:4001)=0;
327   wavefun_1p5_19aextended(1:1000)=0;
328   wavefun_1p5_19aextended(1001:3001)=wavefun_1p5_19a;
329   wavefun_1p5_19aextended(3002:4001)=0;
330   wavefun_1p5_19bextended(1:1000)=0;
331   wavefun_1p5_19bextended(1001:3001)=wavefun_1p5_19b;
332   wavefun_1p5_19bextended(3002:4001)=0;
333   wavefun_1p5_20aextended(1:1000)=0;
334   wavefun_1p5_20aextended(1001:3001)=wavefun_1p5_20a;
335   wavefun_1p5_20aextended(3002:4001)=0;
336   wavefun_1p5_20bextended(1:1000)=0;
337   wavefun_1p5_20bextended(1001:3001)=wavefun_1p5_20b;
338   wavefun_1p5_20bextended(3002:4001)=0;
339
340   wavefun_1p5_21aextended(1:1000)=0;
341   wavefun_1p5_21aextended(1001:3001)=wavefun_1p5_21a;
342   wavefun_1p5_21aextended(3002:4001)=0;
343   wavefun_1p5_21bextended(1:1000)=0;
344   wavefun_1p5_21bextended(1001:3001)=wavefun_1p5_21b;
345   wavefun_1p5_21bextended(3002:4001)=0;
346   wavefun_1p5_22aextended(1:1000)=0;
347   wavefun_1p5_22aextended(1001:3001)=wavefun_1p5_22a;
348   wavefun_1p5_22aextended(3002:4001)=0;
349   wavefun_1p5_22bextended(1:1000)=0;
350   wavefun_1p5_22bextended(1001:3001)=wavefun_1p5_22b;
351   wavefun_1p5_22bextended(3002:4001)=0;
352   wavefun_1p5_23aextended(1:1000)=0;
353   wavefun_1p5_23aextended(1001:3001)=wavefun_1p5_23a;
354   wavefun_1p5_23aextended(3002:4001)=0;
355   wavefun_1p5_23bextended(1:1000)=0;
356   wavefun_1p5_23bextended(1001:3001)=wavefun_1p5_23b;
357   wavefun_1p5_23bextended(3002:4001)=0;
358   wavefun_1p5_24aextended(1:1000)=0;
359   wavefun_1p5_24aextended(1001:3001)=wavefun_1p5_24a;
360   wavefun_1p5_24aextended(3002:4001)=0;
361   wavefun_1p5_24bextended(1:1000)=0;
```

```
wavefun_1p5_24bextended(1001:3001)=wavefun_1p5_24b;
wavefun_1p5_24bextended(3002:4001)=0;
wavefun_1p5_25aextended(1:1000)=0;
wavefun_1p5_25aextended(1001:3001)=wavefun_1p5_25a;
wavefun_1p5_25aextended(3002:4001)=0;
wavefun_1p5_25bextended(1:1000)=0;
wavefun_1p5_25bextended(1001:3001)=wavefun_1p5_25b;
wavefun_1p5_25bextended(3002:4001)=0;

wavefun_1p5_26aextended(1:1000)=0;
wavefun_1p5_26aextended(1001:3001)=wavefun_1p5_26a;
wavefun_1p5_26aextended(3002:4001)=0;
wavefun_1p5_26bextended(1:1000)=0;
wavefun_1p5_26bextended(1001:3001)=wavefun_1p5_26b;
wavefun_1p5_26bextended(3002:4001)=0;
wavefun_1p5_27aextended(1:1000)=0;
wavefun_1p5_27aextended(1001:3001)=wavefun_1p5_27a;
wavefun_1p5_27aextended(3002:4001)=0;
wavefun_1p5_27bextended(1:1000)=0;
wavefun_1p5_27bextended(1001:3001)=wavefun_1p5_27b;
wavefun_1p5_27bextended(3002:4001)=0;
wavefun_1p5_28aextended(1:1000)=0;
wavefun_1p5_28aextended(1001:3001)=wavefun_1p5_28a;
wavefun_1p5_28aextended(3002:4001)=0;
wavefun_1p5_28bextended(1:1000)=0;
wavefun_1p5_28bextended(1001:3001)=wavefun_1p5_28b;
wavefun_1p5_28bextended(3002:4001)=0;
wavefun_1p5_29aextended(1:1000)=0;
wavefun_1p5_29aextended(1001:3001)=wavefun_1p5_29a;
wavefun_1p5_29aextended(3002:4001)=0;
wavefun_1p5_29bextended(1:1000)=0;
wavefun_1p5_29bextended(1001:3001)=wavefun_1p5_29b;
wavefun_1p5_29bextended(3002:4001)=0;
wavefun_1p5_30aextended(1:1000)=0;
wavefun_1p5_30aextended(1001:3001)=wavefun_1p5_30a;
wavefun_1p5_30aextended(3002:4001)=0;
wavefun_1p5_30bextended(1:1000)=0;
wavefun_1p5_30bextended(1001:3001)=wavefun_1p5_30b;
wavefun_1p5_30bextended(3002:4001)=0;

%%

farfield_x=[-2000:2000]*.72; %far field domain to obtain a ff peak_periodicity
of 72um

Y=fft(wavefun_1p5_0extended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_0=a0*fourier_result;

Y=fft(wavefun_1p5_1aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_1a=a1*interp1(farfield_x-1*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_1bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_1b=a1*interp1(farfield_x+1*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_2aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
```

```
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_2a=a2*interp1(farfield_x-2*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_2bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_2b=a2*interp1(farfield_x+2*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_3aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_3a=a3*interp1(farfield_x-3*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_3bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_3b=a3*interp1(farfield_x+3*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_4aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_4a=a4*interp1(farfield_x-4*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_4bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_4b=a4*interp1(farfield_x+4*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_5aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_5a=a5*interp1(farfield_x-5*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_5bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_5b=a5*interp1(farfield_x+5*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_6aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_6a=a6*interp1(farfield_x-6*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_6bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_6b=a6*interp1(farfield_x+6*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_7aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_7a=a7*interp1(farfield_x-7*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_7bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_7b=a7*interp1(farfield_x+7*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_8aextended);
farfieldintensity = real( Y .* conj( Y ) );
```

```
491  farfield_pattern=fftshift( farfieldintensity );
492  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
493  scaledresult_1p5_8a=a8*interp1(farfield_x-8*delx,fourier_result,farfield_x);
494  Y=fft(wavefun_1p5_8bextended);
495  farfieldintensity = real( Y .* conj( Y ) );
496  farfield_pattern=fftshift( farfieldintensity );
497  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
498  scaledresult_1p5_8b=a8*interp1(farfield_x+8*delx,fourier_result,farfield_x);
499
500  Y=fft(wavefun_1p5_9aextended);
501  farfieldintensity = real( Y .* conj( Y ) );
502  farfield_pattern=fftshift( farfieldintensity );
503  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
504  scaledresult_1p5_9a=a9*interp1(farfield_x-9*delx,fourier_result,farfield_x);
505  Y=fft(wavefun_1p5_9bextended);
506  farfieldintensity = real( Y .* conj( Y ) );
507  farfield_pattern=fftshift( farfieldintensity );
508  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
509  scaledresult_1p5_9b=a9*interp1(farfield_x+9*delx,fourier_result,farfield_x);
510
511  Y=fft(wavefun_1p5_10aextended);
512  farfieldintensity = real( Y .* conj( Y ) );
513  farfield_pattern=fftshift( farfieldintensity );
514  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
515  scaledresult_1p5_10a=a10*interp1(farfield_x-10*delx,fourier_result,farfield_x);
516  Y=fft(wavefun_1p5_10bextended);
517  farfieldintensity = real( Y .* conj( Y ) );
518  farfield_pattern=fftshift( farfieldintensity );
519  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
520  scaledresult_1p5_10b=a10*interp1(farfield_x+10*delx,fourier_result,farfield_x);
521
522  Y=fft(wavefun_1p5_11aextended);
523  farfieldintensity = real( Y .* conj( Y ) );
524  farfield_pattern=fftshift( farfieldintensity );
525  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
526  scaledresult_1p5_11a=a11*interp1(farfield_x-11*delx,fourier_result,farfield_x);
527  Y=fft(wavefun_1p5_11bextended);
528  farfieldintensity = real( Y .* conj( Y ) );
529  farfield_pattern=fftshift( farfieldintensity );
530  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
531  scaledresult_1p5_11b=a11*interp1(farfield_x+11*delx,fourier_result,farfield_x);
532
533  Y=fft(wavefun_1p5_12aextended);
534  farfieldintensity = real( Y .* conj( Y ) );
535  farfield_pattern=fftshift( farfieldintensity );
536  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
537  scaledresult_1p5_12a=a12*interp1(farfield_x-12*delx,fourier_result,farfield_x);
538  Y=fft(wavefun_1p5_12bextended);
539  farfieldintensity = real( Y .* conj( Y ) );
540  farfield_pattern=fftshift( farfieldintensity );
541  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
542  scaledresult_1p5_12b=a12*interp1(farfield_x+12*delx,fourier_result,farfield_x);
543
544  Y=fft(wavefun_1p5_13aextended);
545  farfieldintensity = real( Y .* conj( Y ) );
546  farfield_pattern=fftshift( farfieldintensity );
547  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
548  scaledresult_1p5_13a=a13*interp1(farfield_x-13*delx,fourier_result,farfield_x);
549  Y=fft(wavefun_1p5_13bextended);
550  farfieldintensity = real( Y .* conj( Y ) );
551  farfield_pattern=fftshift( farfieldintensity );
552  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
553  scaledresult_1p5_13b=a13*interp1(farfield_x+13*delx,fourier_result,farfield_x);
554
555  Y=fft(wavefun_1p5_14aextended);
```

```
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_14a=a14*interp1(farfield_x-14*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_14bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_14b=a14*interp1(farfield_x+14*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_15aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_15a=a15*interp1(farfield_x-15*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_15bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_15b=a15*interp1(farfield_x+15*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_16aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_16a=a16*interp1(farfield_x-16*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_16bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_16b=a16*interp1(farfield_x+16*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_17aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_17a=a17*interp1(farfield_x-17*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_17bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_17b=a17*interp1(farfield_x+17*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_18aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_18a=a18*interp1(farfield_x-18*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_18bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_18b=a18*interp1(farfield_x+18*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_19aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_19a=a19*interp1(farfield_x-19*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_19bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_19b=a19*interp1(farfield_x+19*delx,fourier_result,farfield_x);

```

```
Y=fft(wavefun_1p5_20aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_20a=a20*interp1(farfield_x-20*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_20bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_20b=a20*interp1(farfield_x+20*delx,fourier_result,farfield_x);


Y=fft(wavefun_1p5_21aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_21a=a21*interp1(farfield_x-21*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_21bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_21b=a21*interp1(farfield_x+21*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_22aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_22a=a22*interp1(farfield_x-22*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_22bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_22b=a22*interp1(farfield_x+22*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_23aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_23a=a23*interp1(farfield_x-23*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_23bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_23b=a23*interp1(farfield_x+23*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_24aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_24a=a24*interp1(farfield_x-24*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_24bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_24b=a24*interp1(farfield_x+24*delx,fourier_result,farfield_x);

Y=fft(wavefun_1p5_25aextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
scaledresult_1p5_25a=a25*interp1(farfield_x-25*delx,fourier_result,farfield_x);
Y=fft(wavefun_1p5_25bextended);
farfieldintensity = real( Y .* conj( Y ) );
farfield_pattern=fftshift( farfieldintensity );
fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
```

```
686  scaledresult_1p5_25b=a25*interp1(farfield_x+25*delx,fourier_result,farfield_x);
687
688  Y=fft(wavefun_1p5_26aextended);
689  farfieldintensity = real( Y .* conj( Y ) );
690  farfield_pattern=fftshift( farfieldintensity );
691  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
692  scaledresult_1p5_26a=a26*interp1(farfield_x-26*delx,fourier_result,farfield_x);
693  Y=fft(wavefun_1p5_26bextended);
694  farfieldintensity = real( Y .* conj( Y ) );
695  farfield_pattern=fftshift( farfieldintensity );
696  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
697  scaledresult_1p5_26b=a26*interp1(farfield_x+26*delx,fourier_result,farfield_x);
698
699  Y=fft(wavefun_1p5_27aextended);
700  farfieldintensity = real( Y .* conj( Y ) );
701  farfield_pattern=fftshift( farfieldintensity );
702  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
703  scaledresult_1p5_27a=a27*interp1(farfield_x-27*delx,fourier_result,farfield_x);
704  Y=fft(wavefun_1p5_27bextended);
705  farfieldintensity = real( Y .* conj( Y ) );
706  farfield_pattern=fftshift( farfieldintensity );
707  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
708  scaledresult_1p5_27b=a27*interp1(farfield_x+27*delx,fourier_result,farfield_x);
709
710  Y=fft(wavefun_1p5_28aextended);
711  farfieldintensity = real( Y .* conj( Y ) );
712  farfield_pattern=fftshift( farfieldintensity );
713  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
714  scaledresult_1p5_28a=a28*interp1(farfield_x-28*delx,fourier_result,farfield_x);
715  Y=fft(wavefun_1p5_28bextended);
716  farfieldintensity = real( Y .* conj( Y ) );
717  farfield_pattern=fftshift( farfieldintensity );
718  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
719  scaledresult_1p5_28b=a28*interp1(farfield_x+28*delx,fourier_result,farfield_x);
720
721  Y=fft(wavefun_1p5_29aextended);
722  farfieldintensity = real( Y .* conj( Y ) );
723  farfield_pattern=fftshift( farfieldintensity );
724  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
725  scaledresult_1p5_29a=a29*interp1(farfield_x-29*delx,fourier_result,farfield_x);
726  Y=fft(wavefun_1p5_29bextended);
727  farfieldintensity = real( Y .* conj( Y ) );
728  farfield_pattern=fftshift( farfieldintensity );
729  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
730  scaledresult_1p5_29b=a29*interp1(farfield_x+29*delx,fourier_result,farfield_x);
731
732  Y=fft(wavefun_1p5_30aextended);
733  farfieldintensity = real( Y .* conj( Y ) );
734  farfield_pattern=fftshift( farfieldintensity );
735  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
736  scaledresult_1p5_30a=a30*interp1(farfield_x-30*delx,fourier_result,farfield_x);
737  Y=fft(wavefun_1p5_30bextended);
738  farfieldintensity = real( Y .* conj( Y ) );
739  farfield_pattern=fftshift( farfieldintensity );
740  fourier_result=abs(farfield_pattern)/max(abs(farfield_pattern));
741  scaledresult_1p5_30b=a30*interp1(farfield_x+30*delx,fourier_result,farfield_x);
742
743  %%
744
745  totalscaledresult=scaledresult_1p5_0+scaledresult_1p5_1a+scaledresult_1p5_1b+scale
     dresult_1p5_2a+scaledresult_1p5_2b+...
746
                                          scaledresult_1p5_3a+scaledresult_1p5_3b+scale
                                          dresult_1p5_4a+scaledresult_1p5_4b+...
747
```

```
                                    scaledresult_1p5_5a+scaledresult_1p5_5b+scale
                                    dresult_1p5_6a+scaledresult_1p5_6b+...
748
                                    scaledresult_1p5_7a+scaledresult_1p5_7b+scale
                                    dresult_1p5_8a+scaledresult_1p5_8b+...
749
                                    scaledresult_1p5_9a+scaledresult_1p5_9b+scale
                                    dresult_1p5_10a+scaledresult_1p5_10b+...
750
                                    scaledresult_1p5_11a+scaledresult_1p5_11b+sca
                                    ledresult_1p5_12a+scaledresult_1p5_12b+...
751
                                    scaledresult_1p5_13a+scaledresult_1p5_13b+sca
                                    ledresult_1p5_14a+scaledresult_1p5_14b+...
752
                                    scaledresult_1p5_15a+scaledresult_1p5_15b+sca
                                    ledresult_1p5_16a+scaledresult_1p5_16b+...
753
                                    scaledresult_1p5_17a+scaledresult_1p5_17b+sca
                                    ledresult_1p5_18a+scaledresult_1p5_18b+...
754
                                    scaledresult_1p5_19a+scaledresult_1p5_19b+sca
                                    ledresult_1p5_20a+scaledresult_1p5_20b+...
755
                                    scaledresult_1p5_21a+scaledresult_1p5_21b+sca
                                    ledresult_1p5_22a+scaledresult_1p5_22b+...
756
                                    scaledresult_1p5_23a+scaledresult_1p5_23b+sca
                                    ledresult_1p5_24a+scaledresult_1p5_24b+...
757
                                    scaledresult_1p5_25a+scaledresult_1p5_25b+sca
                                    ledresult_1p5_26a+scaledresult_1p5_26b+...
758
                                    scaledresult_1p5_27a+scaledresult_1p5_27b+sca
                                    ledresult_1p5_28a+scaledresult_1p5_28b+...
759
                                    scaledresult_1p5_29a+scaledresult_1p5_29b+sca
                                    ledresult_1p5_30a+scaledresult_1p5_30b;
760
761   %%
762   %plot(farfield_x,fourier_result_coherent,'k')
763   hold on
764   %plot(farfield_x,fourier_result_1p5_0,'b')
765   plot(farfield_x,totalscaledresult./max(totalscaledresult),'m')
766   xlim([-30 30])
767   hold off
768   %%
769
770   plot(totalscaledresult,'m')
771   hold on
772   xlim([1950 2050])
773   hold off
774
775
776   %%
777
778   format long
779   f =
      fit(transpose(farfield_x(1950:2050)),transpose(totalscaledresult(1950:2050)),'gaus
      s1');
780   coeffvals = coeffvalues(f);
781   width=2*sqrt(log(2))*coeffvals(3);
```

```
782   length=.1*72/width % = 0.618384553064904
783
784
```

APPENDIX C

THEORETICAL METHOD FOR CLASSICAL BEAM PROPAGATION AND CALCULATION OF TRANSVERSE COHERENCE LENGTH

C.1 Classical Simulation Perpendicular to the Surface

The electron distribution measured at the detector in the Y-direction is modeled (perpendicular to the plane of the surface) by classical simulation. The simulation starts with a distribution of initial positions and momentum which is defined by the 1st and 2nd slits. With a 1st slit of height of 19 μm and a second slit with a height of 12.8 μm separated by 25 cm, this provides a beam with a divergence of ~120 μrad. This produces a small transverse coherence length in the y-direction (~250 nm), therefore a classical approach for motion in the y-direction is appropriate.

Therefore, the initial positions $y(0)$ and corresponding initial velocities $v_y(0)$ at the beginning of the surface are

$$\begin{aligned} y(0) &= s_1 + b\cdot(25\text{ cm} + 6\text{ cm});\ b = (s_2 - s_1)/(25\text{ cm}) \\ v_y(0) &= v_0 \frac{b}{\sqrt{1+b^2}} \\ z(0) &= 0 \\ v_z(0) &= v_0 \frac{1}{\sqrt{1+b^2}}, \end{aligned} \tag{C.1}$$

with $-9.5\ \mu\text{m} \le s_1 \le 9.5\ \mu\text{m}$, $-6.4\ \mu\text{m} \le s_2 \le 6.4\ \mu\text{m}$ in steps of 10 nm.

Over the 1 cm surface, the electron evolves per newton's equations of motion with a force in the y-direction due to image charge. Thus, the kinematic equations are

$$\begin{aligned} \dot{y}\left(t_i\right) &= v_y(t_i) \\ \dot{v}_y\left(t_i\right) &= -\frac{kq^2}{4m_e\left(y\left(t_i\right)+H\right)^2} \text{ (over surface)} \\ \dot{z}\left(t_i\right) &= v_z \text{ (const)} \\ \dot{v}_z\left(t_i\right) &= 0, \end{aligned} \tag{C.2}$$

where H is the height of the surface. Only electrons which do not contact the surface are considered (i.e. trajectories that collide within 10 nm of the surface are thrown out). This is done in time steps of ~825 femtoseconds (the total time of flight over the surface is ~413 picoseconds).

After reaching the end of the surface the electron undergoes free propagation another 23 cm to the detection plan, that is

$$\begin{aligned} \dot{y}\left(t_i\right) &= v_y(t_i) \\ \dot{v}_y\left(t_i\right) &= 0 \text{ (free propogation)} \\ \dot{z}\left(t_i\right) &= v_z \text{ (const)} \\ \dot{v}_z\left(t_i\right) &= 0. \end{aligned} \tag{C.3}$$

For computation time and simplicity this is completed in only one time-step: $\Delta t = \left(23 \text{ cm}\right)/v_z$. The final positions of the electrons are binned into 600 nm intervals along the y-direction. After the accumulation of all the trajectories that reach the detection plane, an electron distribution is produced in the Y-direction that can be compared to experimental results.

In the simulation, the surface is cut in at a height H= -590 nm below the central axis of the electron beam resulting in only 1/3 of the original electron beam flux making it to the detector. This distribution is compared to the case of the electric image charge force turned off, as well as when the surface is brought far away from the beam. From this it can be shown that the classical simulation of propagation of this electron beam with a travelling image charge well-approximates what is observed experimentally at the detector (see Figure C.1).

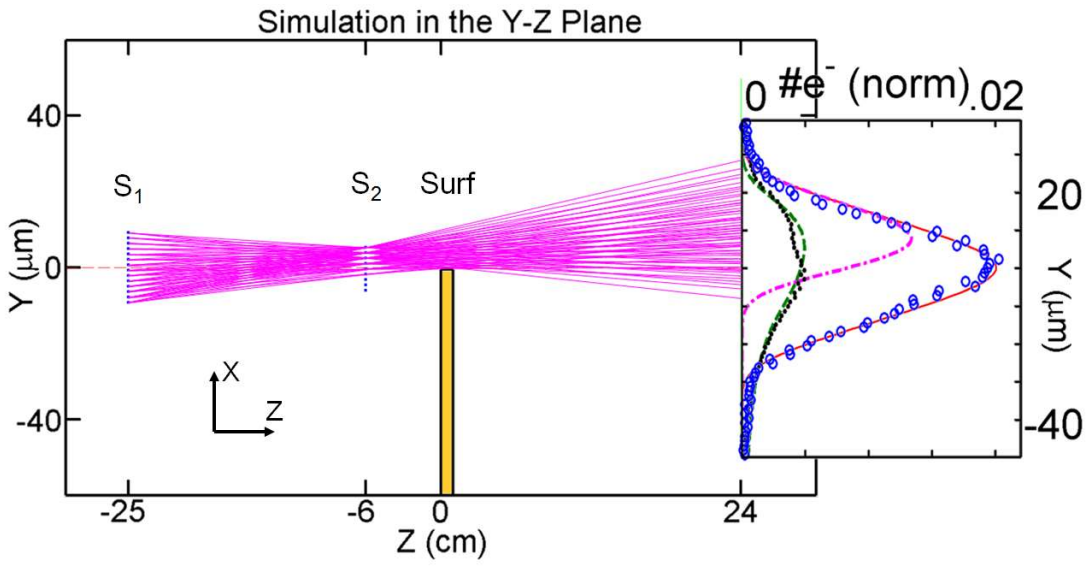

Figure C.1: Classical Simulation in the Y-Z Plane. Left: The Initial positions and velocities at the surface (Surf) are prepared corresponding to ballistic motion from the two collimating slits (S1 and S2). The electron then propagates over the surface with an image charge attraction in the Y-direction, and afterward freely propagates to the detector. The trajectories imaged are a sample of the case those when the surface cuts the beam but no image charge is present. Right: Simulation When there is no wall (solid red line) compared to when the surface is raised to cut 1/3 of the beam (dashed and dotted-dashed lines). When the surface is not present, the experimentally observed distribution (blue open dots) closely fits the simulation when no surface is present. When experimentally the surface cuts 1/3 of the electron flux (black closed dots), the distribution closely fits the simulation when image charge is present (green dashed line) as opposed to no image charge (pink dashed dotted line).

C.2 Quantum Decoherence Simulation Parallel to the Surface

To compute the evolution of the electron's density matrix in the x-direction as it passes over the surface, we first prepare the initial density matrix of the free electron by considering a partially coherent Gaussian beam,

$$\rho_{initial}(x,x') = \frac{1}{\sigma_0^{coh}\sqrt{2\pi}} \exp\left(-\frac{(x-x_0)^2}{2(\sigma_0^{coh})^2}\right) \exp\left(-\frac{(x')^2}{2(\sigma'_{initial})^2}\right). \tag{C.4}$$

Here x and x' describe the coordinates of the matrix element in the direction of the diagonal and in the direction orthogonal to the diagonal respectively (Figure C.2). The position x_0 indicates the center of the Gaussian. The width of the Gaussian in the x' direction, $w'_{initial} \equiv 2\sqrt{2\ln(2)}\sigma'_{initial}$, is proportional to the transverse coherence length. The spatial width along x, $w \equiv 2\sqrt{2\ln(2)}\sigma_0^{coh}$ is determined by a path integral simulation taking into consideration propagation through the first two collimation slits and reaching the beginning of the surface [28]. Note that if $w'_{initial}$ equals w, then the initial beam is fully coherent. If $w'_{initial}$ is smaller than w, then the initial beam is partially coherent as in Figure C.2 (left).

The initial state of the electron $\rho_{initial}$ now starts right before the surface at time t_i. We model the change in transverse coherence length of the electron over the surface due to a given decoherence process by considering the evolution of the density matrix of the electron. It changes according to [33]:

$$\rho_{final} = \rho_{initial} e^{-\int_{t_i}^{t_f} dt/\tau_{dec}}, \tag{C.5}$$

where the decoherence time scale τ_{dec} is model-dependent and depends on Δx and $y(t)$. When computing the integral in equation C.5 the simulated trajectories $y(t)$ are then inserted.

Each element in the density matrix is computed according to equation C.5 with the corresponding model considered. Note that $2x' \equiv \Delta x$ is the distance between symmetric elements across the main diagonal, and equals the variable Δx used in the models. The final state of the electron ρ_{final} is now right after the surface at time t_f. The density of the electron after the surface now has the form

$$\rho_{final}(x,x') = \frac{1}{\sigma_0^{coh}\sqrt{2\pi}} \exp\left(-\frac{(x-x_0)^2}{2(\sigma_0^{coh})^2}\right) \exp\left(-\frac{(x')^2}{2(\sigma'_{final})^2}\right), \tag{C.6}$$

when the width of the final state orthogonal to the diagonal is smaller than the width of the initial state ($w'_{final} < w'_{initial}$) then decoherence has occurred.

$$\rho_{final} = \rho_{initial} e^{-\int_{t_i}^{t_f} dt / \tau_{dec}(\Delta x, y(t))}$$

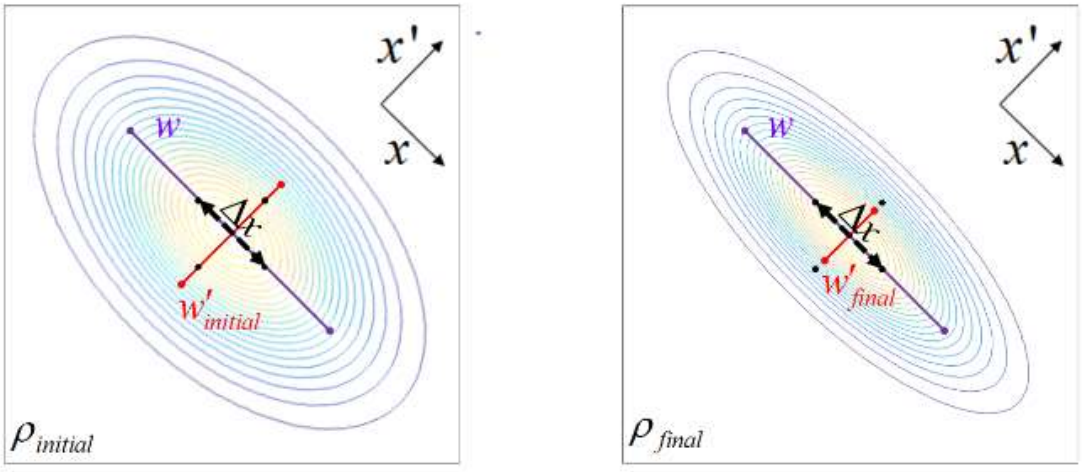

Figure C.2: Evolution of Density Matrix when Propagating over the Surface. As a result of decoherence, the initial state (left) evolves such that the off-diagonal elements reduce in amplitude. Hence the state's width w' decrease ($w'_{initial} \geq w'_{final}$).

For each final position bin at the detector where electrons landed, the final density matrix at the detector is computed by incoherently adding the individual matrices of each electron that reaches that bin. Making use of the ability to write a partial coherent state as a sum of coherent (i.e. pure) states (see Figure C.3),

$$\rho_{final} = \sum_{n=1}^{\infty} c_n \rho_n^{coh} , \tag{C.7}$$

then we can write ρ_{final} as a sum of Gaussian coherent states,

$$\rho_{final} \approx \sum_{n=1}^{N} \exp\left(-\frac{(x_0 - x_n)^2}{2(\sigma_{env})^2} \right) \times \exp\left(-\frac{(x - x_0 - x_n)^2 + (x')^2}{2(\sigma_2^{coh})^2} \right), \tag{C.8}$$

where $\sigma_2^{coh} = \sigma'_{final}$ describes the width of the reduced pure states after decoherence, and

$$\sigma_{env} = \sqrt{(\sigma_0^{coh})^2 - (\sigma_2^{coh})^2} \tag{C.9}$$

is the width of the envelope of the convolution.

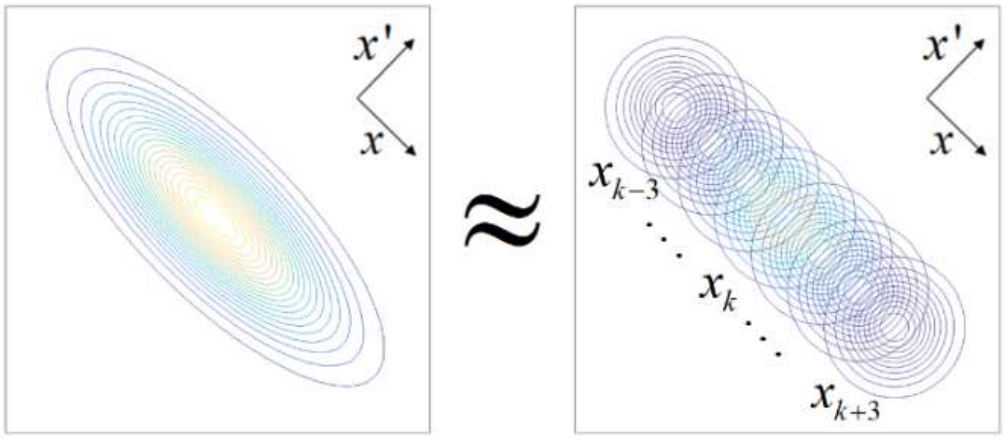

Figure C.3 Deconvolution of a Partial Coherent (Mixed) State by a Series of Coherent (Pure) States.

The initial state is now propagated to the state right after the surface. Next, each wave function corresponding to one of the reduced pure states is acted upon by a grating function (emulating the nanofabricated grating) followed by a Fourier Transform to determine the far field pattern. This is repeated for each of the reduced pure states and the resulting probability distribution patterns give the far field diffraction pattern (Figure C.4 bottom right). It is from this final pattern that a transverse coherence width $L_{coh}(y_{\text{detector}})$ is computed using $L_{coh} \approx \lambda_{dB}/\theta_{coll} \approx ad/w_{FWHM}$, where *a* is the periodicity of the grating, w_{FWHM} is the width of the computed diffraction peaks in the far field, and *d* is the distance between diffraction peaks . It is these values $L_{coh}(y_{\text{detector}})$ which produce the theoretical curves in the Figures 4.3 and 4.4.

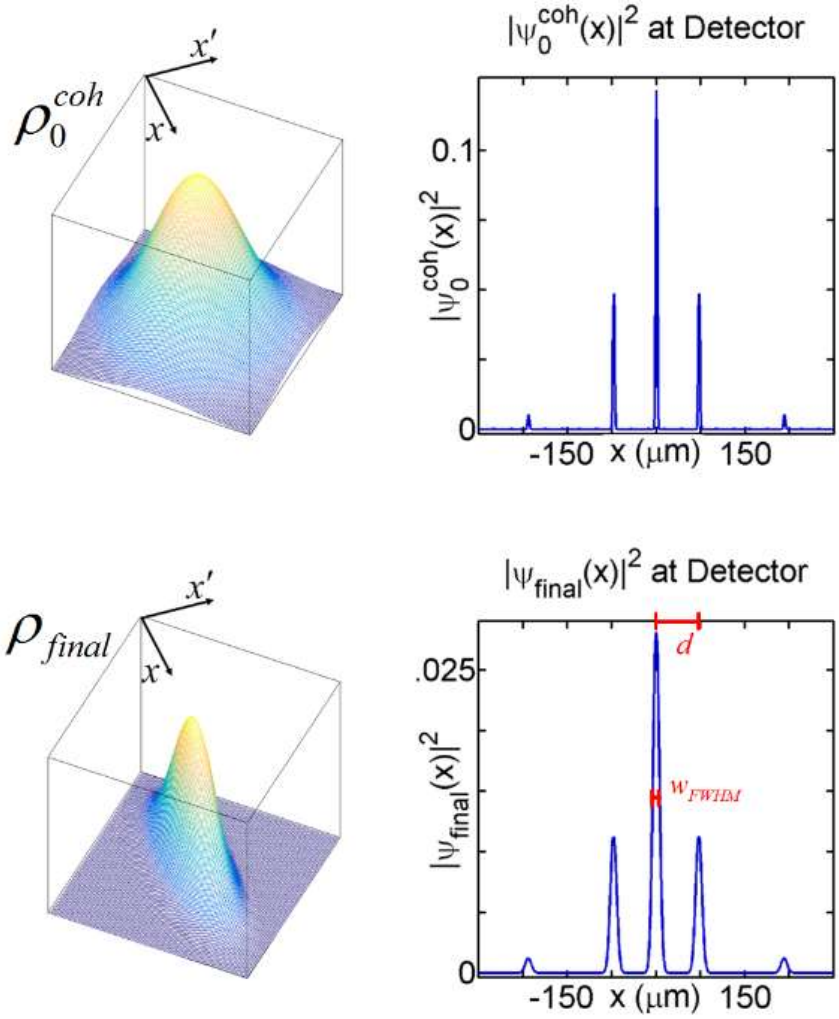

Figure C.4: Reduction of Coherence Elements in the Density Matrix and Corresponding Decrease in Coherence Length. Top Left: Density matrix of a coherent Gaussian electron beam. Top Right: Grating diffraction pattern in the far field after Fourier transformation of coherent state. Bottom Left: Final density matrix after decoherence evolution according to Equation 2. Bottom Right: Grating diffraction pattern in the far field after Fourier transformation of deconvoluted partial coherent state. Notice the stark difference in the widths in the diffraction peaks for the case of the fully coherent case (Top Right) as compared to the partial coherent case (Bottom Right).

C.3 Fortran Code for Classical Beam Propagation and Coherence Propagation

The following is the Fortran code used to produce the classical vertical electron distributions as the electron passes over a surface and lands on the detector, and the corresponding loss of coherence computation based on the various theoretical models outlined in Chapter 2.6 . See the flowchart below (Figure C.5) and the description of the program outlined in Appendix C.2.

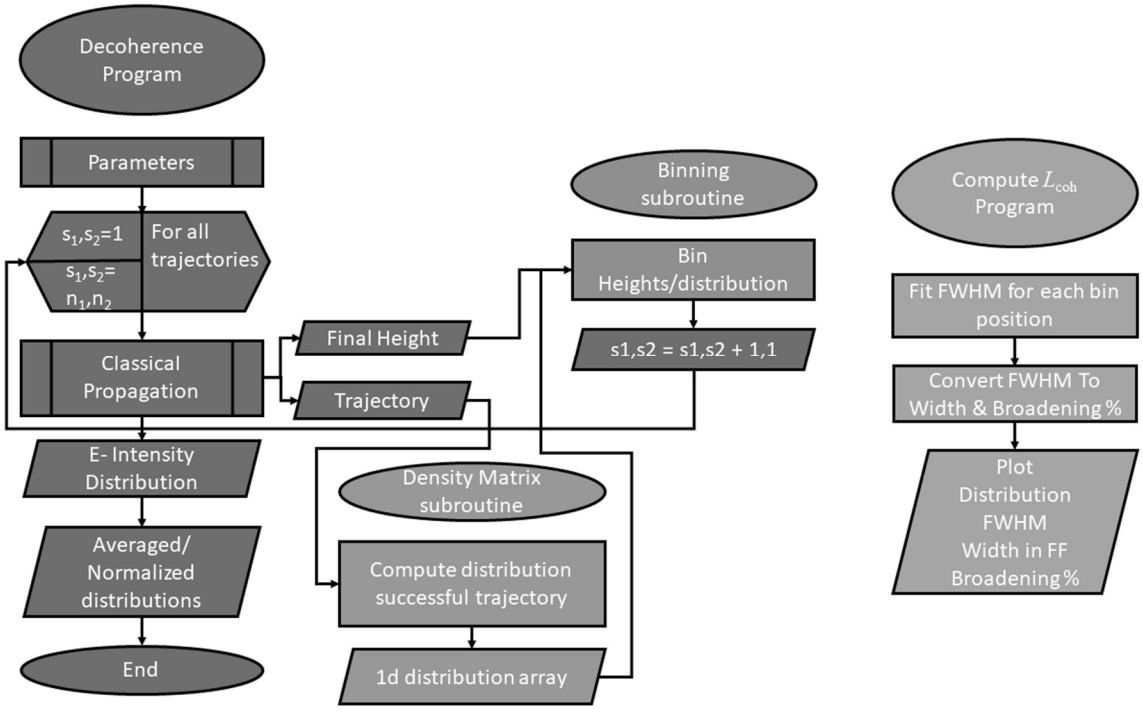

Figure C.5. Flowchart of Fortran Code for Classical Beam Propagation and Coherence Propagation

```
#!/bin/sh
#SBATCH --ntasks=7
#SBATCH --mem-per-cpu=1024
#SBATCH --time=01:00:00
#SBATCH --job-name=Fortran
#SBATCH --error=scheel.%J.err
#SBATCH --output=scheel.%J.out

mpirun ./test.x

```

```
1   PROGRAM Decoherence
2   !====== MPI =====
3       use mpi
4   !================
5
6       IMPLICIT NONE
7       common/par/ me,q,k,v0,theta,length
8       real*8 me,q,k,v0,theta,finalheight,trajectory(500)
9       real*8 length
10      real*8 rho_1p5(2001,2001),near_field_x(2001),reduced_diagonal(2001)
11      real ratio
12      integer s1,s2,flag
13      integer counter,sucesses
14      integer Nbin
15          integer, parameter :: N = 549
16      parameter(Nbin=1100)
17      real*8 binvalue(Nbin),gratingarray(32)
18      real*8 accum_reduced_diagonal(Nbin,2001)
19      real*8 binlower(Nbin),binupper(Nbin),binposition(Nbin)
20
21  !============================== MPI ================================
22      integer ind
23      real*8, dimension(:), allocatable :: y_local
24      integer numnodes,myid,rc,ierr,start_local,end_local,N_local
25      real*8 allsum
26
27      character(len=8) :: fmt !format descriptor
28      character(5) x1
29      fmt = '(I5.5)' ! an integer of width 5 with zeros at the left
30
31  !===================================================================
32  !============================== MPI ================================
33      call mpi_init( ierr )
34      call mpi_comm_rank ( mpi_comm_world, myid, ierr )
35      call mpi_comm_size ( mpi_comm_world, numnodes, ierr )
36
37      write(x1,fmt) myid !converting integer to string using an 'internal file'
38
39  !    OPEN(UNIT=15,FILE='off_diagonal_dist_scheel456'//trim(x1)//'.dat')
40
41
42
43      N_local = N/numnodes
44  !    allocate ( y_local(N_local) )
45
46  !    start_local = N_local*myid + 1 + 732
47  !    end_local =  N_local*myid + N_local + 732
48
49      start_local = N_local*myid + 1
50      end_local =  N_local*myid + N_local
51
52  !===================================================================
53
54
55
56          OPEN(UNIT=11,FILE='propogation_curvedsurface.dat')
57          OPEN(UNIT=12,FILE='trajectory_curvedsurface.dat')
58          OPEN(UNIT=13,FILE='debug.dat')
59                  OPEN(UNIT=15,FILE='off_diagonal_dist_howie456'//trim(x1)//'.dat')
60          OPEN(UNIT=16,FILE='paths_determination.dat')
61          OPEN(UNIT=17,FILE='initial_coherence_distribution.dat')
62
63          call parameters
64      call gratinginitialization(gratingarray)
65      call initialdensitymatrix(rho_1p5,near_field_x)
```

```
66      call
        bininitialization(Nbin,binlower,binupper,binvalue,binposition,accum_reduced_di
        agonal)
67      call write_rho(rho_1p5,near_field_x)
68
69      counter=0
70      sucesses=0
71  !        do s1=1,1901
72  !        do s2=1,1281
73       do s1=1,1901
74           do s2=start_local,end_local
75         call
           propagate(trajectory,finalheight,s1,s2,counter,sucesses,flag,gratingarray)
76          if (flag.eq.0) then
77
78           call howiedecoherence(trajectory,reduced_diagonal,rho_1p5)
79           !call scheeldecoherence(trajectory,reduced_diagonal,rho_1p5)
80           !call zurekdecoherence(trajectory,reduced_diagonal,rho_1p5)
81           !call machdecoherence(trajectory,reduced_diagonal,rho_1p5)
82           !call hasslebachdecoherence(trajectory,reduced_diagonal,rho_1p5)
83
84           call
             binning(finalheight,reduced_diagonal,Nbin,binlower,binupper,binvalue,accum
             _reduced_diagonal,s1,s2)
85
86          endif
87      enddo !end of slit2 do loop
88           enddo !end of slit1 do loop
89          write(6,*) 'total runs= ', counter
90          write(6,*) 'total landing hits= ', sucesses
91  !        ratio=sucesses*100.0/counter
92      ratio=sucesses*100.0/1623398.0
93          write(6,*) 'percent of beam= ', ratio
94      call writeelectrondistribution(binvalue,binposition,Nbin)
95      call writeoffdiagonal(binposition,accum_reduced_diagonal,Nbin)
96           close(11)
97           close(13)
98           close(14)
99           close(15)
100          close(16)
101          close(17)
102
103 !========================================== MPI
    ================================
104     call mpi_finalize(rc)
105 !================================================================================
    ====
106
107 Stop
108 End Program
109
110
111 !================================================================================
    ===
112 !================================================================================
    ===
113 !================================================================================
    ===
114 !================================================================================
    ===
115
116     subroutine parameters
117     implicit none
118     common/par/ me,q,k,v0,theta,length
119     real*8 me,q,k,v0,theta,length
```

```
120
121       me=9.11d-31
122       q=(1.6d-19)*.8425d0
123       k=8.988d9
124       !v0=1.676d7 !corresponds to a .8keV beam
125       v0=2.422d7  !corresponds to a 1.67keV beam
126           length=1.00d-2
127
128
129       return
130       end
131
132
133       subroutine
          propagate(trajectory,finalheight,s1,s2,counter,sucesses,flag,gratingarray)
134       implicit none
135
136       real*8 endoftim,delt
137       integer ido,tel,i,s1,s2,flag,counter,j,sucesses
138       real*8 fcn,t,tend,tft,y(4)
139       common/par/ me,q,k,v0,theta,length
140       real*8 me,q,k,v0,yprime(4),tfinal
141       real*8 length,surfheight,theta,finalheight
142       real*8 slit1position,slit2position,cube
143       external fcn,divprk,sset
144       real*8 trajectory(500)!,sucesstraj(40000,500)
145       real*8 gratingarray(32)
146       real*8 chargeforce,deflection
147
148       do i=1,500
149          trajectory(i)=0d0
150       enddo
151
152
153       t=0.0
154       do i=1,4
155          y(i)=0d0
156       enddo
157
158       y(1)=0d0        !initial x coordinate
159
160       counter=counter+1
161       flag=0
162
163       slit1position=(-9.5d-6)+(s1-1)*(1d-8)
164       slit2position=(-6.4d-6)+(s2-1)*(1d-8)
165
166
167       theta=datan((slit2position-slit1position)/(25d-2))
168
169   !   deflection=
170   !       surfheight=-.59d-6 !-1d0+dsqrt(1d0+((y(1)-5.0d-4)*(y(1)-5.0d-4)))+3d-6
171       surfheight=-.81d-6
172
173
174         y(1)=0d0
175         y(2)=v0*dcos(theta)       !initial x velocity of beam
176         y(3)=slit1position+(31d-2)*((slit2position-slit1position)/(25d-2))
177       !initial y coordinate of beam,curved (1.67keV)
178         y(4)=v0*dsin(theta)       !initial y velocity of beam
179
180   !   write(11,111) y(3)
181
182   !   do i=1,16
183   !    if (y(3).ge.gratingarray(2*i-1)) then
```

```
!     if (y(3).le.gratingarray(2*i)) then
!       flag=1
!     endif
!    endif
!   enddo

!   chargeforce=deflection*me*v0*v0/(length*23.0d-2)

      yprime(1)=y(2)    !y(1)=x, y(2)=vx
      yprime(2)=0d0
      yprime(3)=y(4)    !y(3)=y, y(4)=vy
      yprime(4)=-k*q*q/(4*(y(3)+surfheight)*(y(3)+surfheight)*me)



      ido=1
      tel=0
      tft=0.0d0
      endoftim=length/v0

  if (s1.eq.1) then
    if (s2.eq.1) then
          write(6,*) 'initial=', surfheight
    endif
  endif

        delt=endoftim/500d0
      do i=1,500

            tft=tft+endoftim/500d0
        tend=tft

          y(1)=y(1)+(y(2)*delt)
         !y(2)=y(2)
          y(3)=y(3)+(y(4)*delt)
          y(4)=y(4)+(yprime(4)*delt)
            yprime(4)=-k*q*q/(4*(y(3)+surfheight)*(y(3)+surfheight)*me)
          surfheight=-.81d-6 !+(4*3.33333d-6)*(tft/endoftim)
          trajectory(i)=y(3)+surfheight
        if (y(3)+surfheight.le.5d-8) then
          flag=1
        endif
      enddo
  if (s1.eq.1) then
    if (s2.eq.1) then
      write(6,*) 'final=', surfheight
    endif
  endif
      !travel another 23cm, free propogation

          tfinal=(23d-2)/v0

         y(1)=y(1)+(y(2)*tfinal)
         y(3)=y(3)+(y(4)*tfinal)

      if (flag.eq.0) then

!     write(11,111) y(3)
      finalheight=y(3)
      sucesses=sucesses+1
!           do i=1,500
!            write(12,112) trajectory(i)
!           enddo
      endif
```

```
249
250   !   write (*,'(A,2x,I5)') '+ Number= ',  counter
251
252         !write(6,*) counter
253
254   !   write(6,*) 'finished loop'
255   !   write(6,*) 'total runs= ', counter
256   !   write(6,*) 'total landing hits= ', sucesses
257   !   ratio=sucesses*100.0/counter
258   !   write(6,*) 'percent of beam= ', ratio
259
260
261
262
263         format(1E15.8)
264       format(1E15.8)
265         format(3E15.8,1I12.6)
266         return
267       end
268
269   !---------------------------------------------------------------------
270
271   subroutine initialdensitymatrix(rho_1p5,near_field_x)
272   implicit none
273
274
275   real*8 rho_1(2001,2001),b,x,y,near_field_x(2001)
276   real*8 rho_1p5(2001,2001),fwhm1p67,beta,w,sigma
277   integer i,j
278
279   do i=1,2001
280    do j=1,2001
281      rho_1(i,j)=0d0
282      rho_1p5(i,j)=0d0
283    enddo
284   enddo
285
286   w=2.2745d0*dsqrt(2d0)
287   sigma=w/(2d0*dsqrt(2d0*dlog(2d0)))
288
289   b=1/(2d0*(sigma**2))
290
291   do i=1,2001
292    do j=1,2001
293     x=i*10d0/2000d0
294     y=j*10d0/2000d0
295     rho_1(i,j)=dexp(-b*((x-5.005)**2+(y-5.005)**2))
296    enddo
297   enddo
298
299   do i=1,2001
300    near_field_x(i)=-5d0+(i-1)*.005d0
301   enddo
302
303
304   fwhm1p67=215.44d0
305   beta=fwhm1p67/(2d0*dsqrt(dlog(2d0)))
306
307   do i=1,2001
308     do j=1,2001
309      rho_1p5(i,j)=rho_1(i,j)*exp(-((i-j)/beta)**2)
310     enddo
311   enddo
312
313
```

```
314   return
315   end
316
317   !-------------------------------------------------------------------------------
318   subroutine howiedecoherence(trajectory,reduced_diagonal,rho_1p5)
319   implicit none
320
321   real*8 trajectory(500),reduced_diagonal(2001),rho_1p5(2001,2001)
322   common/par/ me,q,k,v0,theta,length
323   real*8 me,q,k,v0,theta,length
324   real*8 delt
325   integer i,j,m
326   real*8 Pi,hbar,h,kbolt,delx
327   real*8 resistivity,resistivitygold,resistivitysilicon
328   real*8 conductivity
329   real*8 inversezurektime,P,numerator,coefficient,expintegral
330   real*8 A,B,howie,nu
331   real*8 omegam
332
333   omegam=.6d12 !for semiconductors
334
335   delt=length/v0/500d0    !assuming 500 steps (see propogation)
336   resistivitygold=2.2d-8
337   resistivitysilicon=.01d0
338   resistivity=resistivitysilicon !parameter
339   conductivity=1/resistivity
340   hbar=1.05457d-34
341   Pi=3.14159265d0
342   kbolt=1.38065d-23
343   h=hbar*2.0d0*Pi
344
345   coefficient=(q*q*length*omegam*omegam)/(4.0d0*Pi*Pi*hbar*conductivity*v0*v0)
346
347
348   do i=1,2001
349    reduced_diagonal(i)=0d0
350   enddo
351
352   do m=1,1
353    do i=890,1112 !it is unecessary to calculate the change at the tails, suff small
354      j=2002-i
355           delx=abs((i*(5d-6)/1000d0)-(j*(5d-6)/1000d0))
356           if(delx == 0.0d0) then
357             P=0.0d0
358           endif
359           if(delx /= 0.0d0) then
360             nu=trajectory(m)/(4.0d0*delx)
361             A=dlog((.65d0+(0.56146d0/nu))*(1+nu))
362             B=(nu**4)*exp(7.7d0*nu)*(2.0d0+nu)**(3.7)
363             expintegral=((A**(-7.7))+B)**(-.13)
364             P=coefficient*expintegral
365           endif
366       howie=dexp(-P/500d0)
367       reduced_diagonal(i)=rho_1p5(i,j)*howie
368    enddo
369   enddo
370
371   do m=2,500
372    do i=890,1112 !it is unecessary to calculate the change at the tails, suff small
373      j=2002-i
374           delx=abs((i*(5d-6)/1000d0)-(j*(5d-6)/1000d0))
375       if(delx == 0.0d0) then
376         P=0.0d0
377       endif
378       if(delx /= 0.0d0) then
```

```
379             nu=trajectory(m)/(4.0d0*delx)
380             A=dlog((.65d0+(0.56146d0/nu))*(1+nu))
381             B=(nu**4)*exp(7.7d0*nu)*(2.0d0+nu)**(3.7)
382             expintegral=((A**(-7.7))+B)**(-.13)
383             P=coefficient*expintegral
384         endif
385
386             howie=dexp(-P/500d0)
387             reduced_diagonal(i)=reduced_diagonal(i)*howie
388      enddo
389     enddo
390
391
392     return
393     end
394     !-------------------------------------------------------------------------------
        -
395     !-------------------------------------------------------------------------------
396     subroutine hasslebachdecoherence(trajectory,reduced_diagonal,rho_1p5)
397     implicit none
398
399     real*8 trajectory(500),reduced_diagonal(2001),rho_1p5(2001,2001)
400     common/par/ me,q,k,v0,theta,length
401     real*8 me,q,k,v0,theta,length
402     real*8 delt,b,a
403     integer i,j,m
404     real*8 Pi,hbar,h,kbolt,Tkelvin
405     real*8 resistivity,resistivitygold,resistivitysilicon
406     real*8 inversehasstime,hassexp,numerator
407
408     delt=length/v0/500d0    !assuming 500 steps (see propogation)
409
410     a=(10.9d0-5.8d0)*(1d-6)/(500d0)
411
412
413     do i=1,2001
414      reduced_diagonal(i)=0d0
415     enddo
416
417     do m=1,1
418      do i=890,1112 !it is unecessary to calculate the change at the tails, suff small
419        j=2002-i
420         b=a*((i*(5d-6)/1000d0)-(j*(5d-6)/1000d0))**2
421         hassexp=dexp(-b/(trajectory(m)**3))
422         reduced_diagonal(i)=rho_1p5(i,j)*hassexp
423      enddo
424     enddo
425
426     do m=2,500
427      do i=890,1112 !it is unecessary to calculate the change at the tails, suff small
428        j=2002-i
429             b=a*((i*(5d-6)/1000d0)-(j*(5d-6)/1000d0))**2
430             hassexp=dexp(-b/(trajectory(m)**3))
431         reduced_diagonal(i)=reduced_diagonal(i)*hassexp
432      enddo
433     enddo
434
435
436     return
437     end
438     !-------------------------------------------------------------------------------
        -
439     !-------------------------------------------------------------------------------
440     subroutine scheeldecoherence(trajectory,reduced_diagonal,rho_1p5)
441     implicit none
```

```
442
443   real*8 trajectory(500),reduced_diagonal(2001),rho_1p5(2001,2001)
444   common/par/ me,q,k,v0,theta,length
445   real*8 me,q,k,v0,theta,length
446   real*8 delt
447   integer i,j,m
448   real*8 Pi,hbar,h,kbolt,Tkelvin,epsnot
449   real*8 resistivity,resistivitygold,resistivitysilicon
450   real*8 coefficient,gamma,scheel,bracket,xsquared
451   real*8 rprimenot
452
453   delt=length/v0/500d0    !assuming 500 steps (see propogation)
454   resistivitygold=2.2d-8
455   resistivitysilicon=.01d0
456   resistivity=resistivitysilicon !parameter
457   hbar=1.05457d-34
458   Pi=3.14159265d0
459   kbolt=1.38065d-23
460   Tkelvin=300.0d0
461   h=hbar*2.0d0*Pi
462   epsnot=8.85d-12
463
464
465   rprimenot=(2.0d0*epsnot)*resistivity
466
467   coefficient=(q*q*kbolt*Tkelvin*rprimenot)/(2.0d0*Pi*epsnot*hbar*hbar)
468
469
470   do i=1,2001
471    reduced_diagonal(i)=0d0
472   enddo
473
474   do m=1,1
475    do i=890,1112 !it is unecessary to calculate the change at the tails, suff small
476      j=2002-i
477       xsquared=((i*(5d-6)/1000d0)-(j*(5d-6)/1000d0))**2
478       bracket=(1/(2.0d0*trajectory(m)))-1/sqrt((4.0d0*trajectory(m)**2)+xsquared)
479       gamma=-delt*coefficient*bracket
480       scheel=dexp(gamma)
481       reduced_diagonal(i)=rho_1p5(i,j)*scheel
482    enddo
483   enddo
484
485   do m=2,500
486    do i=890,1112 !it is unecessary to calculate the change at the tails, suff small
487      j=2002-i
488       xsquared=((i*(5d-6)/1000d0)-(j*(5d-6)/1000d0))**2
489       bracket=(1/(2.0d0*trajectory(m)))-1/sqrt((4.0d0*trajectory(m)**2)+xsquared)
490       gamma=-delt*coefficient*bracket
491       scheel=dexp(gamma)
492       reduced_diagonal(i)=reduced_diagonal(i)*scheel
493    enddo
494   enddo
495
496
497   return
498   end
499   !-------------------------------------------------------------------------------
      -
500   !-------------------------------------------------------------------------------
501   subroutine zurekdecoherence(trajectory,reduced_diagonal,rho_1p5)
502   implicit none
503
504   real*8 trajectory(500),reduced_diagonal(2001),rho_1p5(2001,2001)
505   common/par/ me,q,k,v0,theta,length
```

```
506   real*8 me,q,k,v0,theta,length
507   real*8 delt
508   integer i,j,m
509   real*8 Pi,hbar,h,kbolt,Tkelvin
510   real*8 resistivity,resistivitygold,resistivitysilicon
511   real*8 inversezurektime,zurek,numerator
512
513   delt=length/v0/500d0    !assuming 500 steps (see propogation)
514   resistivitygold=2.2d-8
515   resistivitysilicon=.01d0
516   resistivity=resistivitysilicon !parameter
517   hbar=1.05457d-34
518   Pi=3.14159265d0
519   kbolt=1.38065d-23
520   Tkelvin=300.0d0
521   h=hbar*2.0d0*Pi
522
523   do i=1,2001
524    reduced_diagonal(i)=0d0
525   enddo
526
527   do m=1,1
528    do i=890,1112 !it is unecessary to calculate the change at the tails, suff small
529      j=2002-i
530
        numerator=(Pi*(q**2)*kbolt*Tkelvin*resistivity*((i*(5d-6)/1000d0)-(j*(5d-6)/10
        00d0))**2)
531         inversezurektime=numerator/((4d0*h**2)*(trajectory(m)**3))
532       zurek=dexp(-delt*inversezurektime)
533       reduced_diagonal(i)=rho_1p5(i,j)*zurek
534    enddo
535   enddo
536
537   do m=2,500
538    do i=890,1112 !it is unecessary to calculate the change at the tails, suff small
539      j=2002-i
540
           numerator=(Pi*(q**2)*kbolt*Tkelvin*resistivity*((i*(5d-6)/1000d0)-(j*(5d-6
           )/1000d0))**2)
541           inversezurektime=numerator/((4d0*h**2)*(trajectory(m)**3))
542           zurek=dexp(-delt*inversezurektime)
543       reduced_diagonal(i)=reduced_diagonal(i)*zurek
544    enddo
545   enddo
546
547
548   return
549   end
550   !-------------------------------------------------------------------------------
      -
551   !-------------------------------------------------------------------------
552   subroutine machdecoherence(trajectory,reduced_diagonal,rho_1p5)
553   implicit none
554
555   real*8 trajectory(500),reduced_diagonal(2001),rho_1p5(2001,2001)
556   common/par/ me,q,k,v0,theta,length
557   real*8 me,q,k,v0,theta,length
558   real*8 delt
559   integer i,j,m
560   real*8 kfermi,epsi
561   real*8 Pi,hbar,h,kbolt,Tkelvin,epsnot
562   real*8 resistivity,resistivitygold,resistivitysilicon
563   real*8 coefficient,mach,xsquared
564   real*8 lambdadb,taur_inverse,taud_inverse
565
```

```
566   delt=length/v0/500d0    !assuming 500 steps (see propogation)
567   resistivitygold=2.2d-8
568   resistivitysilicon=.01d0
569   resistivity=resistivitysilicon !parameter
570   hbar=1.05457d-34
571   Pi=3.14159265d0
572   kbolt=1.38065d-23
573   Tkelvin=300.0d0
574   h=hbar*2.0d0*Pi
575   epsnot=8.85d-12
576   epsi=2.0d0
577   kfermi=1.2d10
578
579   lambdadb=2.0d0*Pi*hbar/dsqrt(2.0d0*me*kbolt*Tkelvin)
580
581   coefficient=(q**2)/(2*Pi*epsnot*epsi*hbar*kfermi)
582
583   do i=1,2001
584    reduced_diagonal(i)=0d0
585   enddo
586
587   do m=1,1
588    do i=890,1112 !it is unecessary to calculate the change at the tails, suff small
589      j=2002-i
590       xsquared=((i*(5d-6)/1000d0)-(j*(5d-6)/1000d0))**2
591       taur_inverse=coefficient/(trajectory(m)**2)
592       taud_inverse=(Pi/32.0d0)*taur_inverse*xsquared/(lambdadb)**2
593       mach=dexp(-delt*taud_inverse)
594       reduced_diagonal(i)=rho_1p5(i,j)*mach
595    enddo
596   enddo
597
598   do m=2,500
599    do i=890,1112 !it is unecessary to calculate the change at the tails, suff small
600      j=2002-i
601           xsquared=((i*(5d-6)/1000d0)-(j*(5d-6)/1000d0))**2
602           taur_inverse=coefficient/(trajectory(m)**2)
603           taud_inverse=(Pi/32.0d0)*taur_inverse*xsquared/(lambdadb)**2
604           mach=dexp(-delt*taud_inverse)
605       reduced_diagonal(i)=reduced_diagonal(i)*mach
606    enddo
607   enddo
608
609
610   return
611   end
612   !-------------------------------------------------------------------------------
      -
613
614
615
616   subroutine
      bininitialization(Nbin,binlower,binupper,binvalue,binposition,accum_reduced_diagon
      al)
617   implicit none
618
619   integer Nbin,i,j
620   real*8 accum_reduced_diagonal(Nbin,2001),binvalue(Nbin)
621   real*8 binlower(Nbin),binupper(Nbin),binposition(Nbin)
622   real*8 lowerbound,binsize
623
624   lowerbound=-6d-4
625   binsize=(6d-4)/1000d0
626
627   do i=1,Nbin
```

```
628    binlower(i)=lowerbound+binsize*(i-1)
629    binupper(i)=lowerbound+binsize*i
630    binposition(i)=(binlower(i)+binupper(i))/2.0d0
631    binvalue(i)=0d0
632   enddo
633
634   do i=1,Nbin
635    do j=1,2001
636     accum_reduced_diagonal(i,j)=0d0
637    enddo
638   enddo
639
640   return
641   end
642
643   !-------------------------------------------------------------------------------
644
645   subroutine
      binning(finalheight,reduced_diagonal,Nbin,binlower,binupper,binvalue,accum_reduced
      _diagonal,s1,s2)
646   implicit none
647
648   integer Nbin,i,j,s1,s2
649   real*8 accum_reduced_diagonal(Nbin,2001)
650   real*8 binlower(Nbin),binupper(Nbin)
651   real*8 lowerbound,binsize,binvalue(Nbin)
652   real*8 reduced_diagonal(2001),finalheight,norm
653   norm=0d0
654
655   do i=1,2001
656     norm=norm+reduced_diagonal(i)
657   enddo
658
659   do i=1,2001
660     reduced_diagonal(i)=reduced_diagonal(i)/norm
661   enddo
662
663   do i=1,Nbin
664    if (binlower(i).le.finalheight) then
665     if (finalheight.le.binupper(i)) then
666      binvalue(i)=binvalue(i)+1d0
667   !   if (binlower(i).ge.-1.82d-5) then
668   !    if (binupper(i).le.1.82d-5) then
669   !      write(16,*) s1,' ',s2
670   !    endif
671   !   endif
672      do j=1,2001
673       accum_reduced_diagonal(i,j)=accum_reduced_diagonal(i,j)+reduced_diagonal(j)
674      enddo
675     endif
676    endif
677   enddo
678
679   return
680   end
681
682   !-------------------------------------------------------------------------------
683
684   subroutine writeelectrondistribution(binvalue,binposition,Nbin)
685   implicit none
686
687   integer Nbin,i
688   real*8 binposition(Nbin),binvalue(Nbin)
689
690   do i=1,Nbin
```

```
691    write(14,114) binposition(i), binvalue(i)
692   enddo
693
694           format(100E16.6E4)
695
696   return
697   end
698
699   !---------------------------------------------------------------
700
701   subroutine writeoffdiagonal(binposition,accum_reduced_diagonal,Nbin)
702   implicit none
703
704   integer Nbin,i,j
705   real*8 binposition(Nbin),accum_reduced_diagonal(Nbin,2001)
706
707   do i=1,Nbin
708    do j=890,1112
709     write(15,115) binposition(i), accum_reduced_diagonal(i,j)
710    enddo
711   enddo
712
713       format(100E16.6E4)
714
715   return
716   end
717
718   !---------------------------------------------------------------
719
720   subroutine gratinginitialization(gratingarray)
721   implicit none
722
723   real*8 gratingarray(32),offset
724   integer i
725   offset=(-11d-6)+.5d-6
726
727
728   do i=1,31,2
729   gratingarray(i)=(.75d-6)*(i-1)+offset
730   enddo
731
732   do i=2,32,2
733   gratingarray(i)=(.75d-6)*(i-2)+.5d-6+offset
734   enddo
735
736   do i=1,32
737   write(6,*) gratingarray(i)
738   enddo
739
740   return
741   end
742
743   !---------------------------------------------------------------
744
745
746   subroutine write_rho(rho_1p5,near_field_x)
747   implicit none
748
749   real*8 rho_1p5(2001,2001),near_field_x(2001)
750   integer i,j
751
752   do i=1,2001
753    j=2002-i
754    write(17,117) near_field_x(i),rho_1p5(i,j)
755   enddo
```

```
756
757
758             format(100E16.6E4)
759
760     return
761     end
762
```

[1] G. Gronniger, B. Barwick, H. Batelaan, T. Savas, D. Pritchard, and A. Cronin, Appl. Phys. Lett. **87**, 124104 (2005).
[2] H.-P. Breuer and F. Petruccione, in *Theory Open Quantum Syst.* (Oxford University Press, 2002), pp. 232–242.

APPENDIX D

SURFACES & GRATING; MOUNT DESIGN AND PREP

D.1 Summary

The following outlines details of the surface and grating mounts used to control the two, as well as the surface preparation needed to minimize sources of dephasing and other contamination in the experiment.

D.2 Surface and Grating Mounts

This experiment underwent two iterations of surface/grating mounts. Both are made primarily of aluminum to minimize the introduction of local magnetic fields.

The first was a monolithic, compact design that was top-loaded into the system (see Figures D.1-D.3). The free-standing grating sat on a lip which extended out by 1mm and whose top was gently held from falling over by a bronze clamp (as well as a ground channel to reduce charging of the grating bars). A large circular hole allows electrons to travel through the grating. The surface itself was separated by 2mm from the grating and held upside-down by two small aluminum clamps. Outside the vacuum chamber the feedthrough was supported by a translational stage which was used to control the height of the surface with respect to the electron beam, as well as a 2-D wobble mount, which provided control over the angle of the surface.

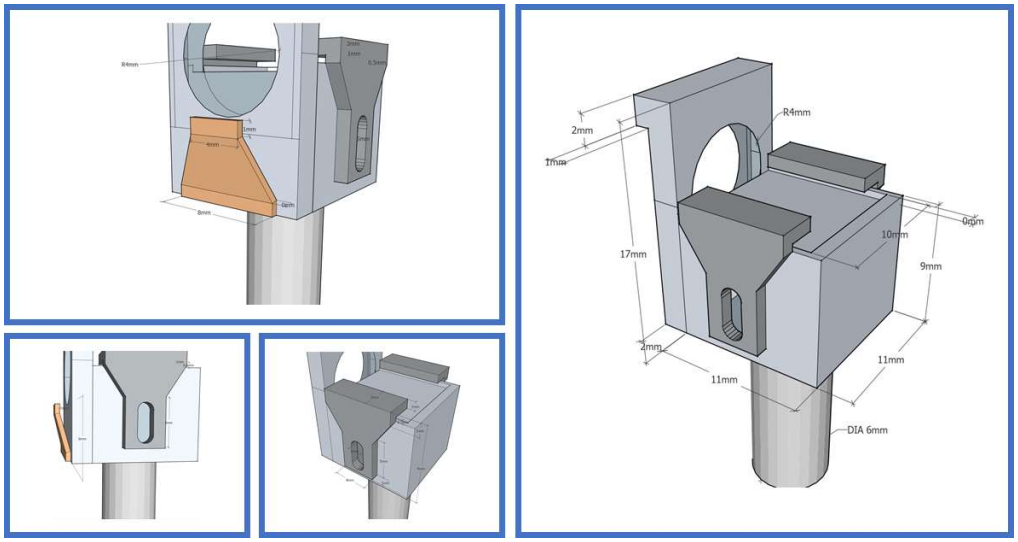

Figure D.1: Schematic of First Surface/Grating Mount

Figure D.2: Preparation Image of Gold Surface and Grating in Mount. Left: View of nanofabricated grating. Right: View of entire feedthrough (without the external translation stage and wobble mount)

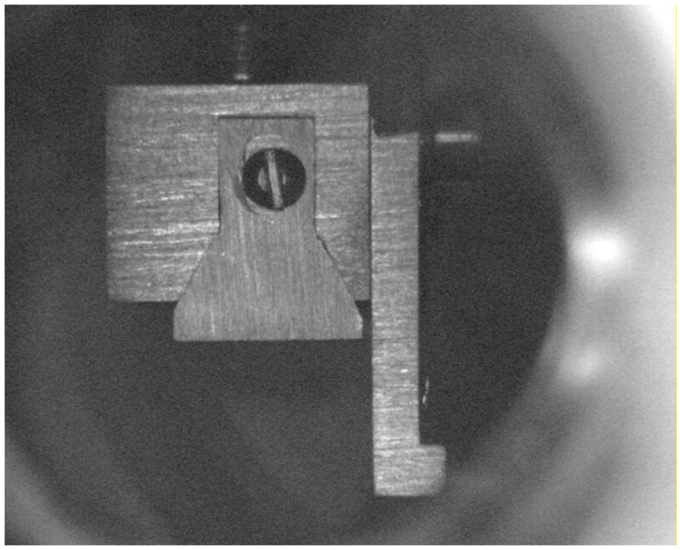

Figure D.3: In-Vacuum Image of Surface/Granting Mount.

Later, two new mounts were designed to hold the grating and slit independently (designed by Liyun Zhang). These mounts were held with separate feedthroughs that are loaded horizontally with respect to the ground The Grating Mount is similar in functionality as the grating had in the first design, except that its angle is held fixed and it has the ability to translate two directions: in the x-transverse direction with respect to the electron beam (and perpendicular to the direction the grating bars), and in the z-direction along the path of the electron beam. This allows for more control of the distance between the grating and the surface, as well as sampling of multiple portions of the surface.

The surface mount is also horizontally loaded with the surface sitting on top. The surface is glued down with a small drop of Silver paste (using a precision pipet). A small link of silver paste is also used to connect the top of the surface (at its edge) to the mount for grounding purposes. Outside the vacuum system the mount is also supported by a two-dimensional translation stage (in the y-direction to control the height, and the x-direction to sample different portions of the surface). Also, the angular pitch is controlled

by a connected rotational stage. This rotational stage has the advantage of rotating in only one direction (not axial to the beamline as the previous wobble mount) and it is micrometer rotational control allowing for more precise adjustment of the pitch.

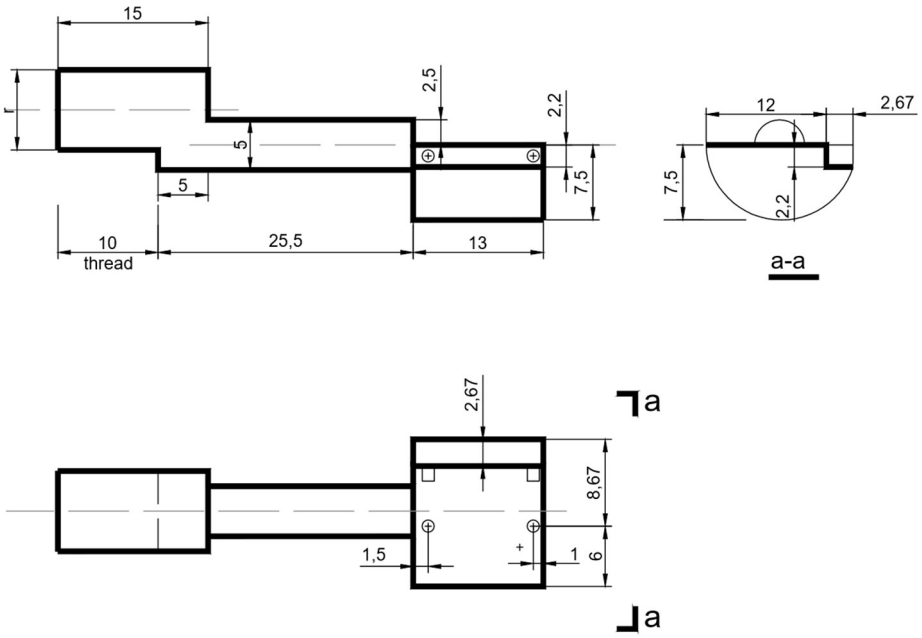

Figure D.4: Second Mount Schematic of Surface. Courtesy of Liyun Zhang. Labeled numbers are in millimeters

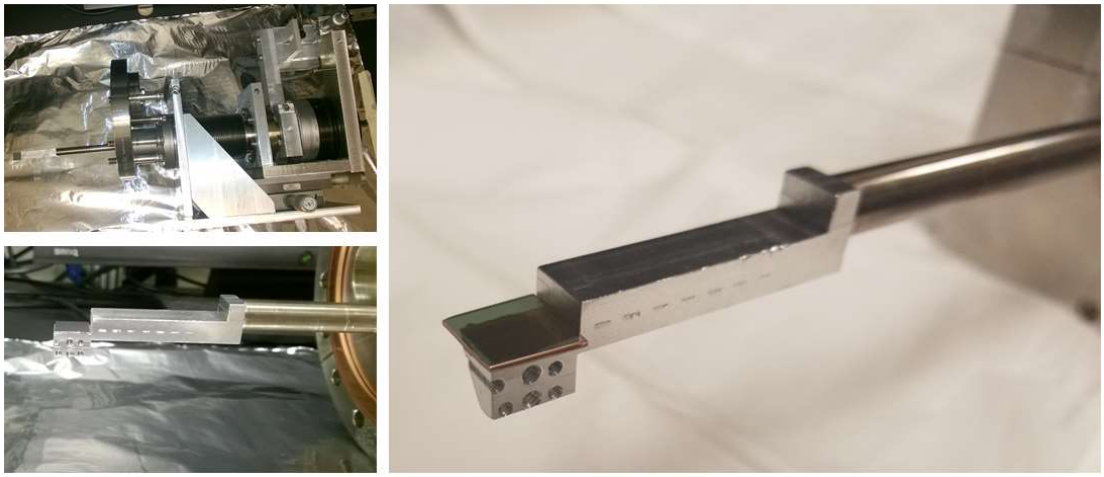

Figure D.5: Out of Chamber Images of Surface Mount. Top Left: Top view of combined image of surface mount, feedthrough, and translational/rotation stage. Bottom Left: Side view of surface mount. Right: Zoomed image of mount and surface.

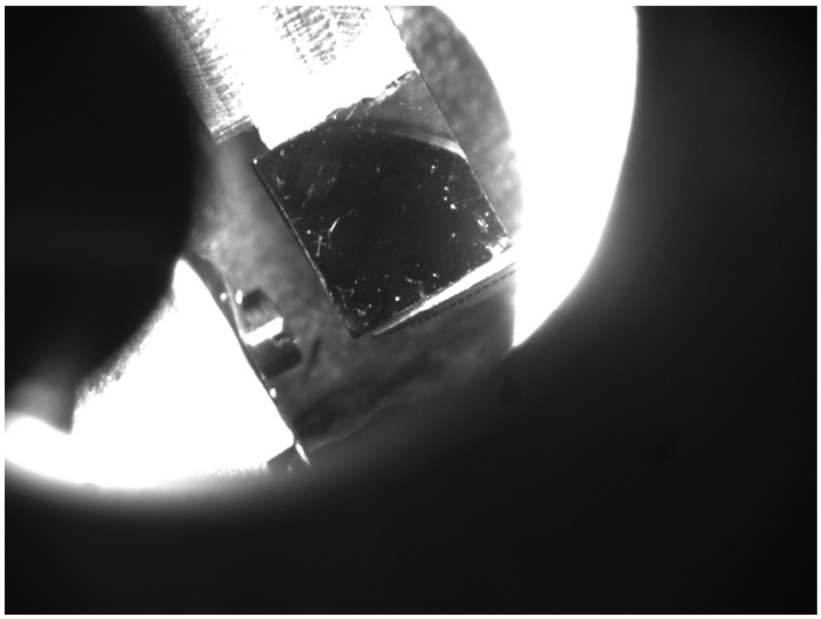

Figure D.6: Top View of Surface Mount and Grating Mount

Other advantages of this upgrade includes, being able to modify the surface or the grating outside of the vacuum system without exposing the other to the elements outside, and independent observation of the electron beams interaction with the grating, the surface, and combined.

D.3 Surface Preparation

Great care needs to be taken to ensure that the surface remains clean of dust or other contaminants. The effects of dust can be observed in the diffraction pattern and reduce contrast, lowering the sensitivity of our experiment. These dust particles tend to charge up and thus deflect the electron diffraction distribution, typically in the form of lensing (see Figure D.7).

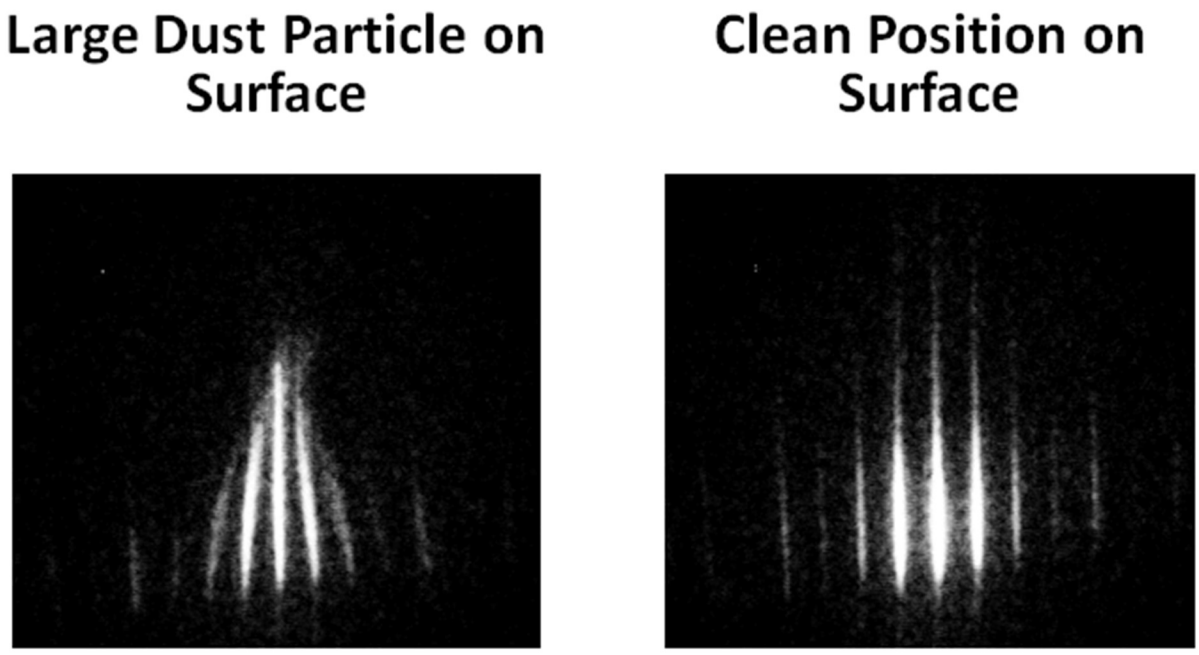

Figure D.7. Effects of Dust on Surface. Left: Diffraction pattern at the detector after interacting with a dust particle on the surface. In such circumstances, deflections (such as lensing) can occur. Right: Clean Position on the surface. No dust or contaminants present result in straight interference fringes in the vertical direction.

After cutting to size (1 cm^2) the surface with a boron carbide wafer cutter, the Si surface was cleaned using a version of the industry-standard RCA cleaning method (without the oxide strip), to remove dust or other contaminants [1]. Specifically, the cleaning procedure that was underwent is as follows:

i. Initial Prep: All glassware is initially cleaned with piranha solution (3:1 mixture of sulfuric acid and hydrogen peroxide) to eliminate any initial contaminants introduced during the cleaning process.
ii. Pre-RCA clean: The silicon surface is first sonicated for 30 minutes in acetone for large contaminants, followed by 30 min in isopropanol to clean off the acetone. The surface is then immediately blow-dried with nitrogen gas. The surface is then submerged in deionized water (and is kept in this state whenever any inaction during the cleaning is taking place)
iii. RCA clean:
    a. the first step is used to remove organics and particles. It involves submerging the surface in a solution of deionized water, 29% Ammonium Hydroxide, and 30% Hydrogen peroxide in a volume ratio of 5:1:1. These solutions were mixed and

measured using disposable glass pipets and each part was allowanced its own Pipet straw. The surface was held here for 10 minutes at approximately 80 degrees Celsius on a hotplate. Upon taking the surface out of the solution it was immediately rinsed with a squeeze bottle of deionized water and then submerged into deionized water

b. The second step, which usually involves removal of an oxide layer via hydrofluoric acid, was skipped.
c. The third step is used to remove ions. It involves transferring the surface from the deionized water where it was last left to a solution of deionized water, 37% hydrochloric acid, and 30% hydrogen peroxide in a volume ratio of 6:1:1 at approximately 80 degrees Celsius for 10 minutes.

iv. Post RCA clean: the surface was taken out with a tweezer and immediately rinsed with deionized water and blow-dried with nitrogen gas. It was then placed in a cleaned dry sample container for immediate transfer to installation into the vacuum chamber (taking roughly 45 minutes until the chamber was closed and began pumping down).

The uncoated gold mirror was immediately installed into the vacuum system upon unpackaging to minimize exposure. In both cases, contaminants were successfully eliminated.

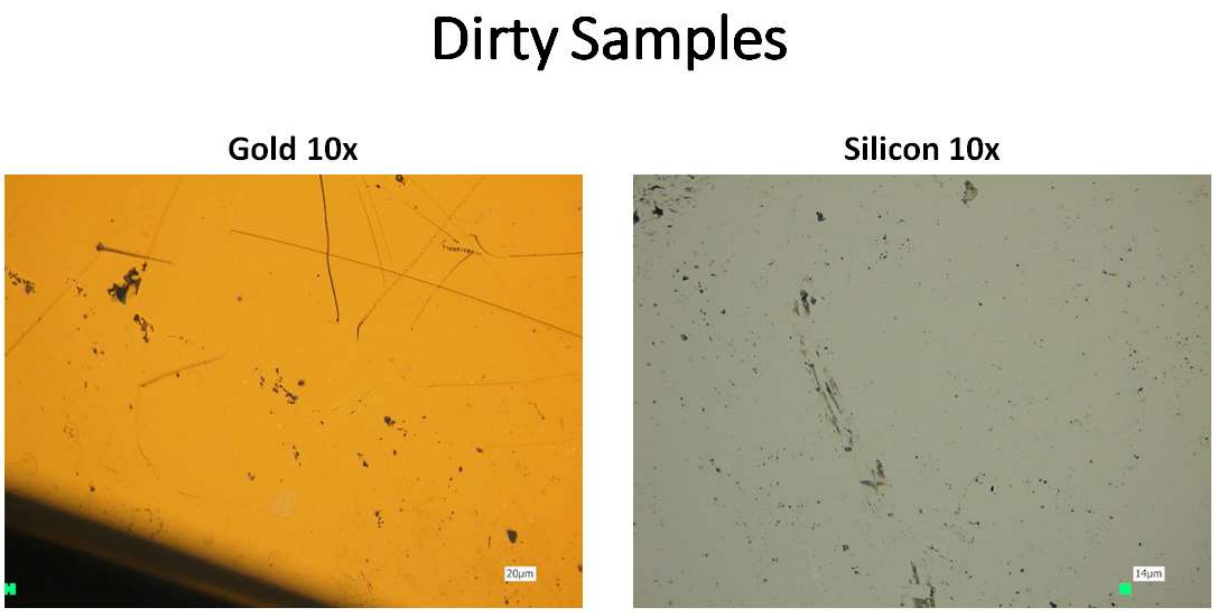

Figure D.8: Uncleaned Silicon and Gold Samples. Not appropriately cleaning the sample results in distortion of diffraction pattern as in Figure 4.9 left.

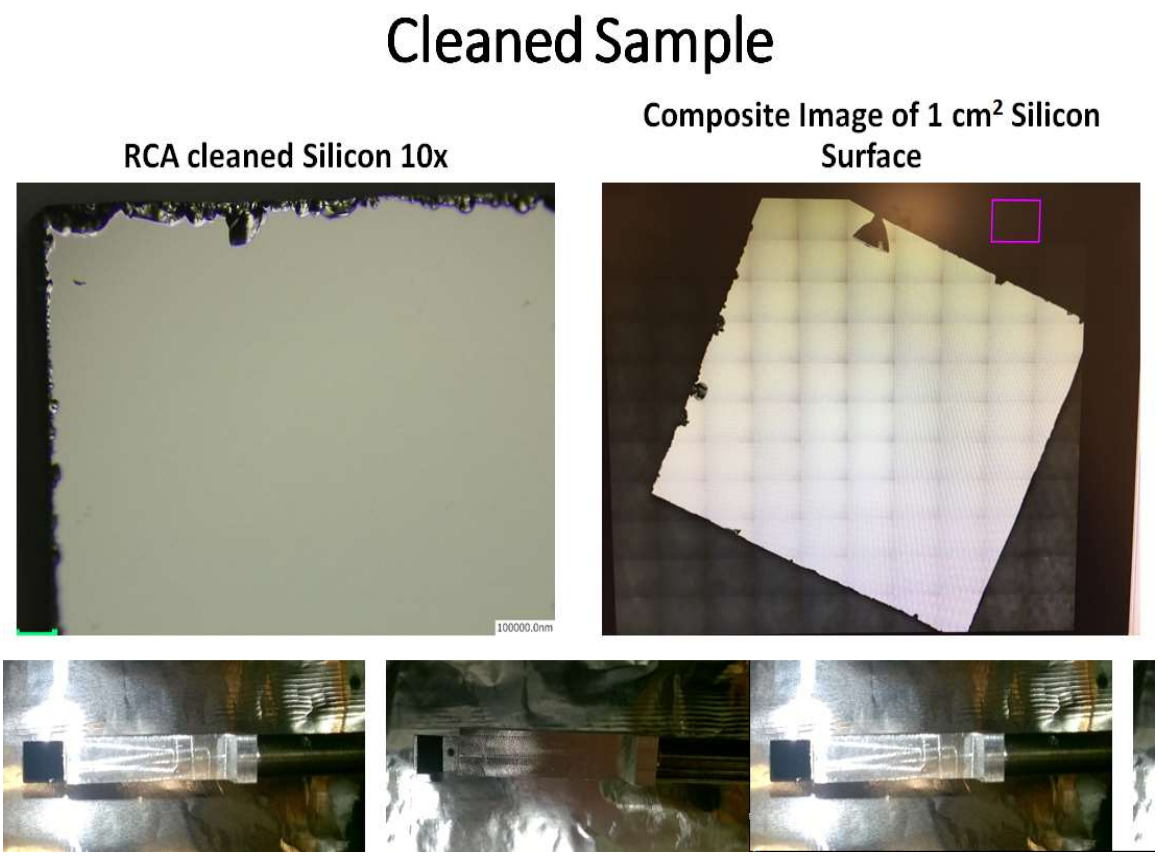

Figure D.9: Image of RCA cleaned Si Surfaces. Top Left: Optical Laser Microscope Image of 10x magnification of corner of Si Surface 1. Top Right Composite image of 1 cm^2 silicon surface. Bottom Left: Out of vacuum chamber image of Si surface 1. Bottom Right: Out of vacuum chamber image of Si surface 2.

D.4 Pitch Alignment

To measure the relative pitch of the surface, a small HeNe laser was mounted on top of the rotational stage. The laser would cast a spot on the wall with 1 mm square Cartesian graph paper 3.3 m away. As the stage rotated, the laser spot moved vertically on the wall. The precision of the laser spot vertically on the graph paper was approximately .5 mm. thus the angular pitch with respect to the beamline can be adjusted with a precision of approx. 0.2 mrad.

This pitch of the surface is adjusted to maximize the electron beam's deflection due to image charge attraction. In practice, what this means is the diffraction pattern was imaged at many pitch angles (near the optimum at steps of ≈ 0.2 mrad) all such that the beam current is 1/3 the original beam current (without the surface). Then the angle in which the image charge most deflects the beam is chosen as the one to be analyzed.

D.5 Effects of Lensing

As is elaborated upon in Chapter 5, a loss of contrast does not imply that decoherence has occurred, as dephasing is also possible cause. Here we will give attention to one form of dephasing: lensing, particularly a simple lens. Suppose a spatial charge distribution is formed on the surface such that the diffraction electron beams are broadened (or focused). This is a reasonable supposition, as the effects of dust particles or other contaminants appear to affect the diffraction pattern in this way (see figure 4.8). It should also be noted that the periodicity of the diffraction peaks also changed in this observation, which is what gave us the suspicion that lensing was occurring in the first place. Indeed, placing a convex lens after a grating can cause simultaneous focusing of the periodicity of the diffraction peaks and an increase in the width of the diffraction peaks.

In the experimental diffraction images for the case of silicon, (figure 4.3 and figure D.10) even after cleaning there is a noticeable amount of reduction in the diffraction peak to peak distance along with broadening. Therefore, it is critical to investigate whether a grating-simple lens model can simultaneously predict both observations in a quantifiable way.

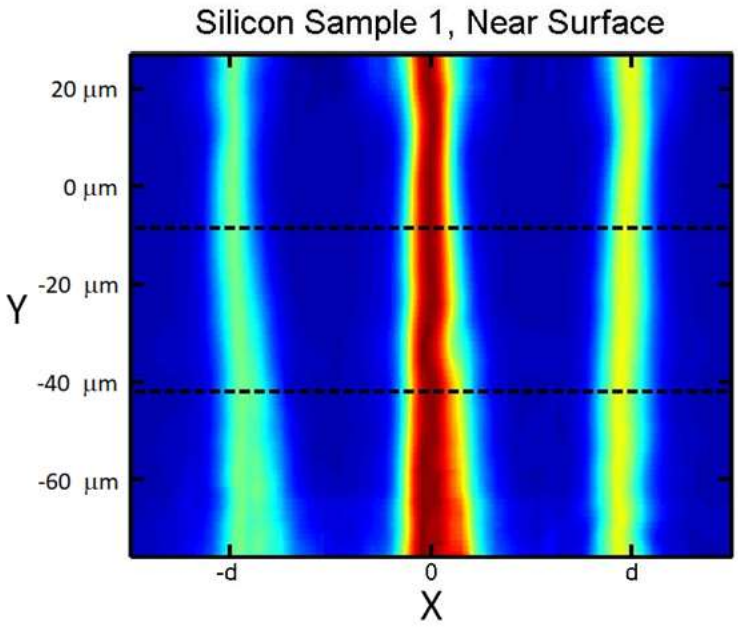

Figure D.10. Diffractogram Image for the case of Silicon

Sketched in Figure D.11 is a combined ray & beam diagram representing the effects of the diffraction beams undergoing diffraction from a grating (G) due to a lens (L). the lens and the grated are separated by a distance $z_3 = .3 \text{ cm} \rightarrow 1.3 \text{ cm}$. From the lens, the distance between the lens to the detection screen (Det.) is $z_4 = Z - z_3$, where $Z = 24 \text{ cm}$ is the distance from the grating to the detector. Without a lens, the observed diffraction peak periodicity is d and the diffraction peaks have a width of w. With a lens that has a focal length of f, the observed diffraction peak periodicity is d' and the diffraction peaks have a width of w'. Starting from a measured difference in peak periodicity from d to d' and treating the diffraction beams as rays, basic optics formulas will be used to to calculate the lens's focal length, and then treating the diffraction beams as Gaussian beams, beam optics formulas are used to calculate what the new diffraction peak's width w' would be.

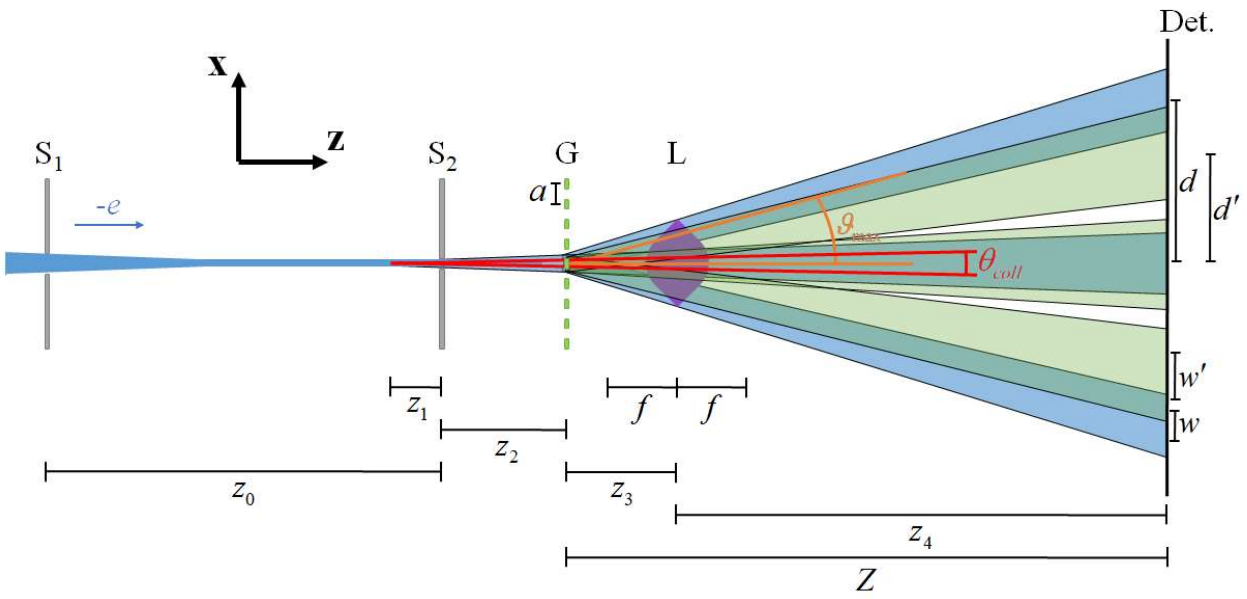

Figure D.11. Collimated Grating-Lens System. sketched is an electron beam (blue) collimated by two slits (S_1 and S_2) that undergoes diffraction through a grating (G) and further propagates to the detection screen (Det.). Without no lens (L) with focal length (*f*) present, the resulting diffraction pattern at the detection screen has a peak periodicity of *d* and a FWHM of *w*. When a lens is present, the diffraction beams (light green) alter in their divergence and land at new positions on the detector. This final pattern has a peak periodicity of *d'* and the peaks have a FWHM of *w'*.

The important basic optics formula to start with is based on the exit equation of a ray from a lens of focal length f. two central assumptions are made. The first assumption is that the position along the x-direction in which the ray enters the lens (define as x) is equal to the position in which the ray leaves the lens (this is the "thin lens approximation"). The second assumption is that the change in the slope $\Delta m = m_2 - m_1$ of a ray from entering the lens with slope m_1 to exiting the lens with slope m_2, is proportional to the distance away from the central axis of the lens (i.e. $\Delta m = ax$, where a is the proportionality constant).

From this, the first order diffraction ray starting at a central point at the grating will pass through the lens and will land on the detector at a height d' with respect to the central axis according to (equation 1.7 in [2]),

$$d' = \left[z_3 + z_4 \left(1 - \frac{z_3}{f} \right) \right] m_1 . \tag{D.1}$$

The entrance slope m_1 can be calculated from the diffraction angle using the diffraction equation,

$$m_1 = \tan\left(\theta_{coll}\right) \approx \sin\left(\theta_{coll}\right) = \lambda_{dB} / a , \tag{D.2}$$

where a is the periodicity of the grating. Rearranging equation D.2 in terms of the focal length yields

$$f = z_3 \left[1 - \frac{1}{z_4} \left(\frac{ad'}{\lambda_{dB}} - z_3 \right) \right]^{-1} . \tag{D.3}$$

Note that it can be seen through this equation that although a focal length can be either positive or negative, it can only have one solution for a given set of parameters.

It can now be inferred what kind of peak broadening would come for a particular focal length. The primary equations used are the beam optics lens equations 3.2-5 through 3.2-9a from [3]. After passing through the lens, the new divergence angle is proportional to the old divergence angle by

$$\theta_{coll}{}' = \theta_{coll} / M , \tag{D.4}$$

where the magnification is written in terms of the parameters $r^{beam} = z_{not}^{beam} / \left(z_{beam} - f \right)$ and $M_r^{beam} = \left| f / \left(z_{beam} - f \right) \right|$,

$$M = \frac{M_r^{beam}}{\sqrt{1 + r^2}} . \tag{D.5}$$

Here $z_{beam} \equiv z_1 + z_2 + z_3$ is the distance between the initial waist position (before the second slit) and the lens, and $z_{not}^{beam} = W_0 / \theta_{coll}$ is the initial depth of beam's focus (i.e. the

Raleigh length) with waist W_0. With an initial, unfocused divergence angle of $\theta_{coll} = 33\ \mu\text{rad}$, the distance from the second slit to the beam waist position is determined to be $z_1 = S_2 / \tan(\theta_{\text{coll}}) = 7.57\ \text{cm}$ and the Raleigh length would be $z_{not}^{beam} \simeq 2.5\ \mu\text{m}/33\ \mu\text{rad} = 7.57\ \text{m}$. That $z_{not}^{beam} \simeq z_1$ is not particularly surprising in light of Figure 3.1-4 of [3].

Once the new collimation angle is established, the final beam's width can be calculated trigonometrically back from the new waist location due to the lens. This waist location can be found using the equation,

$$z'_{beam} = \left(M^2\left(z_{beam} - f\right)\right) + f\ . \tag{D.6}$$

And then the final beam's width can be computed as

$$w'(d', z_3) = \tan\left(\theta_{coll}{}'\right)\left(z_4 + f - \left(M^2\left(z_{beam} - f\right)\right)\right). \tag{D.7}$$

As a reminder, f is expressed in terms of d' and z_3.

Plotted in Figure D.12 are curves of the final FWHM (w') vs peak periodicity (d'), each curve corresponding to a lens with a particular distance to the grating z_3, and are parameterized by their focal length f. This is compared with the experimentally observed values extracted from the horizontal lineouts for different vertical positions (from figure D.10). The intersection of these curves along with the data corresponds to the width and periodicity with no lens present ($d = 72\ \mu\text{m}$ and $w = 13\ \mu\text{m}$).

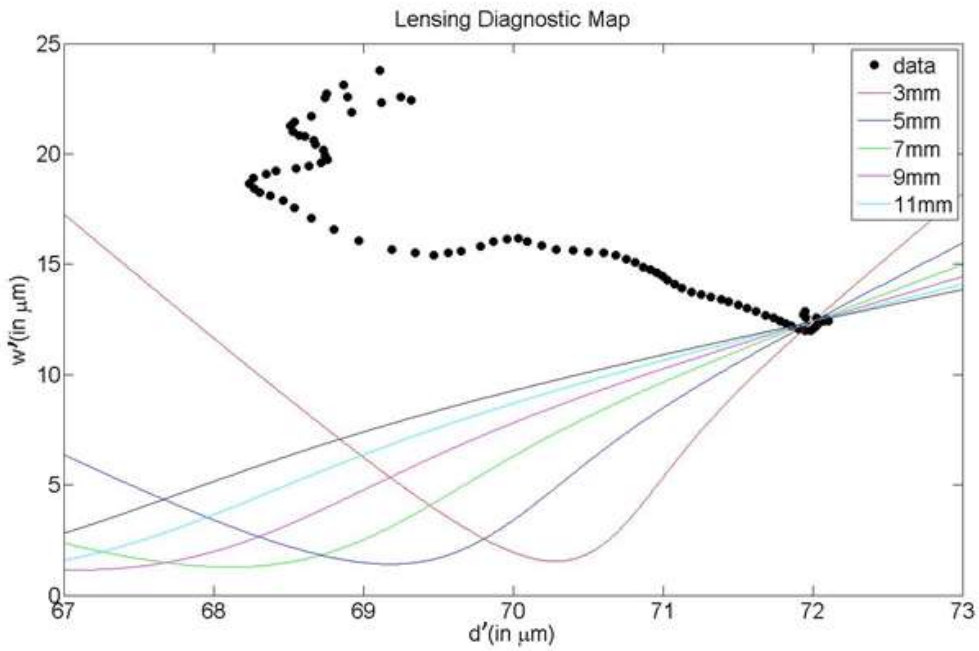

Figure D.12. Lensing Diagnostic Map

If one assumes that different heights to the surface corresponds to different focusing powers (or different focal lengths) it is clear that the a simple lens model cannot explain the behavior of the loss of contrast and change in peak periodicity, because the general trend is that there would be a focusing of the peak's widths before a defocusing as the peak to peak distance comes together, as well as these curves generally do not fall in the same general area on this parameter space as the experimental data. It is still possible that the physical reality is well explained by a more complex lens model (such as a multiple lens model). Nevertheless, we can exclude a simple single lens model to explain the general loss of contrast.

D.6 Appendix D Bibliography

[1] M. Itano, F. W. Kern, M. Miyashita, and T. Ohmi, IEEE Trans. Semicond. Manuf. **6**, 258 (1993).
[2] H. Liebl, in *Appl. Charg. Part. Opt.* (Springer, Berlin, Heidelberg, 2008), pp. 1–3.
[3] B. E. A. Saleh and M. C. Teich, in *Fundam. Photonics*, 2nd ed. (Wiley, 2007), pp. 74–101.

APPENDIX E

PROGRAMS FOR DEPHASING VS DECOHERENCE

The following contains the computations and program while investigating the similarities and differences between dephasing and decoherence as described in Chapter 5.

E.1 Decoherence and Dephasing Path Integral Program Version 1

This is the original version of the decoherence and dephasing path integral code. Stated parameters are based on the experimental setup described in Chapter 4: Decoherence Experiment, where two collimating slits are used, a decoherer is set up separate from the grating. The decoherer contains incoherent summing of random blocks with phase jumps ranging matching that of Section D.1 . The random phase includes random blocks of phase jumps ranging from 0 to $2\pi c$, where $0 \leq c \leq 1$ acts as the variable that changes the contrast. See Chapter 5 section 2 for more details.

The following table (D.1) outlines the set of parameters used in the path integral to test the results of the effects of dephasing/decoherence.

Plane Name	Width	# of Grid-Points	Section Between	Azimuthal Length
Slit 1	12.7 μm	24	Slit 1 & Slit 2	L_1 = 24 cm
Slit 2	2.5 μm	1500	Slit 2 & Dephaser / Decoherer	L_2 = 5.5 cm
Dephaser / Decoherer	20 μm	5000	Dephaser / Decoherer & Grating	L_3 = .5 cm
Grating	15 μm	12000	Grating & Near Field	L_4 = .5 cm
Near Field	20 μm	1500	Near Field & Far Field	L_5 = 24 cm
Far Field	60 μm	1500		

Table E.1: Parameters for Path Integral Simulation Setup 1

```
      PROGRAM Path_Integral_Decoherence
      !Author:Peter Beierle
      !Last Revised 10/13/2015
      implicit none

      integer Nf,Nl, Ng,Ns,Nscr,Ngg,Nd,endrun
      parameter(Nf=24,Nl=1500,Ng=12000,Ns=1500,Nscr=1500,Nd=5000)
      parameter(Ngg=5000,endrun=100)
      integer j,i,inc,k,m,numberofslits,nn,i2,incend,jend
      integer(2) ihr,imin,isec,i100th,starttime,endtime
      real programtime
      real*8 hbar,h,c,position,d,norm,me,Pi,gam,ee ,D0
      real*8 lambdae,sourcesize,dsource,ve,D2,D3,phaseslope,phasestep
      real*8 Dsg,Dgg, gndglsssize,dgndglss
      real*8 Energy,kernel,screen3size,dscreen3,lenssize,dlens
      real*8 dist1,dist2,lngh,image(Ngg)
      complex*8 kern3(Nl,Ng),kern2(Ng,Ngg),kern(Ngg,Ns),kern0(Ns,1)
      complex*8 kern4(Nscr,Nl)
      complex*8 phiin(1),phi2(Nl),amp(Ng),phi0(Ns)
      complex*8 phigndglss(Ngg)
      complex*8 cmI,phioutx,imagephase(Ngg)
      complex*8 phiscreen(Nscr),phil(Ng)
      real*8 probscreen(Nscr),probscreenmem(Nscr,Nf)
      real*8 probscreenaverage(Nscr)
      real*8 probgrating(Ng),probgratingmem(Ng,Nf)
      real*8 probgratingaverage(Ng),finalprob2(Nl,endrun)
      real*8 x,prob2(Nl)
      real*8 dsecondslit,secondslit
      real*8 phasecurveglobal
      real*8 dfirstslit,firstslit
      complex*8 matrix(Nscr,Nscr),finalmatrix(Nscr,Nscr)
      complex*8 matrix2(Nl,Nl),finalmatrix2(Nl,Nl)
      complex*8 matrix3(Ns,Ns),finalmatrix3(Ns,Ns)
      real*8 sigma2(1000),acumphase(1000),positionphase,returnvalue
      integer runner,runnumber,idum,nnmodnumofpnts

      !for random phase
      integer Q,p,numofpoints
      parameter(Q=123456)
      real*8 dephasingsize,value(1000),sigma(1000)
      !call rnset(Q)

      dephasingsize=0.7
      runnumber=1


      !do i=1,Nscr
      ! do j=1,Nscr
      !    finalmatrix(i,j)=0d0
      ! enddo
      !enddo

      !do i=1,Nl
      ! do j=1,Nl
      !    finalmatrix2(i,j)=0d0
      ! enddo
      !enddo

      !do i=1,Ns
      ! do j=1,Ns
      !    finalmatrix3(i,j)=0d0
      ! enddo
      !enddo

      open(unit=17,file='phase.dat') !phase written
```

```
66
67
68
69       ee=1.6d-19
70       Pi=3.14159265d0
71       h=6.626d-34
72       hbar=h/2d0/Pi
73       c=3.d8
74       d=100.d-9
75       me=9.11d-31
76       Energy=1670d0*ee+me*c*c
77       !ve=dsqrt(2.*Energy/(.511E6/c**2.))  !non relativistic leftover
78       gam=Energy/(me*c*c)
79       ve=sqrt(1d0-1d0/(gam*gam))*c
80       lambdae=h/(gam*me*ve)
81       write(6,*) "The gamma factor is  ",gam
82       write(6,*) "The electron velocity is  ",ve
83
84       firstslit=12.7d-6
85       write(6,*) "The first slit is ",firstslit, " wide."
86       secondslit=2.5d-6 !10.22E-6 !14.59E-6
87       write(6,*) "The second slit is  ",secondslit, "   wide."
88       lenssize=20000d-9
89       gndglsssize=20d-6
90       write(6,*) "the ground glass size is ",gndglsssize," wide"
91       write(6,*) "The lenssize is  ",lenssize, "   wide."
92       sourcesize=15000d-9          !YOU NEED TO CHANGE THE NUMBER OF SLITS!
93       write(6,*) "The grating source is  ",sourcesize, "   wide."
94       screen3size=500d-6
95       write(6,*) "The detecttion screen is  ",screen3size, "   wide."
96
97       dfirstslit=firstslit/Nf
98       dsecondslit=secondslit/Ns
99       dlens=lenssize/Nl
100      dsource=sourcesize/Ng
101      dscreen3=screen3size/Nscr
102      dgndglss=gndglsssize/Ngg
103      D0=.24d0        !distance from the first slit to the second slit
104      Dsg=.06d0-.005d0        !0.3
105      write(6,*) "The dis from the 2nd slit to ground glass is ",Dsg
106      Dgg= .005d0
107
108      D2=.005d0   !distance at UNL
109      write(6,*) "The distance from the grating to the lens is  ",D2
110
111      D3=0.24d0   !distance at UNL   0.231
112      write(6,*) "The distance from the lens to the screen is  ",D3
113
114
115      !call gettim(ihr,imin,isec,i100th)
116      !starttime=0.
117      !starttime=ihr*60*60+imin*60+isec
118
119      cmI=(0,1)   !the complex number I
120
121
122
123
124  !              The next subroutines sets up the path integral kernels
125     write(6,*) "01"
126     call
         firstkernalsubroutine(kern,Ngg,Ns,gndglsssize,dgndglss,secondslit,dsecondslit,D
         sg,lambdae)
127     call
         secondkernalsubroutine(kern2,Ng,Ngg,sourcesize,dsource,gndglsssize,dgndglss,Dgg
```

```
      ,lambdae)
128   call
      thirdkernalsubroutine(kern3,Nl,Ng,lenssize,dlens,sourcesize,dsource,D2,lambdae)
129   call
      fourthkernalsubroutine(kern4,Nscr,Nl,screen3size,dscreen3,lenssize,dlens,D3,lam
      bdae)
130
131
132   !              The next subroutine sets up the amplitude at the grating
133      call gratingamplitude (amp,sourcesize,dsource,Ng)
134
135
136
137
138      do runnumber=1,endrun
139      do i=1,Nl
140      finalprob2(i,runnumber)=0d0
141      enddo
142      enddo
143
144      do runnumber=1,endrun
145
146      do i=1,200 !value is an array of 100 random numbers from 0 to 1
147      call ran0(idum,returnvalue)
148       value(i)=returnvalue
149      enddo
150
151
152      numofpoints=25 !should be a factor of 5000
153      p=1
154
155      do k=1,Ngg
156       nn=k
157       image(nn)=dephasingsize*2.0*Pi*value(p)
158       imagephase(nn)=cmI*dsin(image(nn))+dcos(image(nn))
159       nnmodnumofpnts=MOD(nn,numofpoints)
160        if (nnmodnumofpnts.eq.0) then
161         p=p+1
162        endif
163      enddo
164
165
166
167
168      do k=1,Ngg
169         nn=k
170       x=(gndglsssize/2d0-dgndglss*nn)
171       write(17,*) x, image(k)
172      enddo
173
174
175
176
177          ! ---------start of large incoherent loop-----------------------------
178      incend=Nf
179
180      do inc=1,incend
181
182      call
      zerothkernel(kern0,Ns,firstslit,dfirstslit,inc,secondslit,dsecondslit,D0,lambd
      ae)
183
184      !This routine propogates from the first slit to the second slit
185      call wavefunction1(Ns,kern0,phi0)
186
```

```
187       !The next routine propagates from the source (second slit) to the gndglss
188       call
          wavefunction2(phi0,Ngg,Ns,kern,phigndglss,imagephase)

189
190       !The next routine propagates from the gndglss to the grating
191       call wavefunction3(phigndglss,Ng,Ngg,kern2,phi1,amp)
192
193       !The next routine propagates from the grating to the "lens" (just the
          nearfield)
194       call
          wavefunction4(phi1,N1,Ng,kern3,phi2)
195
196       !The next routine propagates from the lens to the screen
197       call wavefunction5(phi2,Nscr,N1,kern4,phiscreen)
198
199
200       !the probability is calculated  next
201       call
          probabilitycalculation(phiscreen,probscreen,probscreenmem,prob2,finalprob2,Nsc
          r,N1,runnumber,inc,incend,endrun,phi2)
202
203
204   !    calculation of matrices
205   !   call
      matrixcalculation(phiscreen,phi2,phi0,matrix,finalmatrix,matrix2,finalmatrix2,matr
      ix3,finalmatrix3,Nscr,N1,Ns)
206
207         enddo   !close incoherent loop
208   write(6,*) 'end of runnumber',runnumber
209    enddo !end of runnumber loop
210
211
212
213   !     the incoherent sum at the screen is calculated next
214   call screenincoherentsumming(Nscr,incend,inc,probscreenaverage,probscreenmem)
215
216
217
218
219   !     next write data to file
220   call
      writeprobabilities(Nscr,N1,screen3size,dscreen3,probscreenaverage,runnumber,endrun
      ,finalprob2)
221
222   !call writematrices(Nscr,N1,Ns,finalmatrix,finalmatrix2,finalmatrix3)
223
224
225   !       next find runtime
226
227   !   call gettim(ihr,imin,isec,i100th)
228   !   endtime=0.
229   !   endtime=ihr*60*60+imin*60+isec
230
231   !   programtime=(endtime-starttime)/60.
232   !   write(6,*) 'Program run time =  ',programtime
233
234       format(E12.6,X,E12.6) !there was a 666 there
235         format(E12.6,X,E12.6)
236
237       END
238
239
240   !----------------------------------------------------------------------
241   subroutine
```

```
      firstkernalsubroutine(kern,Ngg,Ns,gndglsssize,dgndglss,secondslit,dsecondslit,Dsg,
      lambdae)
242   implicit none
243     integer i,j,Ngg,Ns
244     real*8 gndglsssize,dgndglss,secondslit,dsecondslit,Dsg,lambdae
245     real*8 dist1,dist2,lngh,kernel,Pi
246     complex*8 cmI,kern(Ngg,Ns)
247     Pi=3.14159265d0
248     cmI=(0,1) !the complex number I
249
250       do i=1,Ngg
251           do j=1,Ns
252           dist1=(gndglsssize/2d0-dgndglss*i)   !source stand for grating
253           dist2=(secondslit/2d0-dsecondslit*j)
254           lngh=sqrt((dist1-dist2)**2d0+(Dsg)**2d0)
255           kernel=0d0
256           kernel=2d0*Pi*lngh/(lambdae)
257           kern(i,j)=cmI*dsin(kernel)+dcos(kernel)
258           enddo
259       enddo
260       write(6,*) "02"
261   return
262   end
263
264   !---------------------------------------------------------------------
265    subroutine
       secondkernalsubroutine(kern2,Ng,Ngg,sourcesize,dsource,gndglsssize,dgndglss,Dgg,l
       ambdae)
266    implicit none
267    integer i,j,Ng,Ngg
268    real*8 sourcesize,dsource,gndglsssize,dgndglss,Dgg,lambdae
269    real*8 dist1,dist2,lngh,kernel,Pi
270    complex*8 cmI,kern2(Ng,Ngg)
271    Pi=3.14159265d0
272    cmI=(0,1)  !the complex number I
273
274       do i=1,Ng
275           do j=1,Ngg
276           dist1=(sourcesize/2d0-dsource*i)
277           dist2=(gndglsssize/2d0-dgndglss*j)
278           lngh=sqrt((dist1-dist2)**2d0+(Dgg)**2d0)
279           kernel=0d0
280           kernel=2d0*Pi*lngh/(lambdae)
281           kern2(i,j)=cmI*dsin(kernel)+dcos(kernel)
282           enddo
283         enddo
284       write(6,*) "03"
285   return
286   end
287   !---------------------------------------------------------------------
288   subroutine
      thirdkernalsubroutine(kern3,Nl,Ng,lenssize,dlens,sourcesize,dsource,D2,lambdae)
289   implicit none
290   integer i,j,Nl,Ng
291   real*8 lenssize,dlens,sourcesize,dsource,D2,lambdae
292   real*8 dist1,dist2,lngh,kernel,Pi
293   complex*8 cmI,kern3(Nl,Ng)
294   Pi=3.14159265d0
295   cmI=(0,1)   !the complex number I
296
297       do i=1,Nl
298           do j=1,Ng
299           dist1=(lenssize/2d0-dlens*i)
300           dist2=(sourcesize/2d0-dsource*j)
301           lngh=sqrt((dist1-dist2)**2d0+(D2)**2d0)
```

```
302           kernel=0
303           kernel=2d0*Pi*lngh/(lambdae)
304           kern3(i,j)=cmI*dsin(kernel)+dcos(kernel)
305           enddo
306       enddo
307       write(6,*) "04"
308   return
309   end
310   !---------------------------------------------------------------------
311   subroutine
      fourthkernalsubroutine(kern4,Nscr,Nl,screen3size,dscreen3,lenssize,dlens,D3,lambda
      e)
312   implicit none
313   integer i,j,Nscr,Nl
314   real*8 screen3size,dscreen3,lenssize,dlens,D3,lambdae
315   real*8 dist1,dist2,lngh,kernel,Pi
316   complex*8 cmI,kern4(Nscr,Nl)
317   Pi=3.14159265d0
318   cmI=(0,1)   !the complex number I
319
320       do i=1,Nscr
321           do j=1,Nl
322           dist1=(screen3size/2d0-dscreen3*i)
323           dist2=(lenssize/2d0-dlens*j)
324           lngh=sqrt((dist1-dist2)**2d0+(D3)**2d0)
325           kernel=0d0
326           kernel=2d0*Pi*lngh/(lambdae)
327           kern4(i,j)=cmI*dsin(kernel)+dcos(kernel)
328           enddo
329       enddo
330       write(6,*) "05"
331   return
332   end
333
334   !---------------------------------------------------------------------
335
336
337   subroutine gratingamplitude (amp,sourcesize,dsource,Ng)
338   implicit none
339   integer numberofslits,jend,k,m,j,nn,Ng
340   real*8 x,sourcesize,dsource
341   complex*8 amp(Ng)
342       numberofslits=150  !sourcesize/d
343       jend=int(Ng/numberofslits/2)
344       do k=0,numberofslits-1
345       do m=0,1
346       do j=1,jend
347           nn=j+k*jend*2+m*jend
348           x=(sourcesize/2d0-dsource*nn)
349           amp(nn)=m  !1. for pure phasegrating
350       enddo
351       enddo
352       enddo
353   return
354   end
355
356   !---------------------------------------------------------------------
357
358   subroutine
      zerothkernel(kern0,Ns,firstslit,dfirstslit,inc,secondslit,dsecondslit,D0,lambdae)

359   implicit none
360   integer Ns,i,j,inc
361   real*8 firstslit,dfirstslit,secondslit,dsecondslit,D0,lambdae
362   real*8 kernel,dist1,dist2,lngh,Pi
```

```
363   complex*8 cmI,kern0(Ns,1)
364   Pi=3.14159265d0
365   cmI=(0,1)   !the complex number I
366       do i=1,Ns
367           do j=1,1
368           dist1=(firstslit/2d0-dfirstslit*inc)
369           dist2=(secondslit/2d0-dsecondslit*i)
370           lngh=sqrt((dist1-dist2)**2d0+(D0)**2d0)
371           kernel=0d0
372           kernel=2d0*Pi*lngh/(lambdae)
373           kern0(i,j)=cmI*dsin(kernel)+dcos(kernel)
374           enddo
375       enddo
376   return
377   end
378   !---------------------------------------------------------------------
379
380   subroutine wavefunction1(Ns,kern0,phi0)
381   integer Ns,i,i2
382   complex*8 phiin(1),phi0(Ns),phioutx
383   complex*8 kern0(Ns,1)
384       phiin(:)=1d0
385       phi0(:)=0d0
386       write(6,*) "1"
387       do i=1,Ns
388           phioutx=0d0
389           do i2=1,1
390           phioutx=phioutx+kern0(i,i2)*phiin(i2)
391           enddo
392       phi0(i)=phioutx
393       enddo
394   return
395   end
396
397   !---------------------------------------------------------------------
398
399   subroutine
      wavefunction2(phi0,Ngg,Ns,kern,phigndglss,imagephase)

400   implicit none
401   integer Ngg,Ns,i,i2
402   complex*8 phi0(Ns),phigndglss(Ngg),phioutx
403   complex*8 kern(Ngg,Ns),imagephase(Ngg)
404       phigndglss(:)=0d0
405       write(6,*) "2"
406       do i=1,Ngg
407           phioutx=0d0
408           do i2=1,Ns
409           phioutx=phioutx+kern(i,i2)*phi0(i2)
410           enddo
411       phigndglss(i)=phioutx*imagephase(i)
412   !   write(6,*) phi1(i)
413         enddo
414   return
415   end
416
417   !---------------------------------------------------------------------
418   subroutine wavefunction3(phigndglss,Ng,Ngg,kern2,phi1,amp)
419   implicit none
420   integer Ngg,Ng,i,i2
421   complex*8 phigndglss(Ngg),phioutx,phi1(Ng)
422   complex*8 kern2(Ng,Ngg),amp(Ng)
423
424       phi1(:)=0d0
425       do i=1,Ng
```

```
426           phioutx=0d0
427           do i2=1,Ngg
428           phioutx=phioutx+kern2(i,i2)*phigndglss(i2)
429           enddo
430         phi1(i)=phioutx*amp(i)
431
432       enddo
433   return
434   end
435
436
437   !---------------------------------------------------------------------
438
439   subroutine wavefunction4(phi1,Nl,Ng,kern3,phi2)
440   implicit none
441   integer Nl,Ng,i,j
442   complex*8 phi2(Nl),phioutx,kern3(Nl,Ng),phi1(Ng)
443       write(6,*) "2"
444       phi2(:)=0d0
445       do i=1,Nl
446       phioutx=0d0
447           do j=1,Ng
448           phioutx=phioutx+kern3(i,j)*phi1(j)
449           enddo
450       phi2(i)=phioutx
451   !   write(6,*) phi2(i)
452       enddo
453      write(6,*) "3"
454   return
455   end
456
457   !---------------------------------------------------------------------
458
459   subroutine wavefunction5(phi2,Nscr,Nl,kern4,phiscreen)
460   implicit none
461   integer Nscr,Nl,i,j
462   complex*8 phiscreen(Nscr),phi2(Nl),kern4(Nscr,Nl),phioutx
463
464
465       phiscreen(:)=0d0
466       do i=1,Nscr
467       phioutx=0d0
468           do j=1,Nl
469           phioutx=phioutx+kern4(i,j)*phi2(j)
470           enddo
471       phiscreen(i)=phioutx
472       enddo
473
474         write(6,*) "4"
475   return
476   end
477
478
479   !---------------------------------------------------------------------
480
481   subroutine
      probabilitycalculation(phiscreen,probscreen,probscreenmem,prob2,finalprob2,Nscr,Nl
      ,runnumber,inc,incend,endrun,phi2)
482   implicit none
483   integer i,Nscr,inc,incend,runnumber,Nl,endrun
484   real*8 probscreen(Nscr),norm,probscreenmem(Nscr,incend)
485   complex*8 phiscreen(Nscr),phi2(Nl)
486   real*8 prob2(Nl),finalprob2(Nl,endrun)
487
488
```

```
489       do i=1,Nscr
490           probscreen(i)=phiscreen(i)*conjg(phiscreen(i))
491       enddo
492
493       norm=0
494       do i=1,Nscr
495           norm=norm+probscreen(i)
496       enddo
497
498       write(6,*) "before"
499
500       do i=1,Nscr
501           probscreen(i)=1/norm*phiscreen(i)*conjg(phiscreen(i))
502       enddo
503
504       write(6,*) "after"
505
506       do i=1,Nscr
507       probscreenmem(i,inc)=probscreen(i)
508       enddo
509
510       !not normalized correctly
511       prob2(:)=0d0
512       do i=1,Nl
513         prob2(i)=phi2(i)*conjg(phi2(i))
514       enddo
515
516
517       do i=1,Nl
518        finalprob2(i,runnumber)=finalprob2(i,runnumber)+prob2(i)
519       enddo
520
521     return
522     end
523
524     !---------------------------------------------------------------
525
526     subroutine
      matrixcalculation(phiscreen,phi2,phi0,matrix,finalmatrix,matrix2,finalmatrix2,matr
      ix3,finalmatrix3,Nscr,Nl,Ns)
527     implicit none
528
529     integer i,j,Nscr,Nl,Ns
530     complex*8 phiscreen(Nscr),matrix(Nscr,Nscr),finalmatrix(Nscr,Nscr)
531     complex*8 phi2(Nl),matrix2(Nl,Nl),finalmatrix2(Nl,Nl)
532     complex*8 phi0(Ns),matrix3(Ns,Ns),finalmatrix3(Ns,Ns)
533
534       do i=1,Nscr
535        do j=1,Nscr
536          matrix(i,j)=phiscreen(i)*conjg(phiscreen(j))
537        enddo
538       enddo
539
540       do i=1,Nscr
541         do j=1,Nscr
542          finalmatrix(i,j)=finalmatrix(i,j)+matrix(i,j)
543         enddo
544       enddo
545
546       do i=1,Nl
547        do j=1,Nl
548          matrix2(i,j)=phi2(i)*conjg(phi2(j))
549        enddo
550       enddo
551
```

```
552       do i=1,Nl
553        do j=1,Nl
554          finalmatrix2(i,j)=finalmatrix2(i,j)+matrix2(i,j)
555        enddo
556       enddo
557
558       do i=1,Ns
559        do j=1,Ns
560           matrix3(i,j)=phi0(i)*conjg(phi0(j))
561        enddo
562       enddo
563
564       do i=1,Ns
565        do j=1,Ns
566           finalmatrix3(i,j)=finalmatrix3(i,j)+matrix3(i,j)
567        enddo
568       enddo
569   return
570   end
571
572    !---------------------------------------------------------------
573
574   subroutine
      screenincoherentsumming(Nscr,incend,inc,probscreenaverage,probscreenmem)
575   implicit none
576
577   integer i,inc,incend,Nscr
578   real*8 probscreenaverage(Nscr),probscreenmem(Nscr,incend)
579
580         probscreenaverage(:)=0d0
581         do i=1,Nscr
582         do inc=1,incend
583           probscreenaverage(i)=probscreenaverage(i)+probscreenmem(i,inc)
584         enddo
585         enddo
586   return
587   end
588
589    !---------------------------------------------------------------
590
591   subroutine
      writeprobabilities(Nscr,Nl,screen3size,dscreen3,probscreenaverage,runnumber,endrun
      ,finalprob2)
592   implicit none
593   integer i,j,Nscr,Nl,runnumber,endrun
594   real*8 position,screen3size,dscreen3
595   real*8 probscreenaverage(Nscr),finalprob2(Nl,endrun)
596
597       open(unit=12,file='PX.dat')     !probability distribution
598       open(unit=13,file='probabilitydephasor.dat') !density matrix at dephasor
599
600
601       do j=1,Nscr
602           position=(screen3size/2d0-dscreen3*j)
603           write(12,666) position,probscreenaverage(j)
604       enddo
605
606       do runnumber=1,endrun
607        do i=1,Nl
608         write(13,*) runnumber, finalprob2(i,runnumber)
609        enddo
610       enddo
611
612       format(E12.6,X,E12.6) !there was a 666 there
613
```

```
614   return
615   end
616
617    !---------------------------------------------------------------
618
619   subroutine writematrices(Nscr,Nl,Ns,finalmatrix,finalmatrix2,finalmatrix3)
620   implicit none
621
622   integer i,j,Nscr,Nl,Ns
623   complex*8 finalmatrix(Nscr,Nscr),finalmatrix2(Nl,Nl),finalmatrix3(Ns,Ns)
624
625       open(unit=14,file='matrixscreen.dat') !density matrix at screen
626       open(unit=15,file='matrixlens.dat') !density matrix at lens
627       open(unit=16,file='matrixsecondslit.dat')!density matrix at 2nd slit
628
629
630       do i=1,Nscr
631        do j=1,Nscr
632           write(14,667) finalmatrix(i,j)
633        enddo
634       enddo
635
636
637       do i=1,Nl
638        do j=1,Nl
639           write(15,667) finalmatrix2(i,j)
640        enddo
641       enddo
642
643       do i=1,Ns
644        do j=1,Ns
645           write(16,667) finalmatrix3(i,j)
646        enddo
647       enddo
648
649          format(E12.6,X,E12.6)
650
651    return
652    end
653
654
655    !---------------------------------------------------------------
656
657
658   subroutine bubblepot(acumphase,sigma2,idum,runnumber)
659   !use msimsl
660   implicit none
661   !this program has its own random number generator to avoid
662   !using msimsl
663   integer i,endset,j,runnumber
664   parameter(endset=40000)
665   real*8 devmean,devdeviation,x1(endset),x2(endset),sigma(endset),z1(1000),z2(1000)
666   real*8 ampmean,ampdeviation,y1(endset),y2(endset),amplitude(endset)
667   real*8 Pi,phase(1000),velocity,hbar,totallength(1000),sigma2(1000)
668   integer idum
669   real*8 value
670   real*8 acumphase(1000)
671   !integer Q !,p,numofpoints(1000)
672   !parameter(Q=987654)
673   !real*8 dephasingsize,value(1000)
674   !call rnset(Q)
675
676   open(unit=11,file='phasetest.dat') !testingthephase
677
678
```

```
679
680
681   do i=1,1000
682   acumphase(i)=0.0
683   enddo
684
685   !open(unit=12,file='random.dat')
686   hbar=6.582e-16
687   Pi=3.14159265
688   velocity=2.422E7
689
690
691   devmean=250.E-9
692   devdeviation=250.E-9
693   ampdeviation=.35
694   ampmean=0.
695
696   do i=1,1000
697     phase(i)=0.0
698     totallength(i)=0.0
699   enddo
700
701   !using the box-mueller method
702   do j=1,1000
703   do i=1,endset
704      call ran0(idum,value)
705      x1(i)=value
706      call ran0(idum,value)
707      x2(i)=value
708      call ran0(idum,value)
709      y1(i)=value
710      call ran0(idum,value)
711      y2(i)=value
712
         sigma(i)=(devdeviation/(2*sqrt(2.0)))*(sqrt(-2.0*log(x1(i)))*cos(2*Pi*x2(i)))+d
         evmean
713
         amplitude(i)=(ampdeviation/(2*sqrt(2.0)))*(sqrt(-2.0*log(y1(i)))*cos(2*Pi*y2(i)
         ))+ampmean
714      phase(j)=phase(j)+(sqrt(Pi)*abs(sigma(i))*amplitude(i)/sqrt(log(16.0)))
715      totallength(j)=totallength(j)+sigma(i)
716   enddo
717   !write(*,*) j
718   enddo
719
720   do i=1,1000
721      call ran0(idum,value)
722      z1(i)=value
723      call ran0(idum,value)
724      z2(i)=value
725
         sigma2(i)=(devdeviation/(2*sqrt(2.0)))*(sqrt(-2.0*log(z1(i)))*cos(2*Pi*z2(i)))+
         devmean
726   enddo
727
728   do i=1,1000
729    acumphase(i)=phase(i)/(velocity*hbar)
730   enddo
731   !do i=1,1000
732   ! write(12,*)  acumphase(i),sigma2(i)
733   !enddo
734   if (runnumber.eq.1) then
735   do i=1,1000
736       write(11,*) acumphase(i),sigma2(i)
737   enddo
```

```
738   endif
739
740   return
741   end
742
743   !---------------------------------------------------------------
744
745   subroutine randomnumberdistros(acumphase,sigma2,idum,runnumber)
746   implicit none
747   ! this is a patch subroutine that uses the output of bubblepot and sets the
748   !fwhm of the accumulated phase as ampdeviation
749   integer i,runnumber
750   real*8 devmean,devdeviation,ampmean,ampdeviation,Pi
751   real*8 acumphase(1000),sigma2(1000),value
752   integer idum
753   real*8 x1(1000),x2(1000),y1(1000),y2(1000)
754
755   open(unit=11,file='phasetest2.dat') !testingthephase
756    devmean=250.0d-9
757    devdeviation=250.0d-9
758    ampmean=0.0d0
759    ampdeviation=1.0335d3*1.1935d0
760    Pi=3.14159265d0
761
762   do i=1,1000
763    call ran0(idum,value)
764    x1(i)=value
765    call ran0(idum,value)
766    x2(i)=value
767    call ran0(idum,value)
768    y1(i)=value
769    call ran0(idum,value)
770    y2(i)=value
771
       sigma2(i)=(devdeviation/(2.0d0*sqrt(2.0d0)))*(sqrt(-2.0d0*log(x1(i)))*cos(2.0d0*P
       i*x2(i)))+devmean
772
       acumphase(i)=(ampdeviation/(2.0d0*sqrt(2.0d0)))*(sqrt(-2.0d0*log(x1(i)))*cos(2.0d
       0*Pi*x2(i)))+ampmean
773   enddo
774
775   if (runnumber.eq.1) then
776   do i=1,1000
777       write(11,*) acumphase(i),sigma2(i)
778   enddo
779   endif
780
781   !return
782   end
783
784
785   !---------------------------------------------------------------
786   subroutine ran0(idum,returnvalue)
787   implicit none
788   integer idum,ia,im,iq,ir,mask
789   real*8 returnvalue,am
790   parameter(ia=16807,im=2147483647,am=1.0d0/im)
791   parameter(iq=127773,ir=2836,mask=123459876)
792
793   integer k
794   idum=ieor(idum,mask)
795   k=idum/iq
796   idum=ia*(idum-k*iq)-ir*k
797   if(idum.lt.0)idum=idum+im
798   returnvalue=am*idum
```

```
799     idum=ieor(idum,mask)
800     return
801     end
802
```

E.2 Dephasing Path Integral Program Version 2

This is the reduced and revised version of the dephasing and decoherence path integral code, rewritten by Zilin Chen. It is based on a double-slit setup (no collimating

slits) with random potentials acting on the wave function based on a sum of random Gaussian potentials. See Chapter 5 for more details. Table D.2 outlines the parameters used in this program;

Plane Name	Width	# of Grid-Points	Section Between	Azimuthal Length
Source	15 μm	1500	Source & Double Slit	L_1 = 24 cm
Double Slit Screen	500 nm	1000	Dephaser / Decoherer & Double Slit	0 cm
Dephaser / Decoherer	500 nm	1000	Double Slit & Far Field	L_2 = 24 cm
Distance btw. Slits	150 nm	300		
Slit Width	50	100		
Far Field	800 μm	1500		

Table E.2: Parameters for Path Integral Simulation Setup 2

```
1       PROGRAM Dephaser
2       !includes incoherent summing at first slit
3       ! phaseslope, phasestep can be set at a value, the amp can be turned on and
        off
4       !use msimsl
5       !turned off dephas
6
7       implicit none
8
9       integer Nsource,Nin,Nout,Ndet,endrun,p,q,r,Nx,idum
10      integer blksum,blknum1,blknum,blknumm !if block exceed range,block number
        will be changed
11      parameter(Nsource=1500,Nin=1000,Nout=1000,Ndet=1500,endrun=500,blknum=2)
12
13      integer blkwid(blknum),blkwid1 !the first one is for grating, the second one
        is for source
14      real*8 ee,Pi,h,hbar,c,d,a,me,Energy,gam,ve,lambdae
15      real*8 source,dsource,phiin,dphiin,phiout,dphiout
16      real*8 Dsd,Dds,dephase,image(Nin,endrun),delx,w
17      real*8 screen,dscreen
18      complex*8 kern2(Ndet,Nout),kern(Nin,Nsource),decfactor(Nin)
19      complex*8 amp(Nin)
20      complex*8 cmI,imagephase(Nin)
21      complex*8 wavephidet(Ndet),wavephiin(Nin),wavephiout(Nout)
22      real*8 probscreenmem(Ndet,endrun)
23      integer runnumber
24
25
26      real*8 dephasingsize
27      !call rnset(Q)
28      dephasingsize=0.7d0
29
30
31      idum=6
32
33
34      ee=1.6d-19
35      Pi=3.14159265d0
36      h=6.626d-34
37      hbar=h/2d0/Pi
38      c=3.d8
39
40
41      d=150d-9 !periodicity of grating
42      a=50d-9 !slit width of grating
43
44      me=9.11d-31
45      Energy=1670d0*ee+me*c*c
46      gam=Energy/(me*c*c)
47      ve=sqrt(1d0-1d0/(gam*gam))*c
48      lambdae=h/(gam*me*ve)
49      write(6,*) "The gamma factor is  ",gam
50      write(6,*) "The electron velocity is  ",ve
51
52      source=15d-6
53      write(6,*) "The source is ",source, " wide."
54      screen=8d-4
55      write(6,*) "The screen is  ",screen, "   wide."
56      dephase=0.5d-6
57      write(6,*) "the dephasor ",dephase," wide"
58
59      phiin=dephase
60      phiout=dephase
61
62      dsource=source/Nsource
```

```
63       dscreen=screen/Ndet
64       dphiin=phiin/Nin
65       dphiout=phiout/Nout
66
67       delx=0.25d0*a !shift of gaussian
68       w=2d0*a !width of gaussian
69
70       blknum1=1 !seperate source into 25 parts
71       blkwid1=Nsource/blknum1 !make sure it is an integer
72       Nx=idnint(dephase/(delx))!this is how many gaussians are seperated on the
         grating,if want to close decoherer, make it 1.
73       Nx=1
74
75       Dsd=.24d0       !distanace from the source to the dephasor
76       write(6,*) "The distance from the source to dephasor is ",Dsd
77
78       Dds=.24d0   !distance from the dephasor to the detectorscreen
79       write(6,*) "The distance from the dephasor to the detectorscreen is  ",Dds
80
81       cmI=(0,1)   !the complex number I
82
83    !              The next subroutines sets up the path integral kernels
84       write(6,*) "01"
85
86       call
         firstkernalsubroutine(kern,Nin,Nsource,phiin,dphiin,source,dsource,Dsd,lambdae)
87       call
         secondkernalsubroutine(kern2,Ndet,Nout,screen,dscreen,phiout,dphiout,Dds,lambda
         e)
88    !              The next subroutine sets up the amplitude at the grating
89       call gratingamplitude(amp,phiin,Nin,d,a)
90       call gratingwrite(amp,Nin,phiin,dphiin)
91
92
93
94       do runnumber=1,endrun
95           wavephiin(:)=0d0
96           probscreenmem(:,runnumber)=0d0
97
98    !------------random widthblock on grating ------------------------
99           !this routine create random with block
100          call randomblock(idum,blknum,blknumm,Nin,blkwid)!if close blocked
             source, make the blkwid very big
101
102
103
104   !------------smooth phase-------------------------------!choose one of these
      potential
105          call
             smoothpotential(dephasingsize,idum,Nin,image,imagephase,runnumber,endrun)
106
107   !-------------------------------------------------------
108
109
110          write(6,*) "grating block number is",blknumm
111
112              do q=1,blknum1
113
114                  !This routine propogates from the source to the place right
                    before grating
115                 write(6,*) "source block no.",q
116
117                 call wavefunction1(Nin,Nsource,kern,wavephiin,q,blkwid1)
118                 !plot wavephiin
119                 call plotwavephiin(Nin,wavephiin)
```

```
!------------tiltphase potential ------------------------ !choose one of these potential
        !create tilt potential in each block
                !call tiltphase(idum,imagephase,image,Nin,runnumber,endrun,blknum,blknumm,blkwid)
                !write phase to check
                call phasewrite(endrun,Nin,image)
                !dephasor
                call dephasor(wavephiin,Nin,imagephase,Nsource)
                !grating

                call grating(wavephiin,Nin,amp)


!There starts the decoherent loop
!------------start of decoherence loop---------------------
!wavephiout will change in every loop


                do r=1,Nx
                    write(6,*) r,Nx,q,blknum1
                    blksum=0 !this number is accumulating blockwidth
                    do p=1,blknumm
                        call decoheregenerator(Nin,dephase,decfactor,r,delx,w)

                        call decohere(decfactor,wavephiout,wavephiin,Nin,Nout)!to cancel decoherer, you need go into this route and comment out defactor
                        !The next routine propagates from after the grating to the detector
                        call plotwavephiout(Nout,wavephiout)
                        call wavefunction2(Ndet,Nout,kern2,wavephiout,wavephidet,p,blkwid,blksum,blknum)

                        !the probability is calculated  next
                        call probabilitycalculation(wavephidet,probscreenmem,Ndet,runnumber,endrun)
                        write(6,*) 'block ',p
                    enddo
                    write(6,*) blksum
                enddo !close decoherence loop
            enddo

        write(6,*) 'end of runnumber',runnumber


    enddo
        !next write data to file
        call writeprobabilities(probscreenmem,Ndet,endrun)


    format(E20.6,X,E20.6) !there was a 666 there
    format(E20.6,X,E20.6)

    END
```

```
174
175   !----------------------------------------------------------------------
176
177   subroutine
      firstkernalsubroutine(kern,Nin,Nsource,phiin,dphiin,source,dsource,Dsd,lambdae)
178   implicit none
179     integer i,j,Nin,Nsource
180     real*8 phiin,dphiin,Dsd,lambdae
181     real*8 dist1,dist2,lngh,kernel,Pi,source,dsource
182     complex*8 cmI,kern(Nin,Nsource)
183     Pi=3.14159265d0
184     cmI=(0d0,1d0) !the complex number I
185
186
187       do i=1,Nin
188           do j=1,Nsource
189               dist1=phiin/2d0-dphiin*i+dphiin/2d0                    !checked
190               dist2=source/2d0-dsource*(j)+dsource/2d0     !source    !checked
191               !write(6,*) 'the coordinates of',j,' source point is ',dist2
192               lngh=sqrt((dist1-dist2)**2d0+(Dsd)**2d0)
193               !write(6,*) 'the length of source',j,' to phiin',i,' is',lngh
194               kernel=2d0*Pi*lngh/(lambdae)
195               !write(6,*) i,'to',j,kernel
196               kern(i,j)=cmI*dsin(kernel)+dcos(kernel)
197               !write(6,*) j,' to ',i, real(kern(i,j)),imag(kern(i,j))
198           enddo
199               !write(6,*) 'the coordinates of',i,' phiin point is ',dist1
200       enddo
201       write(6,*) "02"
202   return
203   end
204
205   !----------------------------------------------------------------------
206
207    subroutine
      secondkernalsubroutine(kern2,Ndet,Nout,screen,dscreen,phiout,dphiout,Dds,lambdae)
208    implicit none
209    integer i,j,Nout,Ndet
210    real*8 screen,dscreen,phiout,dphiout,Dds,lambdae
211    real*8 dist1,dist2,lngh,kernel,Pi
212    complex*8 cmI,kern2(Ndet,Nout)
213    Pi=3.14159265d0
214    cmI=(0d0,1d0)  !the complex number I
215
216
217       do i=1,Ndet
218           do j=1,Nout
219               dist1=(screen/2d0-dscreen*i+dscreen/2d0)
220               dist2=(phiout/2d0-dphiout*j+dphiout/2d0)
221               lngh=sqrt((dist1-dist2)**2d0+(Dds)**2d0)
222               kernel=2d0*Pi*lngh/(lambdae)
223               kern2(i,j)=cmI*dsin(kernel)+dcos(kernel)
224           enddo
225         enddo
226       write(6,*) "03"
227   return
228   end
229
230   !----------------------------------------------------------------------
231
232   subroutine gratingamplitude(amp,phiin,Nin,d,a)
233   implicit none
234   integer hfdist,hfsltwid,Nin,j
235   real*8 a,d,phiin
236   complex*8 amp(Nin)
```

```
237       amp(:)=0
238
239       hfdist=idnint(d/phiin/2*Nin)
240       hfsltwid=idnint(a/phiin/2*Nin)
241
242
243
244       do j=(Nin/2-hfdist)-hfsltwid,(Nin/2-hfdist)+hfsltwid
245            amp(j)=1
246        enddo
247        do j=(Nin/2+hfdist)-hfsltwid,(Nin/2+hfdist)+hfsltwid
248            amp(j)=1
249        enddo
250
251
252   return
253   end
254
255
256   !---------------------------------------------------------------------
257   !modified grating to remove unsymmetric phenomenon
258
259   ! subroutine gratingamplitude (amp,Nin)
260   ! implicit none
261   ! integer   jend,k,m,j,nn,Nin
262   ! complex*8 amp(Nin)
263
264       ! jend=25
265       ! do k=0,199
266           ! do m=0,1
267               ! do j=1,jend
268                   ! nn=j+k*jend*2+m*jend
269
270                   ! amp(nn)=m
271               ! enddo
272           ! enddo
273       ! enddo
274       ! do j=1,jend
275           ! nn=j+10000
276           ! amp(nn)=0d0
277       ! enddo
278   ! return
279   ! end
280   !-----------------------------------------------------------------
281   subroutine gratingwrite(amp,Nin,phiin,dphiin)
282   implicit none
283   integer Nin,i
284   real*8 phiin,dphiin
285   complex*8 amp(Nin)
286
287   open(unit=15,file='grating.dat')
288       do i=1,Nin
289           write(15,666) (phiin/2d0-dphiin*i), Real(amp(i))
290       enddo
291       format(E20.6,X,E20.6)
292   close(15)
293
294   return
295   end
296
297
298   !-----------------------------------------------------------------
299
300   subroutine phasewrite(endrun,Nin,image)
301   integer i,runnumber,Nin,endrun
```

```
302   real*8 image(Nin,endrun)
303
304   open(unit=14,file='phase.dat')
305   do runnumber=1,endrun
306       do i=1,Nin
307               write(14,*) runnumber, image(i,runnumber)
308       enddo
309   enddo
310       format(E20.6,X,E20.6)
311   close(14)
312
313   return
314   end
315
316   !---------------------------------------------------------------------
317
318   subroutine wavefunction1(Nin,Nsource,kern,wavephiin,q,blkwid1)
319   implicit none
320   integer Nin,Nsource,i,i2,q,blkwid1
321   complex*8 wavephiin(Nin),phisource(Nsource)
322   complex*8 kern(Nin,Nsource),phiinsum
323       phisource(:)=1d0
324       wavephiin(:)=0d0
325       write(6,*) "1"
326
327       do i=1,Nin
328           phiinsum=0d0
329           do i2=(q-1)*blkwid1+1,q*blkwid1
330               phiinsum=phiinsum+kern(i,i2)*phisource(i2)
331               !write(6,*) i2,'to',i,kern(i,i2)
332           enddo
333           wavephiin(i)=phiinsum
334           !write(6,*) wavephiin(i)
335       enddo
336   return
337   end
338
339   !---------------------------------------------------------------------
340   subroutine plotwavephiin(Nin,wavephiin)
341   implicit none
342   integer Nin,i
343   real*8 phiin_probability(Nin)
344   complex*8 wavephiin(Nin)
345       do i=1,Nin
346           phiin_probability(i)=wavephiin(i)*conjg(wavephiin(i))
347       enddo
348   open(unit=18,file='wavephiin.dat')
349           do i=1,Nin
350               write(18,666) real(wavephiin(i)), phiin_probability(i)
351           enddo
352           format(E20.6,X,E20.6)
353       close(18)
354
355   return
356   end
357
358
359
360   !---------------------------------------------------------------------
361   subroutine dephasor(wavephiin,Nin,imagephase,Nsource)
362   implicit none
363   integer i,Nin,Nsource
364   complex*8 wavephiin(Nin),imagephase(Nsource)
365       do i=1,Nin
366           wavephiin(i)=wavephiin(i)*imagephase(i)
```

```
367       enddo
368   return
369   end
370
371
372   !-----------------------------------------------------------------
373
374   subroutine grating(wavephiin,Nin,amp)
375   implicit none
376   integer Nin,i
377   complex*8 wavephiin(Nin),amp(Nin)
378       do i=1,Nin
379           wavephiin(i)=wavephiin(i)*amp(i)
380           !write(6,*) wavephiout(i)
381       enddo
382
383
384
385   return
386   end
387   !-----------------------------------------------------------------
388   subroutine decoheregenerator(Nin,dephase,decfactor,r,delx,w)
389   implicit none
390   integer Nin,i,r
391   real*8 A,w,Pi,l,dephase,delx
392   complex*8 decfactor(Nin)
393
394   Pi=3.14159265d0
395
396   A=delx/sqrt(w*Pi)
397
398   l=dephase/Nin
399
400
401   do i=1,Nin
402       !if (dexp(-(((i-1)*l-r*w/10)/w)**2).le.1d-6) then
403
404       decfactor(i)=A*dexp(-(((i-1)*l-r*delx)/w)**2)
405
406   enddo
407
408
409
410
411
412   return
413   end
414   !-----------------------------------------------------------------
415   subroutine decohere(decfactor,wavephiout,wavephiin,Nin,Nout)
416   implicit none
417   integer i,Nin,Nout
418   complex*8 wavephiin(Nin),wavephiout(Nout),decfactor(Nin)
419
420
421
422
423   do i=1,Nout
424       wavephiout(i)=wavephiin(i)!*decfactor(i)!comment out decfactor to cancel
          decoherer
425       !write(6,*) wavephiout(i)
426   enddo
427
428
429
430
```

```
431
432   return
433   end
434   !---------------------------------------------------------------------
435   !plot the wave right after the grating
436   subroutine plotwavephiout(Nout,wavephiout)
437   implicit none
438   integer Nout,i
439   real*8 phiout_probability(Nout)
440   complex*8 wavephiout(Nout)
441       do i=1,Nout
442           phiout_probability(i)=wavephiout(i)*conjg(wavephiout(i))
443       enddo
444
445       open(unit=17,file='wavephiout.dat')
446           do i=1,Nout
447               write(17,666) Imag(wavephiout(i)), phiout_probability(i)
448           enddo
449           format(E20.6,X,E20.6)
450       close(17)
451
452   return
453   end
454
455
456
457   !---------------------------------------------------------------------
458   subroutine randomblock(idum,blknum,blknumm,Nin,blkwid)
459   implicit none
460   integer idum,Nin,blknum,i,blknumm
461   integer blkwid(blknum),blkwidsum
462   real*8 returnvalue,Pi,y1,x1,x2
463   blkwidsum=0
464   blkwid(:)=0
465   Pi=3.14159265d0
466
467   do i=1,blknum-1
468     !this is box muller method
469     call ran0(idum,returnvalue)
470     x1=returnvalue
471     call ran0(idum,returnvalue)
472     x2=returnvalue
473     y1=dsqrt(-2.0d0*dlog(x1))*dcos(2.0d0*Pi*x2)
474    !y2=dsqrt(-2.0d0*dlog(x1))*dsin(2.0d0*Pi*x2)
475     blkwid(i)=idnint((300.0d0*y1)+20000.0d0)
476     blkwidsum=blkwidsum+blkwid(i)
477    !in case of blocks exceed the width of grating
478       if (blkwidsum.ge.Nin) then
479           blkwidsum=blkwidsum-blkwid(i)
480           blknumm=i
481           exit
482       endif
483     !write(6,*) blkwid(i)
484   !  write(28,*) amplitude(i), x0(i), sigma(i)
485   enddo
486     blkwid(blknumm)=Nin-blkwidsum   !the rest points will form the last block
487     write(6,*) blkwid
488
489
490   return
491   end
492   !---------------------------------------------------------------------
493
494   subroutine
      tiltphase(idum,imagephase,image,Nin,runnumber,endrun,blknum,blknumm,blkwid)
```

```
495   implicit none
496   integer idum,p,i,Nin,runnumber,endrun,blknum,blknumm,blkwid(blknum),x,blkwidsum
497   real*8 k(blknum),image(Nin,endrun),returnvalue,center
498   complex*8 imagephase(Nin),cmI
499
500   cmI=(0d0,1d0)       !the complex number I
501   blkwidsum=0
502   do i=1,blknumm
503       call ran0(idum,returnvalue)
504       k(i)=(-8d-1)*returnvalue+4d-1
505       write(6,*) returnvalue   !slope
506   enddo
507
508   do p=1,blknumm
509       do i=blkwidsum+1,blkwidsum+blkwid(p)
510       center=(blkwid(p)+1)/2
511       x=i-blkwidsum !so it starts from 1 again
512       image(i,runnumber)=(x-center)*k(p)
513       imagephase(i)=cmI*dsin(image(i,runnumber))+dcos(image(i,runnumber))
514       enddo
515       blkwidsum=blkwidsum+blkwid(p)
516   enddo
517
518   write(6,*) "tiltphase is created"
519   return
520   end
521
522   !---------------------------------------------------------------
523
524   subroutine
      smoothpotential(dephasingsize,idum,Nin,image,imagephase,runnumber,endrun)
525   implicit none
526   integer Ngauss,idum,runnumber,endrun
527   parameter(Ngauss=500)
528   integer Nin,i,j,x0(Ngauss)
529   real*8 dephasingsize,image(Nin,endrun),Pi,returnvalue,amplitude(Ngauss)
530   real*8 x1,x2,y1,sigma(Nin),sumimage(Nin),maxsum
531   complex*8 imagephase(Nin),cmI
532   Pi=3.14159265d0
533   cmI=(0,1)       !the complex number I
534
535   do i=1,Nin
536    sumimage(i)=0d0
537   enddo
538
539   !open(unit=28,file='debug.dat') !phase written
540
541
542   do i=1,Ngauss
543     call ran0(idum,returnvalue)
544     amplitude(i)=dephasingsize*2.0d0*Pi*returnvalue
545     call ran0(idum,returnvalue)
546     x0(i)=idnint(returnvalue*Nin)
547
548     call ran0(idum,returnvalue)
549     x1=returnvalue
550     call ran0(idum,returnvalue)
551     x2=returnvalue
552     y1=dsqrt(-2.0d0*dlog(x1))*dcos(2.0d0*Pi*x2)
553    !y2=dsqrt(-2.0d0*dlog(x1))*dsin(2.0d0*Pi*x2)
554     sigma(i)=(1d0*y1)+8d0
555   !  write(28,*) amplitude(i), x0(i), sigma(i)
556   enddo
557
558   do i=1,Ngauss
```

```
559    do j=1,Nin
560
         sumimage(j)=sumimage(j)+amplitude(i)*dexp(-((j-x0(i))/(dsqrt(2.0d0)*sigma(i)))*
         *2)
561    enddo
562   enddo
563
564   !find the maximum value of sumimage
565   maxsum=sumimage(1)
566   do i=1,Nin-1
567    if (maxsum.le.sumimage(i+1)) then
568    maxsum=sumimage(i+1)
569    endif
570   enddo
571
572   do j=1,Nin
573      image(j,runnumber)=sumimage(j)*dephasingsize*2*Pi
574      imagephase(j)=cmI*dsin(image(j,runnumber))+dcos(image(j,runnumber))
575   enddo
576
577   return
578   end
579   !---------------------------------------------------------------------
580
581   subroutine
      wavefunction2(Ndet,Nout,kern2,wavephiout,wavephidet,p,blkwid,blksum,blknum)

582   implicit none
583   integer Ndet,Nout,i,i2,p,blknum,blkwid(blknum),blksum
584   complex*8 phidetsum
585   complex*8 kern2(Ndet,Nout),wavephidet(Ndet),wavephiout(Nout)
586
587
588
589       wavephidet(:)=0d0
590       write(6,*) "2"
591       do i=1,Ndet
592           phidetsum=0d0
593           do i2=blksum+1,blksum+blkwid(p)
594               phidetsum=phidetsum+kern2(i,i2)*wavephiout(i2)
595           enddo
596           wavephidet(i)=phidetsum
597           !write(6,*) wavephiout(i)
598       enddo
599       blksum=blksum+blkwid(p)
600   return
601   end
602
603   !---------------------------------------------------------------------
604
605   subroutine probabilitycalculation(wavephidet,probscreenmem,Ndet,runnumber,endrun)
606   implicit none
607   integer i,Ndet,runnumber,endrun
608   real*8 probscreen(Ndet),norm,probscreenmem(Ndet,endrun)
609   complex*8 wavephidet(Ndet)
610           probscreen(:)=0d0
611       do i=1,Ndet
612           probscreen(i)=wavephidet(i)*conjg(wavephidet(i))
613       enddo
614
615       norm=0d0
616       do i=1,Ndet
617           norm=norm+probscreen(i)
618       enddo
619           write(6,*) norm
```

```
620       write(6,*) "before"
621
622       do i=1,Ndet
623           probscreen(i)=(1/norm)*probscreen(i)
624       enddo
625
626       write(6,*) "after"
627
628       do i=1,Ndet
629           probscreenmem(i,runnumber)=probscreen(i)+probscreenmem(i,runnumber)
630
631       enddo
632
633    return
634    end
635
636    !---------------------------------------------------------------
637
638   subroutine writeprobabilities(probscreenmem,Ndet,endrun)
639   implicit none
640   integer i,Ndet,runnumber,endrun
641   real*8 probscreenmem(Ndet,endrun)
642
643   open(unit=13,file='probabilitydetector.dat') !probability on detector
644       do runnumber=1,endrun
645           do i=1,Ndet
646               write(13,*) runnumber, probscreenmem(i,runnumber)
647           enddo
648       enddo
649       format(E20.6,X,E20.6) !there was a 666 there
650
651   close(13)
652
653   return
654   end
655
656   !---------------------------------------------------------------
657
658   subroutine ran0(idum,returnvalue)
659   implicit none
660   integer idum,ia,im,iq,ir,mask
661   real*8 returnvalue,am
662   parameter(ia=16807,im=2147483647,am=1.0d0/im)
663   parameter(iq=127773,ir=2836,mask=123459876)
664
665   integer k
666   idum=ieor(idum,mask)
667   k=idum/iq
668   idum=ia*(idum-k*iq)-ir*k
669   if(idum.lt.0)idum=idum+im
670   returnvalue=am*idum
671   idum=ieor(idum,mask)
672   return
673   end
674
675
676
677   !---------------------------------------------------------------
```

E.3 Decoherence Path Integral Program Version 2

This is the revised and reduced version of the decoherence path integral code, written by Zilin Chen. It is based on a double-slit setup (no collimating slits) with an

incoherent summing of Gaussians at the grating with random potentials acting on the wave function identical to that of D.3. See Chapter 5 for more details

```
      PROGRAM Decoherer
      !includes incoherent summing at first slit
      ! phaseslope, phasestep can be set at a value, the amp can be turned on and
      off
      !use msimsl
      !turned off dephas

      implicit none

      integer Nsource,Nin,Nout,Ndet,endrun,p,q,r,Nx,idum
      integer blksum,blknum1,blknum,blknumm !if block exceed range,block number
      will be changed
      parameter(Nsource=1500,Nin=1000,Nout=1000,Ndet=1500,endrun=1,blknum=2)

      integer blkwid(blknum),blkwid1 !the first one is for grating, the second one
      is for source
      real*8 ee,Pi,h,hbar,c,d,a,me,Energy,gam,ve,lambdae
      real*8 source,dsource,phiin,dphiin,phiout,dphiout
      real*8 Dsd,Dds,dephase,image(Nin,endrun),delx,w
      real*8 screen,dscreen
      complex*8 kern2(Ndet,Nout),kern(Nin,Nsource),decfactor(Nin)
      complex*8 amp(Nin)
      complex*8 cmI,imagephase(Nin)
      complex*8 wavephidet(Ndet),wavephiin(Nin),wavephiout(Nout)
      real*8 probscreenmem(Ndet,endrun)
      integer runnumber


      real*8 dephasingsize
      !call rnset(Q)
      dephasingsize=0.7d0


      idum=5


      ee=1.6d-19
      Pi=3.14159265d0
      h=6.626d-34
      hbar=h/2d0/Pi
      c=3.d8


      d=150d-9 !periodicity of grating
      a=50d-9 !slit width of grating

      me=9.11d-31
      Energy=1670d0*ee+me*c*c
      gam=Energy/(me*c*c)
      ve=sqrt(1d0-1d0/(gam*gam))*c
      lambdae=h/(gam*me*ve)
      write(6,*) "The gamma factor is  ",gam
      write(6,*) "The electron velocity is  ",ve

      source=15d-6
      write(6,*) "The source is ",source, " wide."
      screen=8d-4
      write(6,*) "The screen is  ",screen, "   wide."
      dephase=0.5d-6
      write(6,*) "the dephasor ",dephase," wide"

      phiin=dephase
      phiout=dephase

      dsource=source/Nsource
```

```
63       dscreen=screen/Ndet
64       dphiin=phiin/Nin
65       dphiout=phiout/Nout
66
67       delx=0.25d0*a !shift of gaussian
68       w=2d0*a !width of gaussian
69
70       blknum1=1 !seperate source into 25 parts
71       blkwid1=Nsource/blknum1 !make sure it is an integer
72       Nx=idnint(dephase/(delx))!this is how many gaussians are on the grating
73
74
75       Dsd=.24d0       !distanace from the source to the dephasor
76       write(6,*) "The distance from the source to dephasor is ",Dsd
77
78       Dds=.24d0   !distance from the dephasor to the detectorscreen
79       write(6,*) "The distance from the dephasor to the detectorscreen is  ",Dds
80
81       cmI=(0,1)   !the complex number I
82
83   !               The next subroutines sets up the path integral kernels
84      write(6,*) "01"
85
86       call
         firstkernalsubroutine(kern,Nin,Nsource,phiin,dphiin,source,dsource,Dsd,lambdae)
87       call
         secondkernalsubroutine(kern2,Ndet,Nout,screen,dscreen,phiout,dphiout,Dds,lambda
         e)
88   !               The next subroutine sets up the amplitude at the grating
89       call gratingamplitude(amp,phiin,Nin,d,a)
90       call gratingwrite(amp,Nin,phiin,dphiin)
91
92
93
94        do runnumber=1,endrun
95            wavephiin(:)=0d0
96            probscreenmem(:,runnumber)=0d0
97
98   !------------random widthblock on grating ------------------------
99            !this routine create random with block
100           call randomblock(idum,blknum,blknumm,Nin,blkwid)
101
102
103
104  !------------smooth phase-------------------------------!choose one of these
     potential
105           call
              smoothpotential(dephasingsize,idum,Nin,image,imagephase,runnumber,endrun)
106
107  !-------------------------------------------------------
108
109
110           write(6,*) "grating block number is",blknumm
111
112               do q=1,blknum1
113
114                   !This routine propogates from the source to the place right
                      before grating
115                   write(6,*) "source block no.",q
116
117                   call wavefunction1(Nin,Nsource,kern,wavephiin,q,blkwid1)
118                   !plot wavephiin
119                   call plotwavephiin(Nin,wavephiin)
120  !------------tiltphase potential ------------------------ !choose one of these
     potential
```

```
121          !create tilt potential in each block
122                  !call
                     tiltphase(idum,imagephase,image,Nin,runnumber,endrun,blknum,blknum
                     m,blkwid)
123                  !write phase to check
124                  call phasewrite(endrun,Nin,image)
125                  !dephasor
126                  call dephasor(wavephiin,Nin,imagephase,Nsource)
127                  !grating
128
129                  call grating(wavephiin,Nin,amp)
130
131
132  !There starts the decoherent loop
133  !------------start of decoherence loop---------------------
134  !wavephiout will change in every loop
135
136
137                  do r=1,Nx
138                      write(6,*) r,Nx,q,blknum1
139                      blksum=0 !this number is accumulating blockwidth
140                      do p=1,blknumm
141                          call decoheregenerator(Nin,dephase,decfactor,r,delx,w)
142
143                          call decohere(decfactor,wavephiout,wavephiin,Nin,Nout)
144                          !The next routine propagates from after the grating to
                             the detector
145                          call plotwavephiout(Nout,wavephiout)
146                          call
                             wavefunction2(Ndet,Nout,kern2,wavephiout,wavephidet,p,blkw
                             id,blksum,blknum)
147
148                          !the probability is calculated  next
149                          call
                             probabilitycalculation(wavephidet,probscreenmem,Ndet,runnu
                             mber,endrun)
150                          write(6,*) 'block ',p
151                      enddo
152                      write(6,*) blksum
153                  enddo !close decoherence loop
154              enddo
155
156          write(6,*) 'end of runnumber',runnumber
157
158
159      enddo
160          !next write data to file
161          call writeprobabilities(probscreenmem,Ndet,endrun)
162
163
164      format(E20.6,X,E20.6) !there was a 666 there
165      format(E20.6,X,E20.6)
166
167      END
168
169
170
171
172
173
174
175  !---------------------------------------------------------------------
176
177  subroutine
     firstkernalsubroutine(kern,Nin,Nsource,phiin,dphiin,source,dsource,Dsd,lambdae)
```

```
178   implicit none
179     integer i,j,Nin,Nsource
180     real*8 phiin,dphiin,Dsd,lambdae
181     real*8 dist1,dist2,lngh,kernel,Pi,source,dsource
182     complex*8 cmI,kern(Nin,Nsource)
183     Pi=3.14159265d0
184     cmI=(0d0,1d0) !the complex number I
185
186
187       do i=1,Nin
188           do j=1,Nsource
189               dist1=phiin/2d0-dphiin*i+dphiin/2d0                      !checked
190               dist2=source/2d0-dsource*(j)+dsource/2d0     !source    !checked
191               !write(6,*) 'the coordinates of',j,' source point is ',dist2
192               lngh=sqrt((dist1-dist2)**2d0+(Dsd)**2d0)
193               !write(6,*) 'the length of source',j,' to phiin',i,' is',lngh
194               kernel=2d0*Pi*lngh/(lambdae)
195               !write(6,*) i,'to',j,kernel
196               kern(i,j)=cmI*dsin(kernel)+dcos(kernel)
197               !write(6,*) j,' to ',i, real(kern(i,j)),imag(kern(i,j))
198           enddo
199               !write(6,*) 'the coordinates of',i,' phiin point is ',dist1
200       enddo
201       write(6,*) "02"
202   return
203   end
204
205   !---------------------------------------------------------------------
206
207    subroutine
      secondkernalsubroutine(kern2,Ndet,Nout,screen,dscreen,phiout,dphiout,Dds,lambdae)
208    implicit none
209    integer i,j,Nout,Ndet
210    real*8 screen,dscreen,phiout,dphiout,Dds,lambdae
211    real*8 dist1,dist2,lngh,kernel,Pi
212    complex*8 cmI,kern2(Ndet,Nout)
213    Pi=3.14159265d0
214    cmI=(0d0,1d0)  !the complex number I
215
216
217       do i=1,Ndet
218           do j=1,Nout
219               dist1=(screen/2d0-dscreen*i+dscreen/2d0)
220               dist2=(phiout/2d0-dphiout*j+dphiout/2d0)
221               lngh=sqrt((dist1-dist2)**2d0+(Dds)**2d0)
222               kernel=2d0*Pi*lngh/(lambdae)
223               kern2(i,j)=cmI*dsin(kernel)+dcos(kernel)
224           enddo
225         enddo
226       write(6,*) "03"
227   return
228   end
229
230   !---------------------------------------------------------------------
231
232   subroutine gratingamplitude(amp,phiin,Nin,d,a)
233   implicit none
234   integer hfdist,hfsltwid,Nin,j
235   real*8 a,d,phiin
236   complex*8 amp(Nin)
237       amp(:)=0
238
239       hfdist=idnint(d/phiin/2*Nin)
240       hfsltwid=idnint(a/phiin/2*Nin)
241
```

```
242
243
244         do j=(Nin/2-hfdist)-hfsltwid,(Nin/2-hfdist)+hfsltwid
245              amp(j)=1
246          enddo
247          do j=(Nin/2+hfdist)-hfsltwid,(Nin/2+hfdist)+hfsltwid
248              amp(j)=1
249          enddo
250
251
252       return
253       end
254
255
256       !---------------------------------------------------------------------
257       !modified grating to remove unsymmetric phenomenon
258
259       ! subroutine gratingamplitude (amp,Nin)
260       ! implicit none
261       ! integer   jend,k,m,j,nn,Nin
262       ! complex*8 amp(Nin)
263
264           ! jend=25
265           ! do k=0,199
266               ! do m=0,1
267                   ! do j=1,jend
268                       ! nn=j+k*jend*2+m*jend
269
270                       ! amp(nn)=m
271                   ! enddo
272               ! enddo
273           ! enddo
274           ! do j=1,jend
275               ! nn=j+10000
276               ! amp(nn)=0d0
277           ! enddo
278       ! return
279       ! end
280       !---------------------------------------------------------------
281       subroutine gratingwrite(amp,Nin,phiin,dphiin)
282       implicit none
283       integer Nin,i
284       real*8 phiin,dphiin
285       complex*8 amp(Nin)
286
287       open(unit=15,file='grating.dat')
288           do i=1,Nin
289               write(15,666) (phiin/2d0-dphiin*i), Real(amp(i))
290           enddo
291           format(E20.6,X,E20.6)
292       close(15)
293
294       return
295       end
296
297
298       !---------------------------------------------------------------
299
300       subroutine phasewrite(endrun,Nin,image)
301       integer i,runnumber,Nin,endrun
302       real*8 image(Nin,endrun)
303
304       open(unit=14,file='phase.dat')
305       do runnumber=1,endrun
306           do i=1,Nin
```

```
307                 write(14,*) runnumber, image(i,runnumber)
308             enddo
309         enddo
310             format(E20.6,X,E20.6)
311         close(14)
312
313         return
314         end
315
316         !---------------------------------------------------------------------
317
318         subroutine wavefunction1(Nin,Nsource,kern,wavephiin,q,blkwid1)
319         implicit none
320         integer Nin,Nsource,i,i2,q,blkwid1
321         complex*8 wavephiin(Nin),phisource(Nsource)
322         complex*8 kern(Nin,Nsource),phiinsum
323             phisource(:)=1d0
324             wavephiin(:)=0d0
325             write(6,*) "1"
326
327             do i=1,Nin
328                 phiinsum=0d0
329                 do i2=(q-1)*blkwid1+1,q*blkwid1
330                     phiinsum=phiinsum+kern(i,i2)*phisource(i2)
331                     !write(6,*) i2,'to',i,kern(i,i2)
332                 enddo
333                 wavephiin(i)=phiinsum
334                 !write(6,*) wavephiin(i)
335             enddo
336         return
337         end
338
339         !---------------------------------------------------------------------
340         subroutine plotwavephiin(Nin,wavephiin)
341         implicit none
342         integer Nin,i
343         real*8 phiin_probability(Nin)
344         complex*8 wavephiin(Nin)
345             do i=1,Nin
346                 phiin_probability(i)=wavephiin(i)*conjg(wavephiin(i))
347             enddo
348         open(unit=18,file='wavephiin.dat')
349                 do i=1,Nin
350                     write(18,666) real(wavephiin(i)), phiin_probability(i)
351                 enddo
352                 format(E20.6,X,E20.6)
353             close(18)
354
355         return
356         end
357
358
359
360         !---------------------------------------------------------------------
361         subroutine dephasor(wavephiin,Nin,imagephase,Nsource)
362         implicit none
363         integer i,Nin,Nsource
364         complex*8 wavephiin(Nin),imagephase(Nsource)
365             do i=1,Nin
366                 wavephiin(i)=wavephiin(i)*imagephase(i)
367             enddo
368         return
369         end
370
371
```

```
!-----------------------------------------------------------------

subroutine grating(wavephiin,Nin,amp)
implicit none
integer Nin,i
complex*8 wavephiin(Nin),amp(Nin)
    do i=1,Nin
        wavephiin(i)=wavephiin(i)*amp(i)
        !write(6,*) wavephiout(i)
    enddo



return
end
!-----------------------------------------------------------------
subroutine decoheregenerator(Nin,dephase,decfactor,r,delx,w)
implicit none
integer Nin,i,r
real*8 A,w,Pi,l,dephase,delx
complex*8 decfactor(Nin)

Pi=3.14159265d0

A=delx/sqrt(w*Pi)

l=dephase/Nin


do i=1,Nin
    !if (dexp(-(((i-1)*l-r*w/10)/w)**2).le.1d-6) then

    decfactor(i)=A*dexp(-(((i-1)*l-r*delx)/w)**2)

enddo




return
end
!-----------------------------------------------------------------
subroutine decohere(decfactor,wavephiout,wavephiin,Nin,Nout)
implicit none
integer i,Nin,Nout
complex*8 wavephiin(Nin),wavephiout(Nout),decfactor(Nin)



do i=1,Nout
    wavephiout(i)=wavephiin(i)*decfactor(i)
    !write(6,*) wavephiout(i)
enddo




return
end
!-----------------------------------------------------------------
!plot the wave right after the grating
subroutine plotwavephiout(Nout,wavephiout)
```

```
437   implicit none
438   integer Nout,i
439   real*8 phiout_probability(Nout)
440   complex*8 wavephiout(Nout)
441       do i=1,Nout
442           phiout_probability(i)=wavephiout(i)*conjg(wavephiout(i))
443       enddo
444
445       open(unit=17,file='wavephiout.dat')
446           do i=1,Nout
447               write(17,666) Imag(wavephiout(i)), phiout_probability(i)
448           enddo
449           format(E20.6,X,E20.6)
450       close(17)
451
452   return
453   end
454
455
456
457   !---------------------------------------------------------------------
458   subroutine randomblock(idum,blknum,blknumm,Nin,blkwid)
459   implicit none
460   integer idum,Nin,blknum,i,blknumm
461   integer blkwid(blknum),blkwidsum
462   real*8 returnvalue,Pi,y1,x1,x2
463   blkwidsum=0
464   blkwid(:)=0
465   Pi=3.14159265d0
466
467   do i=1,blknum-1
468     !this is box muller method
469     call ran0(idum,returnvalue)
470     x1=returnvalue
471     call ran0(idum,returnvalue)
472     x2=returnvalue
473     y1=dsqrt(-2.0d0*dlog(x1))*dcos(2.0d0*Pi*x2)
474    !y2=dsqrt(-2.0d0*dlog(x1))*dsin(2.0d0*Pi*x2)
475     blkwid(i)=idnint((300.0d0*y1)+20000.0d0)
476     blkwidsum=blkwidsum+blkwid(i)
477    !in case of blocks exceed the width of grating
478       if (blkwidsum.ge.Nin) then
479           blkwidsum=blkwidsum-blkwid(i)
480           blknumm=i
481           exit
482       endif
483     !write(6,*) blkwid(i)
484   !  write(28,*) amplitude(i), x0(i), sigma(i)
485   enddo
486     blkwid(blknumm)=Nin-blkwidsum   !the rest points will form the last block
487     write(6,*) blkwid
488
489
490   return
491   end
492   !---------------------------------------------------------------------
493
494   subroutine
      tiltphase(idum,imagephase,image,Nin,runnumber,endrun,blknum,blknumm,blkwid)
495   implicit none
496   integer idum,p,i,Nin,runnumber,endrun,blknum,blknumm,blkwid(blknum),x,blkwidsum
497   real*8 k(blknum),image(Nin,endrun),returnvalue,center
498   complex*8 imagephase(Nin),cmI
499
500   cmI=(0d0,1d0)       !the complex number I
```

```
501   blkwidsum=0
502   do i=1,blknumm
503       call ran0(idum,returnvalue)
504       k(i)=(-8d-1)*returnvalue+4d-1
505       write(6,*) returnvalue   !slope
506   enddo
507
508   do p=1,blknumm
509       do i=blkwidsum+1,blkwidsum+blkwid(p)
510       center=(blkwid(p)+1)/2
511       x=i-blkwidsum !so it starts from 1 again
512       image(i,runnumber)=(x-center)*k(p)
513       imagephase(i)=cmI*dsin(image(i,runnumber))+dcos(image(i,runnumber))
514       enddo
515       blkwidsum=blkwidsum+blkwid(p)
516   enddo
517
518   write(6,*) "tiltphase is created"
519   return
520   end
521
522   !---------------------------------------------------------------------
523
524   subroutine
      smoothpotential(dephasingsize,idum,Nin,image,imagephase,runnumber,endrun)
525   implicit none
526   integer Ngauss,idum,runnumber,endrun
527   parameter(Ngauss=500)
528   integer Nin,i,j,x0(Ngauss)
529   real*8 dephasingsize,image(Nin,endrun),Pi,returnvalue,amplitude(Ngauss)
530   real*8 x1,x2,y1,sigma(Nin),sumimage(Nin),maxsum
531   complex*8 imagephase(Nin),cmI
532   Pi=3.14159265d0
533   cmI=(0,1)       !the complex number I
534
535   do i=1,Nin
536    sumimage(i)=0d0
537   enddo
538
539   !open(unit=28,file='debug.dat') !phase written
540
541
542   do i=1,Ngauss
543     call ran0(idum,returnvalue)
544     amplitude(i)=dephasingsize*2.0d0*Pi*returnvalue
545     call ran0(idum,returnvalue)
546     x0(i)=idnint(returnvalue*Nin)
547
548     call ran0(idum,returnvalue)
549     x1=returnvalue
550     call ran0(idum,returnvalue)
551     x2=returnvalue
552     y1=dsqrt(-2.0d0*dlog(x1))*dcos(2.0d0*Pi*x2)
553    !y2=dsqrt(-2.0d0*dlog(x1))*dsin(2.0d0*Pi*x2)
554     sigma(i)=(1d0*y1)+8d0
555   !  write(28,*) amplitude(i), x0(i), sigma(i)
556   enddo
557
558   do i=1,Ngauss
559    do j=1,Nin
560
        sumimage(j)=sumimage(j)+amplitude(i)*dexp(-((j-x0(i))/(dsqrt(2.0d0)*sigma(i)))*
        *2)
561    enddo
562   enddo
```

```
563
564   !find the maximum value of sumimage
565   maxsum=sumimage(1)
566   do i=1,Nin-1
567    if (maxsum.le.sumimage(i+1)) then
568    maxsum=sumimage(i+1)
569    endif
570   enddo
571
572   do j=1,Nin
573      image(j,runnumber)=sumimage(j)*dephasingsize*2*Pi
574      imagephase(j)=cmI*dsin(image(j,runnumber))+dcos(image(j,runnumber))
575   enddo
576
577   return
578   end
579   !---------------------------------------------------------------------
580
581   subroutine
      wavefunction2(Ndet,Nout,kern2,wavephiout,wavephidet,p,blkwid,blksum,blknum)

582   implicit none
583   integer Ndet,Nout,i,i2,p,blknum,blkwid(blknum),blksum
584   complex*8 phidetsum
585   complex*8 kern2(Ndet,Nout),wavephidet(Ndet),wavephiout(Nout)
586
587
588
589       wavephidet(:)=0d0
590       write(6,*) "2"
591       do i=1,Ndet
592           phidetsum=0d0
593           do i2=blksum+1,blksum+blkwid(p)
594               phidetsum=phidetsum+kern2(i,i2)*wavephiout(i2)
595           enddo
596           wavephidet(i)=phidetsum
597           !write(6,*) wavephiout(i)
598       enddo
599       blksum=blksum+blkwid(p)
600   return
601   end
602
603   !---------------------------------------------------------------------
604
605   subroutine probabilitycalculation(wavephidet,probscreenmem,Ndet,runnumber,endrun)
606   implicit none
607   integer i,Ndet,runnumber,endrun
608   real*8 probscreen(Ndet),norm,probscreenmem(Ndet,endrun)
609   complex*8 wavephidet(Ndet)
610           probscreen(:)=0d0
611       do i=1,Ndet
612           probscreen(i)=wavephidet(i)*conjg(wavephidet(i))
613       enddo
614
615       norm=0d0
616       do i=1,Ndet
617           norm=norm+probscreen(i)
618       enddo
619           write(6,*) norm
620       write(6,*) "before"
621
622       do i=1,Ndet
623           probscreen(i)=(1/norm)*probscreen(i)
624       enddo
625
```

```
626       write(6,*) "after"
627
628       do i=1,Ndet
629           probscreenmem(i,runnumber)=probscreen(i)+probscreenmem(i,runnumber)
630
631       enddo
632
633     return
634     end
635
636     !---------------------------------------------------------------------
637
638   subroutine writeprobabilities(probscreenmem,Ndet,endrun)
639   implicit none
640   integer i,Ndet,runnumber,endrun
641   real*8 probscreenmem(Ndet,endrun)
642
643   open(unit=13,file='probabilitydetector.dat') !probability on detector
644       do runnumber=1,endrun
645           do i=1,Ndet
646               write(13,*) runnumber, probscreenmem(i,runnumber)
647           enddo
648       enddo
649       format(E20.6,X,E20.6) !there was a 666 there
650
651   close(13)
652
653   return
654   end
655
656   !---------------------------------------------------------------------
657
658   subroutine ran0(idum,returnvalue)
659   implicit none
660   integer idum,ia,im,iq,ir,mask
661   real*8 returnvalue,am
662   parameter(ia=16807,im=2147483647,am=1.0d0/im)
663   parameter(iq=127773,ir=2836,mask=123459876)
664
665   integer k
666   idum=ieor(idum,mask)
667   k=idum/iq
668   idum=ia*(idum-k*iq)-ir*k
669   if(idum.lt.0)idum=idum+im
670   returnvalue=am*idum
671   idum=ieor(idum,mask)
672   return
673   end
674
675
676
677   !---------------------------------------------------------------------
678
```

E.4 Matlab Code for Entropy Calculation

```
%Entropy Calculation
%Author: Peter Beierle
%Last modified:7/6/2016

for i=1:numel(VarName1)
    element(i)=VarName1(i)+sqrt(-1)*VarName2(i);
end



matrix=zeros(1500,1500);
for i=1:1500
    for j=1:1500
        gogo=(j-1)*1500+i;
        matrix(j,i)=element(gogo);
    end
end


totalprob=0;
for i=1:1500
    totalprob=totalprob+matrix(i,i);
end

matrix=matrix/totalprob;



for i=1:1500
    probability(i)=matrix(i,i);
end
%plot(probability)

 values=eig(matrix);

 eigenmatrix=zeros(1500,1500);
 for i=1:1500
     eigenmatrix(i,i)=values(i);
 end


 logvalues=(log(values));

 diagonalloggedmatrix=zeros(1500,1500);
 for i=1:1500
    diagonalloggedmatrix(i,i)=logvalues(i);
 end



 %product=matrix*diagonalloggedmatrix;
 product=eigenmatrix*diagonalloggedmatrix;


 entropy=0;
 for i=1:1500
```

```
entropy=entropy+product(i,i);
end
entropy
```

Published with MATLAB® R2014a

E.5 Matlab Code for Correlation Calculation

```
%written by peter beierle
%last revised 09.08.2016
%testing correlation of fields equation
clc;
pi=3.14159265;
x1=rand(1,1);
x1prime=rand(1,1);

x2=rand(1,1);
x2prime=rand(1,1);

lambda=1;
i=sqrt(-1);
n=10000;
phiR1=rand(n,1)*2*pi;
phiR2=rand(n,1)*2*pi;

E1star_x1prm=exp(-i*phiR1)*exp(-i*2*pi*x1prime/lambda);
E2star_x2prm=exp(-i*phiR2)*exp(-i*2*pi*x2prime/lambda);
E1_x1=exp(i*phiR1)*exp(i*2*pi*x1/lambda);
E2_x2=exp(i*phiR2)*exp(i*2*pi*x2/lambda);

E1star_x1=exp(-i*phiR1)*exp(-i*2*pi*x1/lambda);
E1_x1prm=exp(i*phiR1)*exp(i*2*pi*x1prime/lambda);

E2star_x2=exp(-i*phiR2)*exp(-i*2*pi*x2/lambda);
E2_x2prm=exp(i*phiR2)*exp(i*2*pi*x2prime/lambda);

left=sum(E1star_x1prm.*E2star_x2prm.*E1_x1.*E2_x2)/n
%right1 is from Incoherent Coincidence Imaging... PRL (2004)
right1=(sum(E1star_x1.*E1_x1prm)*sum(E2star_x2.*E2_x2prm)/(n*n))+(sum(E1star_x1.*E2_x2prm)*sum(E2star_x2.*E1_x1prm)/(n*n))
%right2 is from J.W. Goodman's Statistical Optics
right2=(sum(E1star_x1prm.*E1_x1)*sum(E2star_x2prm.*E2_x2)/(n*n))+(sum(E1star_x1prm.*E2_x2)*sum(E2star_x2prm.*E1_x1)/(n*n))


left =

   0.7179 + 0.6961i


right1 =

   0.7180 - 0.6962i


right2 =

   0.7180 + 0.6962i
```

Published with MATLAB® R2014a

List of Figures

Chapter 1: Introduction

Chapter 2: Theory

Chapter 3: Optical Electron Switch

Chapter 4: Decoherence Experiment

Chapter 5: Dephasing vs Decoherence