


SINGLE-BEAM LASER TRAPS FOR
OPAQUE OR REFLECTIVE
PARTICLES

by
Benjamin A Bellville

Submitted to the Department of Physics and Astronomy in partial fulfillment of
graduation requirements for the degree of
Bachelor of Science

Brigham Young University
April 2002

Advisor and Thesis Coordinator:
Justin Peatross

Signature: 

Department Chair: Steven Turley

Signature 

Abstract

To date, single-beam laser traps have not been available for opaque or reflective particles, which are high-intensity fleeing. Three potential laser traps for opaque and reflective particles are Chaloupka's trap, the Poisson trap, and the spherical aberration trap. The latter two are being developed at Brigham Young University, and are presented for the first time here. The Poisson trap is best characterized and clearly customizable for experimental application. The spherical aberration trap is not well understood as yet, but promises to be the simplest to set up and use once it is characterized.

The setup for the Poisson trap also provides an opportunity to examine the behavior of the Schrödinger wave equation in two dimensions. The paraxial wave equation, which describes laser propagation, has the same form as the Schrödinger wave equation where the longitudinal dimension plays the role of time. Following this analogy, negative centrifugal force is demonstrated.

Acknowledgements

I would like to acknowledge the efforts of Cody Bliss in experimentally trapping the particles in the spherical aberration trap, even though it was our mistake. I would also like to thank Nate Terry and John Madsen for helping me learn Matlab, and Dr. Ross Spencer for his discussion on various numerical methods, some of which were beyond me. And, of course, I would like to thank Dr. Justin Peatross for his patience with my computational efforts and fingerprints.

Contents

1. Introduction	1
1.1 Chaloupka's Trap	1
1.2 Poisson Trap	3
1.3 Spherical Aberration Trap	4
1.4 Diffraction Integrals	5
2. Poisson Trap	8
2.1 Ring Location	9
2.2 Trap Radius	10
2.3 Wall Intensity	11
2.4 Trap Length	12
2.5 Implementing a Poisson Trap	13
3. Laser Propagation and Quantum Mechanics	14
3.1 Making the Correlation	14
3.2 Negative Centrifugal Force	15
3.3 Experimental and Numerical Data from Poisson's Spot	16
4 Spherical Aberration Trap	20
4.1 Numerical Method for Modeling the SAT	20
4.2 Derivation of the Spherical Lens	22
4.3 Conclusion on SAT	24
5 Conclusion	26
6 References	27
Appendices	
A Programs for Numerical Profiling	28
A.1 PoissonTrap2.m	28
A.2 intPoly4.m	32
A.3 shuffle.m	33
A.4 intFresnel.m	34
A.5 poly10.m	36
A.6 mirror.m	37
A.7 mirrorN.m	37
B Numerical Integration Method	38

List of Figures

1.1	Half-wave Plate	2
1.2	Making Chaloupka's Trap	2
1.3	Chaloupka's Trap	2
1.4	Optical Configuration for the Poisson Trap	3
1.5	Paraxial Approximation	4
1.6	Charcoal Particle in SAT	5
1.7	Variables for Diffraction Integrals	6
2.1	Poisson Trap	8
2.2	Variables for the Poisson Trap	9
2.2	Poisson Trap with $1\omega_0$ and $2\omega_0$, numerically generated	10
2.3	Experimental vs. Theoretical Profiles Near the Axial Boundary of a Poisson Trap, $z=0.5 Z_T$	12
3.1	Light Doughnut	16
3.2	Radial, Mass-center of Energy	16
3.3	Experimental and Numeric Profiles of Poisson's Spot	17-19
4.1	Optical Configuration for SAT	20
4.2	Spherical Aberration Trap	21
4.3	On-axis Intensity	21
4.4	Axis-Parallel Rays	22
4.5	Distances	23
5.1	Contour Plots of Chaloupka and Poisson	26

Chapter 1: Introduction

Laser traps use laser light to move, manipulate and contain small particles. Light has momentum. Estimates of how much force light exerts on a particle can be determined by classic Newtonian conservation of momentum [1]. Clearly, translucent particles experience forces very different from those exerted on opaque or reflective particles by the same light. It has been shown that translucent particles are high-intensity seeking [2]; they stay in a laser focus.

Opaque or reflective particles are high-intensity fleeing, and cannot be trapped in a simple laser focus. They must be surrounded by, but not in, high-intensity light to be trapped. To date, no simple laser traps have been developed for opaque or translucent particles. This thesis discusses and compares three potential laser traps for opaque or reflective particles. All profiles for these traps are investigated by numerical evaluation of the Fresnel diffraction integral. [3]

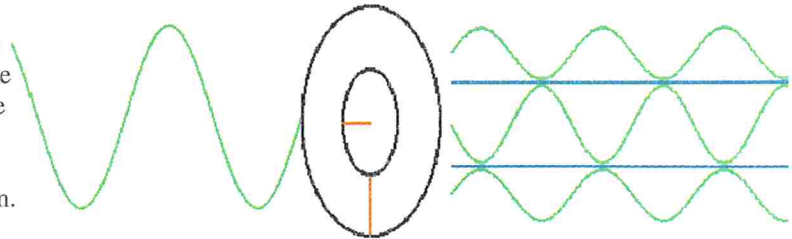
1.1 Chaloupka's Trap

Chaloupka et al. developed a ponderomotive-optical trap for free electrons [4]. This trap has a dark spot at the center of a laser focus. To the author's knowledge, it has never been used to trap either opaque or translucent macroscopic particles, but fits the description of a trap for such particles.

The Chaloupka trap is generated by cutting the center out of a half-wave plate, rotating it by 90° , and reinserting it. Light polarized parallel to the fast axis of the wave plate propagates half a wavelength further than light polarized perpendicular to the fast

axis. Placing this modified half-wave plate in a collimated laser causes the center of the laser beam to be phase-inverted relative to the outside as shown in figure 1.1. Focusing the laser after the half-wave plate will cause destructive interference to occur at the

Figure 1.1 Half-Wave Plate
Rotating the center of the plate by 90° causes the center of the laser going through it to be phase inverted relative to the outer portion of the laser beam.



focus [2] (see figure 1.2). This destructive interference results in the intensity profile depicted in figure 1.3. Cutting the half-wave plate at a radius of $0.825 w$, where w is the Gaussian beam radius at the wave plate, results in an optimized trap with the highest possible intensity saddles [2], [3].

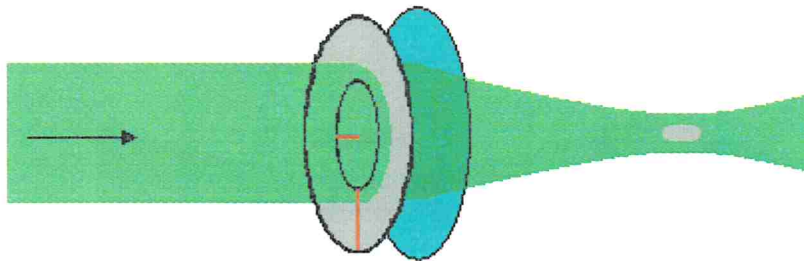


Figure 1.2 Making Chaloupka's Trap
This trap is made by running a collimated laser through a half-wave plate and a lens.

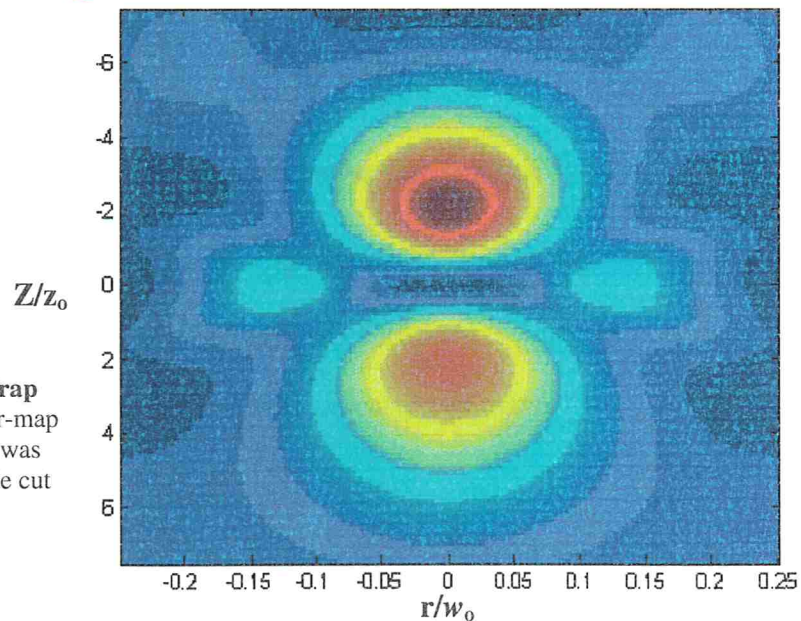


Figure 1.3 Chaloupka's Trap
Numerically generated color-map plot of laser intensity. This was made using a half-wave plate cut at a radius of $0.825 w$.

1.2 Poisson Trap

This past year, we developed a trap similar to Chaloupka's by imaging a small round obstacle in the laser focus. This trap also has not yet been used experimentally. The concept is presented for the first time here. We call it the Poisson trap because it resembles the famous experiment by Siméon D. Poisson [5]. When an obstacle is inserted in a laser, light washes in behind the obstacle and diffracts to create Poisson's spot. After the obstacle, the laser no longer has the information about what was before the obstacle. Therefore, when Poisson's spot is imaged by a lens, the diffraction effect is seen in both directions relative to the new focal plane. This leaves a dark focal center surrounded by high-intensity light. Figure 1.4 shows the optical configuration used to generate Poisson's trap.

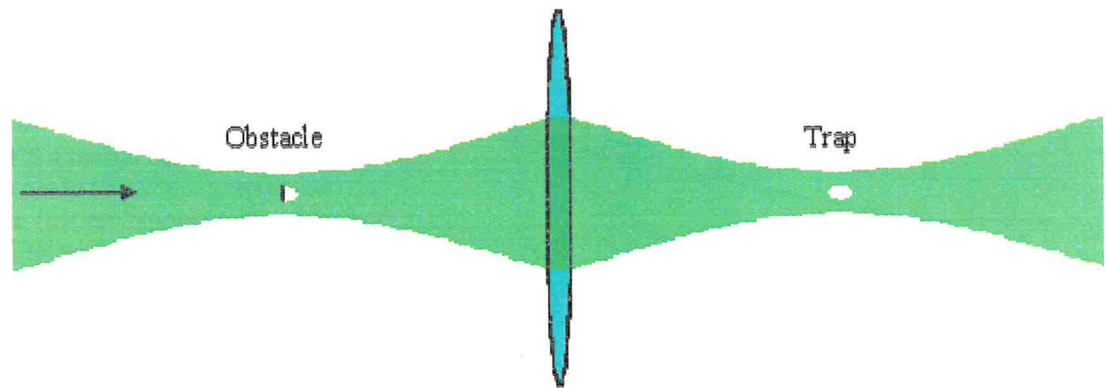


Figure 1.4 Optical Configuration for the Poisson Trap

This trap is discussed in more detail in Chapter 2. An analogy between quantum mechanics in two dimensions and laser propagation is discussed in Chapter 3. This analogy provides a way to perform “quantum mechanical experiments”.

1.3 Spherical Aberration Trap

The spherical aberration trap (SAT) is generated using a spherical lens. Most laser applications using spherical lenses rely on the paraxial approximation, the assumption that the laser is very close to the axis of the lens. When the laser radius is small relative to the radius of curvature of the lens, a spherical lens is effectively identical to a parabolic lens and the laser comes to a very distinct focus. If the radius of the laser beam is large, then the outside edge of the beam will focus at an appreciably different location than the center (see figure 1.5). In this situation, the laser beam develops several dark pockets caused by interference near the focus. These dark pockets constitute the SAT.

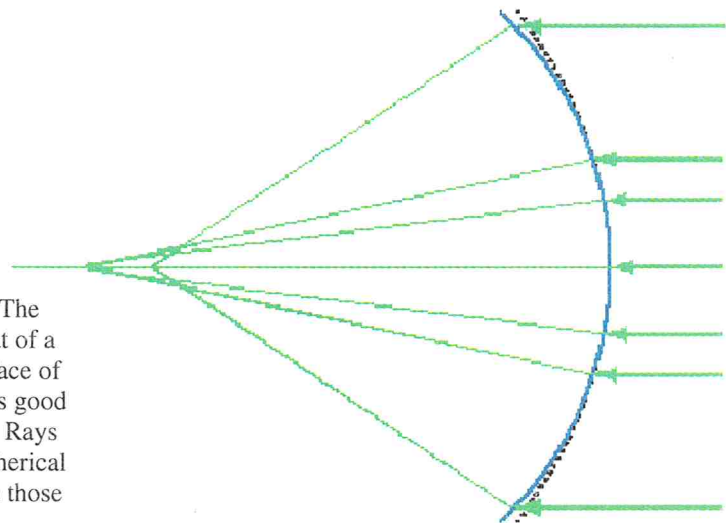
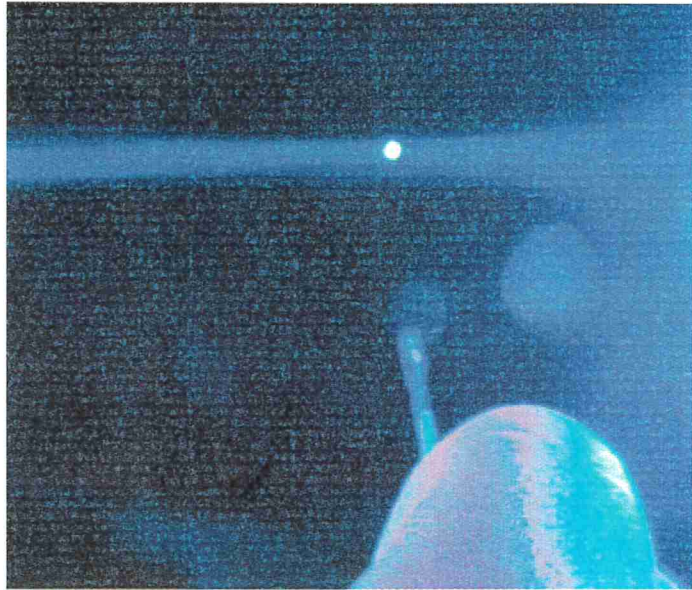


Figure 1.5 Paraxial Approximation The line of a circle (thick blue) versus that of a parabola (dotted) represents the surface of a lens. The paraxial approximation is good to the extent that these are the same. Rays traveling through the edges of the spherical lens focus at a different location than those near the center.

This thesis represents the first public disclosure of the SAT; it is discussed in more detail in chapter 4. This trap differs from the other two traps in that it has successfully trapped charcoal particles (see figure 1.6). The particles are about 5 microns in diameter, and the laser is a continuous argon ion laser with 200 mW. This is part of ongoing research at Brigham Young University. [6]

Figure 1.6
Charcoal particle in a SAT
 In the bottom right corner is a
 person's finger holding a
 sewing needle.



1.4 Diffraction Integrals

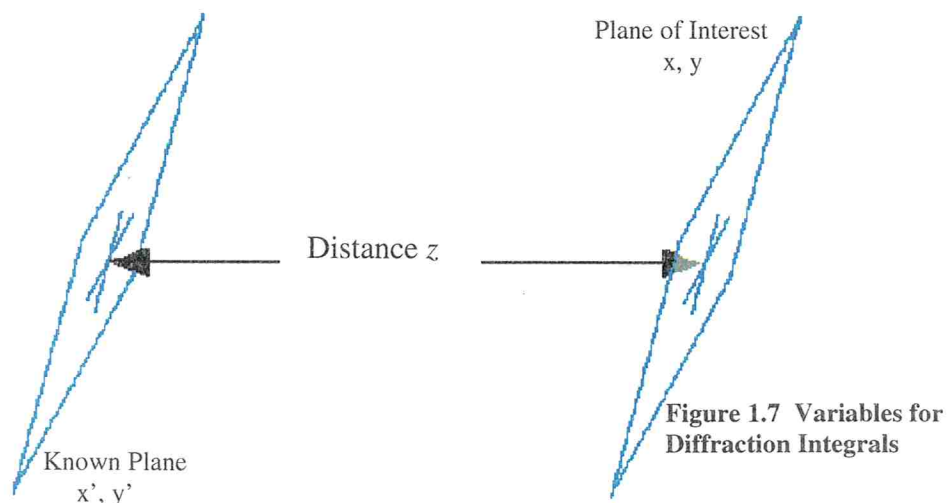
The propagation of Gaussian lasers through the center of a lens is well-known and easily modeled with a bit of algebra within the paraxial approximation [7]. When dealing with laser traps for opaque particles, the phase variations and discontinuities in intensity means that the diffraction integrals cannot be handled without numeric integration. The Fresnel-Kirchhoff diffraction integral treats light as point sources of spherical waves of electric fields that propagate and interfere at some point of interest. The full Fresnel-Kirchhoff diffraction integral is

$$\vec{E}(\vec{r}_{(x',y',x,y,z)}) = -\frac{i}{\lambda} \int_{\text{KnownPlane}} \vec{E}_{\text{known}} \frac{e^{ikr}}{r} \left[\frac{1 + \cos(\vec{r}, \hat{z})}{2} \right] d(x', y'), \quad (1.1)$$

where r is the magnitude of the vector from the location in the source of light to the point of interest. The cosine of the angle between r and the z -direction also plays a role, this is not a simple integral. To simplify it, Fresnel made the approximation that the plane of interest is significantly farther away from the known plane than any dimension in x or y . Figure 1.7 shows the coordinate system used for the diffraction integral. In the Fresnel approximation, the integral becomes

$$\bar{E}(\bar{r}_{(x',y',x,y,z)}) = -\frac{i}{\lambda} \int_{\text{KnownPlane}} \bar{E}_{\text{known}} \frac{e^{ik\left(\frac{x^2+y^2+x'^2+y'^2}{2z} - \frac{xx'+yy'}{z}\right)}}{z} [1] d(x',y'), \quad (1.2)$$

which is known as the Fresnel diffraction integral [3]. For many applications this is a good approximation. Generally, if the laser beam waist is less than one tenth the distance z , then the approximation is good.



To calculate the profile of the laser traps, it was necessary to find the light intensities at small distances z (on the order of the laser waist). This is accomplished by performing the Fresnel diffraction integral once with a large-negative value for z . The results of this integration are then used for future integrations to find the light intensity

near the known plane. The numerical accuracy of this method was tested by performing the integration from the $-z$ plane to the known plane for a simple Gaussian. All errors were verified at less than 2% at the known plane. Appendices A and B are comprised of Matlab code and discussion of the numerical methods used to characterize the laser traps.

Chapter 2: Poisson Trap

Characterizing the Poisson trap requires careful numerics. Taking out the center of a laser beam creates a hard edge in the known electric field. To determine the light intensity near the obstacle, or after a lens following the obstacle, requires exponentially more time and data points. It is useful to have a formula for the approximate size of the Poisson trap before performing the calculation. The features of interest in the Poisson trap are the locations of local maxima and minima in laser intensity, the radial extent, r_T , the intensity of the light at the trap boundaries, and the axial extent, Z_T . The variables Z_T and r_T are shown in figure 2.1. All of these characteristics are functions of the obstacle

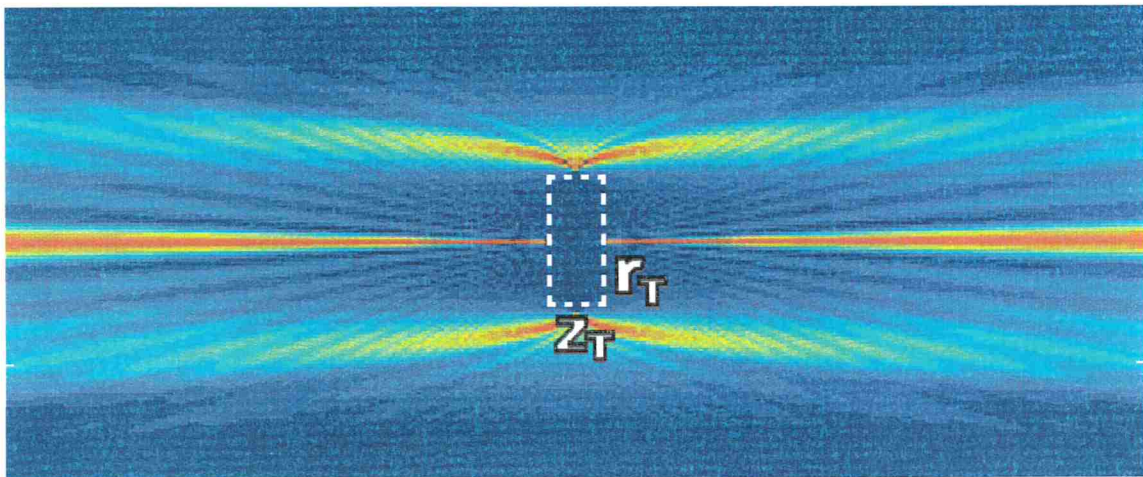


Figure 2.1 Poisson Trap This is a numerically generated color-scale plot of the laser intensity. The laser is propagating from left to right. The region of the trap is outlined with the dotted white box. The radial extent of the trap, r_T , and axial extent, Z_T , are labeled.

radius, r_b , the beam waist, w_0 , the wavelength of light, λ , and the focal length and placement of the lens, f and z_{Before} (see figure 2.2). For this investigation λ will be considered constant, and the obstacle will only be considered at the beam waist before the lens. Regardless of parameters used to construct a Poisson trap, the resulting intensity

profile appears similar to figure 2.1. The following discussion of the individual characteristics of the Poisson trap requires the definition of several variables. These variables are defined graphically in figure 2.2.

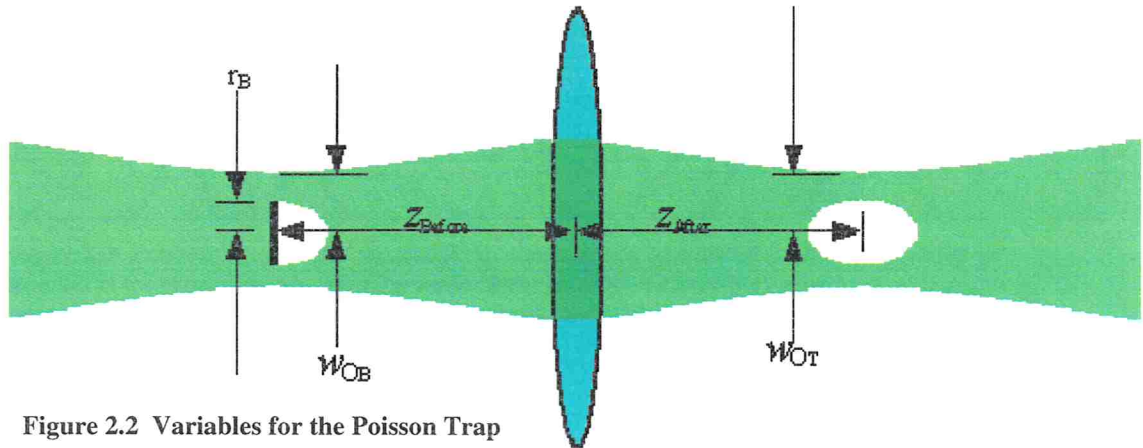


Figure 2.2 Variables for the Poisson Trap
For simplicity, Z_T and r_T are on figure 2.1.

2.1 Ring Location

The intensity profile, plotted as a function of r at various distances, Z_{After} , shows many small rings. Ring locations are a function of λ and r_B . Figure 2.2 demonstrates that it is independent of the beam waist. Studying the rings can be useful in determining r_B and the roundness of the obstacle. The ring locations are easily determined by the optical path difference from the outer edges of the obstacle to various points some distance away, just as is done in many introductory physics texts.

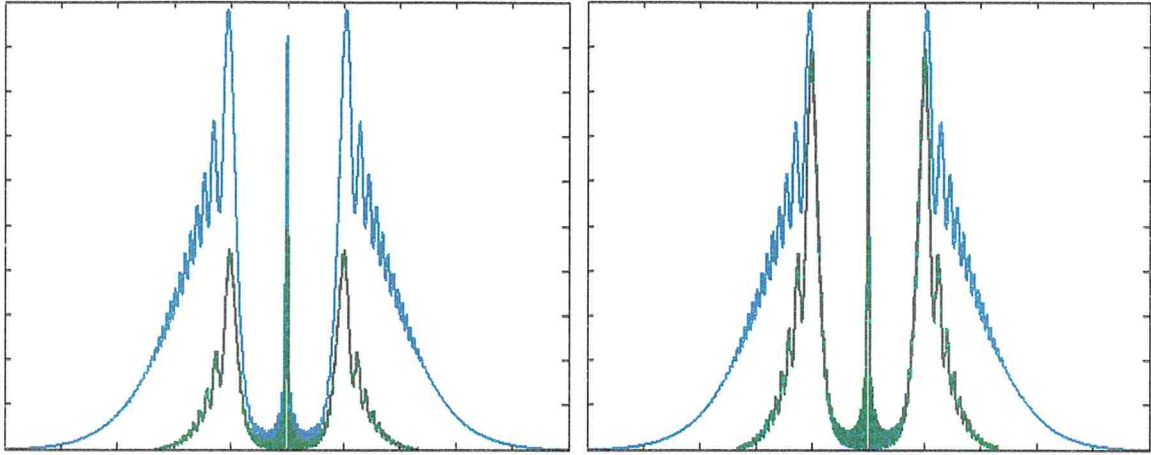


Figure 2.2 Poisson Trap with $1w_0$ and $2w_0$, numerically generated Left: both profiles are depicted at the same distance from the same obstacle. The one profile is generated with a laser of twice the waist radius as the other. Both are normalized to their own intensity at the unmodified waist center. Right: the same profiles normalized to the same maximum height to show the placement of rings.

2.2 Trap Radius

The radius of the trap is defined to be the distance from the center of the trap to the inner edge of the intense region. Any particle to be trapped in Poisson's trap must have a diameter that is larger than r_T but less than $2r_T$. The trap radius, r_T , is dependent on r_B , w_{OB} , and the waist of the laser focus if there were no trap, w_T . As stated by Eq 2.1, it is a simple ratio of proportionality:

$$\frac{r_T}{r_B} = \frac{w_{OT}}{w_{OB}} \quad (2.1)$$

All of the variables in Eq 2.1 are easily measured experimentally except for w_{OT} , which requires its own formulas,

$$\begin{aligned} Z_B &= \frac{\pi}{\lambda} w_{OB}^2 \\ Z_T &= \frac{\pi}{\lambda} w_{OT}^2 \end{aligned} \quad (2.2)$$

$$z_{After} - iZ_T = \frac{-f(z_{Before}^2 - fz_{Before} + Z_B^2)}{(z_{Before} - f)^2 + Z_B^2} - i \frac{f^2 Z_B}{(z_{Before} - f)^2 + Z_B^2}. \quad (2.3)$$

In Eq 2.2, the variables Z_B and Z_T are, by definition, the Rayleigh ranges before and after the lens. Eq 2.3 is obtained from the ABCD law [7].

2.3 Wall Intensity

The intensity of the laser at the trap boundaries, I_T , is directly proportional to the amount of force with which the trap can hold onto a particle. I_T is also directly proportional to the intensity of the laser at the edge of the obstacle, I_{OB} . I_{OB} is easily determined by assuming a Gaussian profile and placing r_{OB} in a standard formula for a Gaussian beam, where I_o is the intensity of the laser at the center of the beam:

$$I_{ob} = I_o e^{\frac{-r_o^2}{w_o^2}}. \quad (2.4)$$

The constant of proportionality, k_A , between I_{OB} and I_T is determined by conservation of energy. The power (energy/time) in a Gaussian laser is $\frac{\pi I_o^2}{4} w_o^2$. The energy at the focus before the lens must be equal to the energy at the focus after the lens, so that

$$k_A = \frac{I_T}{I_{ob}} = \frac{w_o^2}{w_T^2}. \quad (2.5)$$

We can write the ratio of the waists in terms of the quantities in Eq 2.2 and Eq 2.3 so that

$$k_A = \frac{(z_{Before} - f)^2 + \frac{\pi}{\lambda} w_o^2}{f^2} \quad \text{and} \quad I_T = k_A I_{ob}. \quad (2.6)$$

2.4 Trap Length

Let Z_T be defined specifically as the distance between points where the central spike is the same intensity as the primary outer ring. There is generally some discrepancy between ideal and experimentally measured distances Z_T , as in figure 2.3. Deriving an equation for Z_T analytically is difficult, but comparing numerical calculations from several configurations yields a very simple and consistent function,

$$Z_T = \frac{2\pi^2}{\lambda} w_T^2 r_B. \quad (2.7)$$

Variations between the experimental and ideal distances Z_T can be accounted for with variations between the ideal and experimental beam waists.

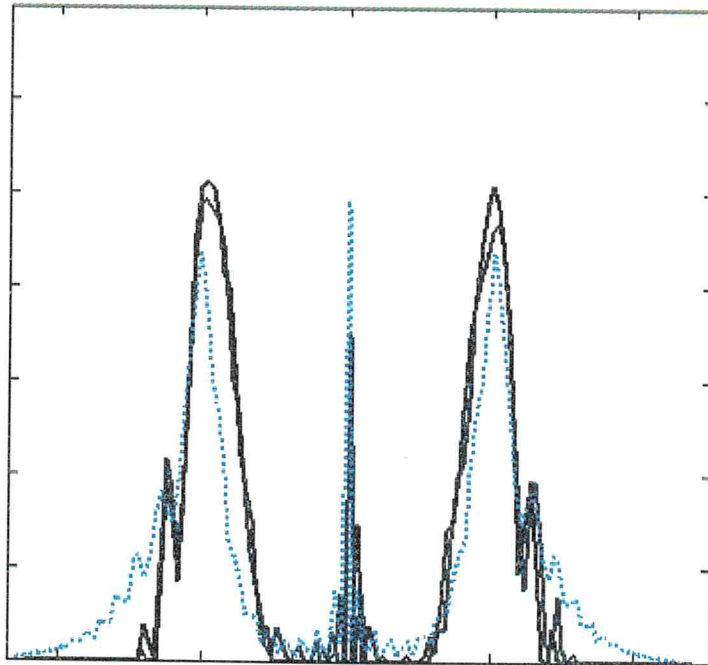


Figure 2.3 Experimental vs. Theoretical Profiles Near the Axial Boundary of a Poisson Trap, $z=0.5 Z_T$. The experimental profiles (solid lines) indicate that the location $0.5 Z_T$ has not yet been reached. The theoretical profile (dotted line) indicates that the same place has been passed. The profiles were normalized by conservation of energy with no correction for the dark-level.

2.5 Implementing a Poisson Trap

With these relations for Z_T , r_T , and I_T , it becomes possible to design a Poisson trap for a specific application without doing numeric integration or further experimental guess-work. It should be a direct matter to make a Poisson trap and use it to trap an opaque or reflective particle. It is necessary, that the particle be between one and two times r_T so that the central spike will keep it from rolling out.

Chapter 3: Laser Propagation and Quantum Mechanics

An analogy between the Schrödinger wave equation and laser propagation allows one to visualize the evolution of quantum mechanical wave packets. After drawing the analogy, the propagation of a wave after an obstacle will be considered in terms of a centrifugal force in wave mechanics.

3.1 Making the Correlation

The quantum mechanical wave equation constrained to two dimensions is

$$i\hbar \frac{\partial \Psi}{\partial t} = \frac{-\hbar^2}{2m} \nabla_{\perp}^2 \Psi, \quad (3.1)$$

where m is the particle mass, Ψ is the wave probability function, t is time, and \hbar is Planck's constant divided by 2π . Here, ∇_{\perp}^2 refers to the 2-dimensional laplacian in the x and y dimensions. The scalar electromagnetic wave equation is

$$\nabla^2 E = \frac{1}{c^2} \frac{\partial^2 E}{\partial t^2}, \quad (3.2)$$

where c is the speed of light and E is the magnitude of the electric field. [8] If we assume a planewave-like solution such that

$$E = E(x, y, z)e^{i(kz - \omega t)}, \quad (3.3)$$

then $\frac{\partial^2 E}{\partial t^2} = -\omega^2 E$ and $\nabla^2 E = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) E + \frac{\partial^2 E}{\partial z^2}$. The second partial derivative with

respect to z becomes

$$\frac{\partial^2 E}{\partial z^2} = 2ike^{i(kz - \omega t)} \frac{\partial E(x, y, z)}{\partial z} + e^{i(kz - \omega t)} \frac{\partial^2 E(x, y, z)}{\partial z^2} - k^2 E(x, y, z)e^{i(kz - \omega t)} \quad (3.4)$$

Assuming that the function $E(x,y,z)$ changes much faster in x and y than it does in z , we can neglect the second partial of $E(x,y,z)$ with respect to z . This is effectively the paraxial approximation. Plugging these results into Eq. 3.2 and canceling the exponential terms gives

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) E(x, y, z) + 2ik \frac{\partial E(x, y, z)}{\partial z} - k^2 E(x, y, z) = \frac{-\omega^2}{c^2} E(x, y, z). \quad (3.5)$$

But, since $k^2 = \frac{\omega^2}{c^2}$, this simplifies even further to

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) E(x, y, z) + 2ik \frac{\partial E(x, y, z)}{\partial z} = 0. \quad (3.6)$$

This can be rearranged to look more like the quantum mechanical wave equation in two dimensions as follows

$$i \frac{\partial E(x, y, z)}{\partial z} = -\frac{1}{2k} \nabla_{\perp}^2 E(x, y, z). \quad (3.7)$$

Here, the electric field takes the place of the probability wave so that the probability density is replaced by light intensity. Similarly, time is replaced by distance of laser propagation, z . With these connections, we see that a laser propagates effectively the same way as a two dimensional quantum mechanical wave changes in time.

3.2 Negative Centrifugal Force

The laser as it passes an obstacle at its focus resembles a doughnut probability wave; see figure 3.1. As Poisson's spot forms, some of the light diffracts in towards the middle. The average radial distance of light intensity decreases before eventually increasing. This corresponds to negative centrifugal force, which is the proposition of a quantum mechanics paper pending publication [9]. Matter, confined to two dimensions,

starting in a doughnut shape rushes in before eventually spreading out, as seen in figure 3.2.

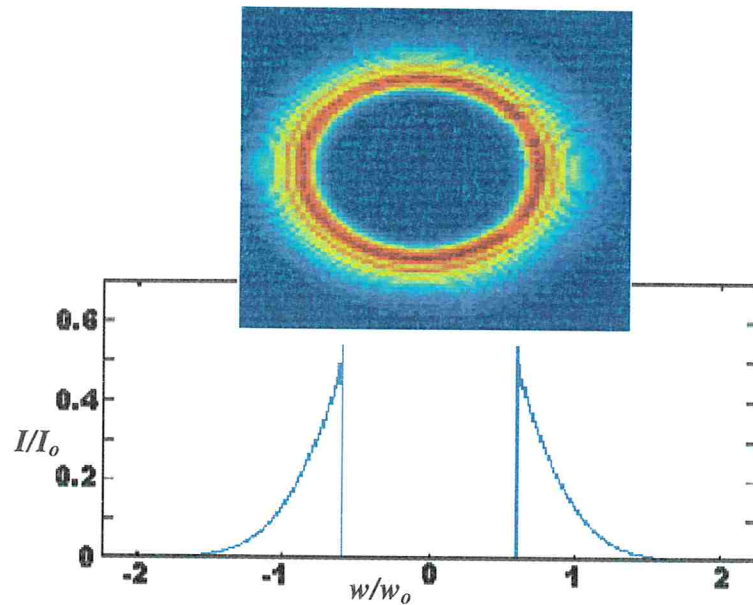


Figure 3.1 Light Doughnut (top) Picture taken of a 1.33 mm waist HeNe laser at 3.2 cm after a 0.79 mm obstacle. Poisson's spot has not had time to fill in yet. (bottom) Numerically generated plot of the laser intensity normalized to the central intensity and scaled to the beam waist. The two pictures are aligned to show the fit of the internal radius.

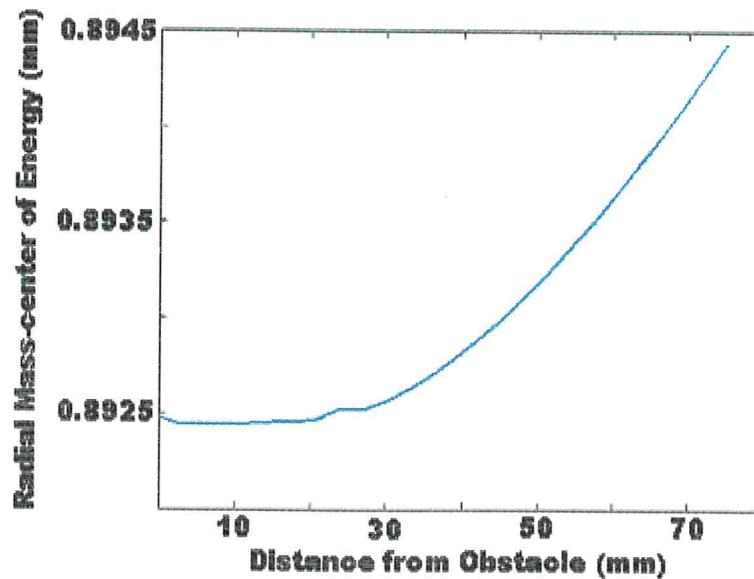
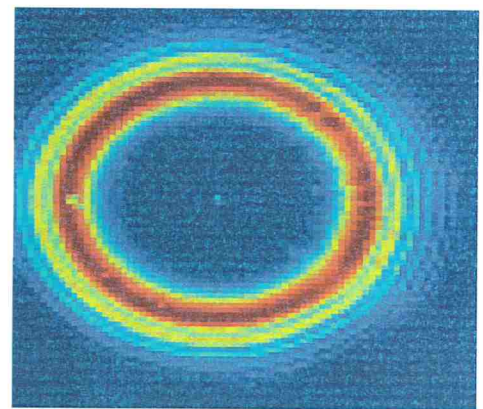
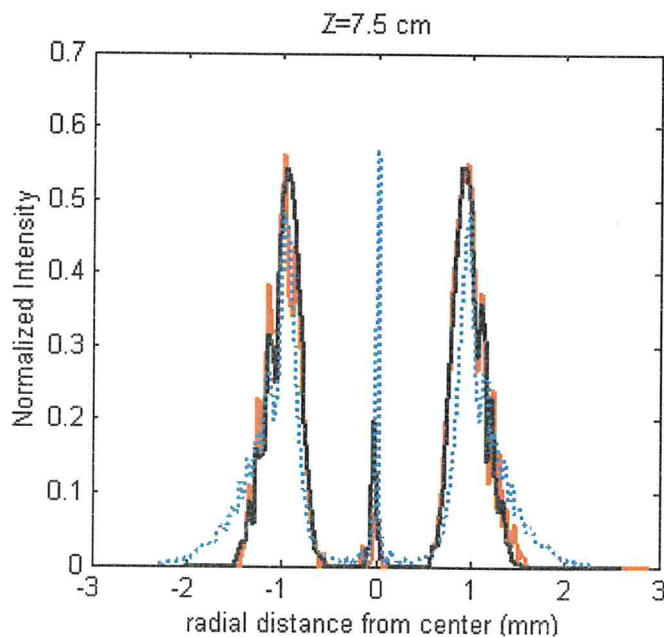
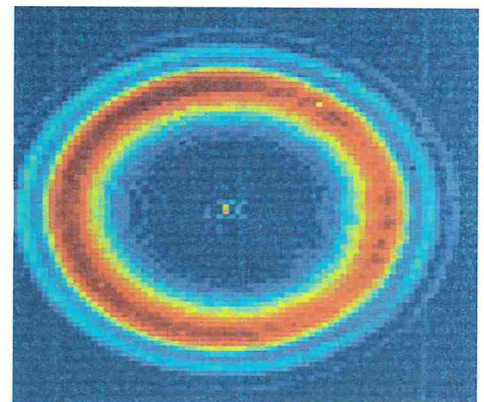
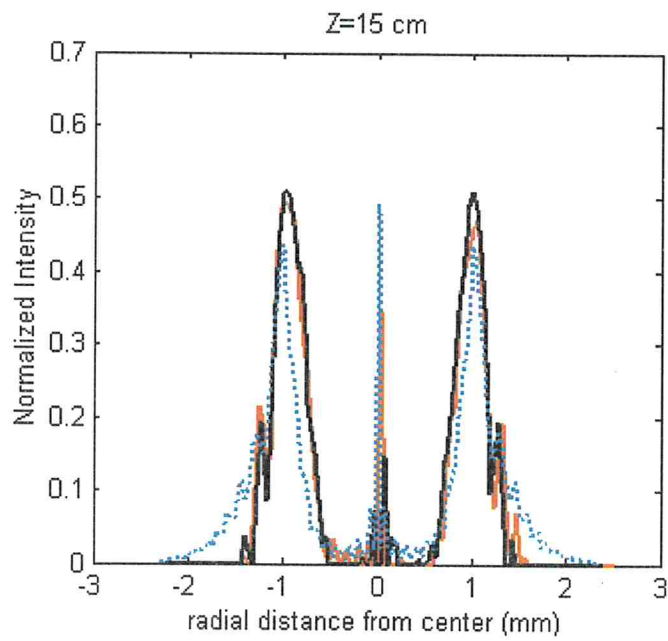
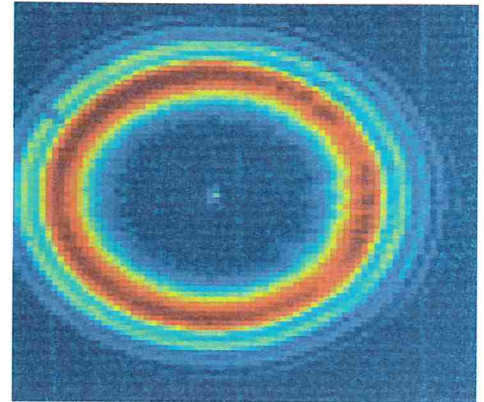
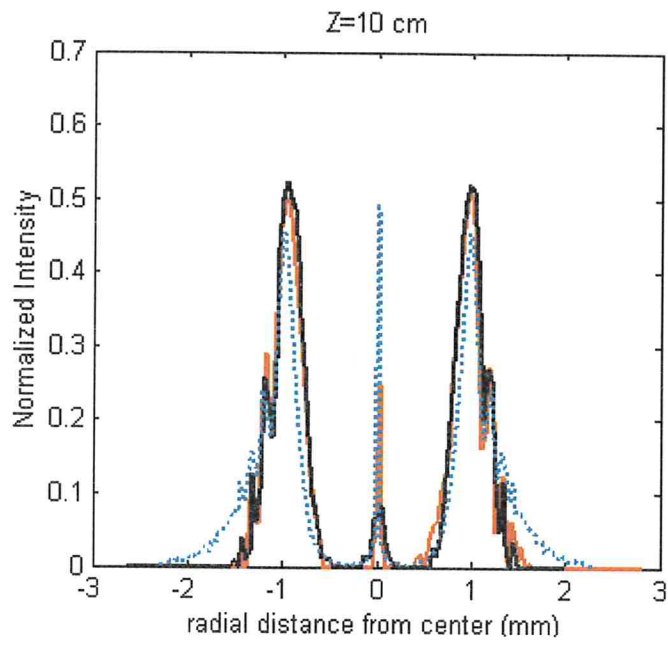


Figure 3.2 Radial, Mass-center of Energy
When starting with a doughnut shaped laser profile (see figure 3.1), the laser goes in before it goes out. This corresponds to negative centrifugal force in quantum mechanics.

3.3 Experimental and Numerical Data from Poisson's Spot

All of the figures in this section are experimentally or numerically from a HeNe laser ($\lambda = 633\text{nm}$) with a waist of 1.33 mm. A 0.79 mm radius obstacle was placed at the waist, and pictures were taken with a CCD camera down stream from the obstacle. Each figure is labeled with the physical distance from the obstacle; no adjustments were made for the optical path difference caused by filters. The black and red lines are perpendicular profiles from the experimental data. The dotted line is the numerical profile. The numerical data is normalized to the maximum intensity at the focus. The experimental data is scaled to fit the numeric data according to the areas under the curves, this follows conservation of energy. No adjustment was made for the dark-level. The pseudo-color plots are from the CCD and correspond to the adjacent profiles.





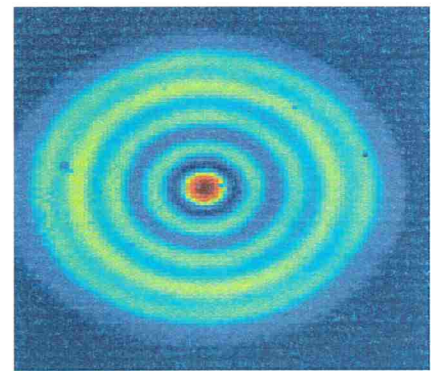
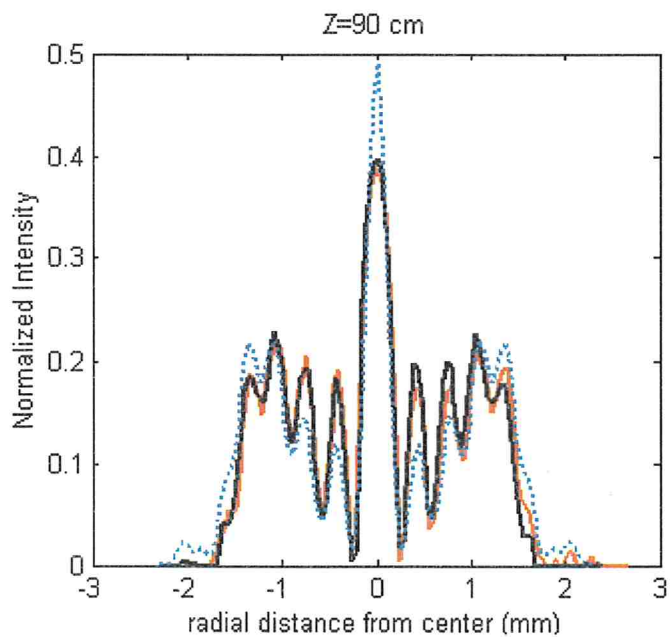
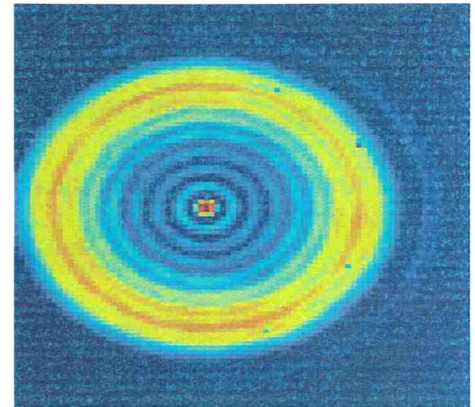
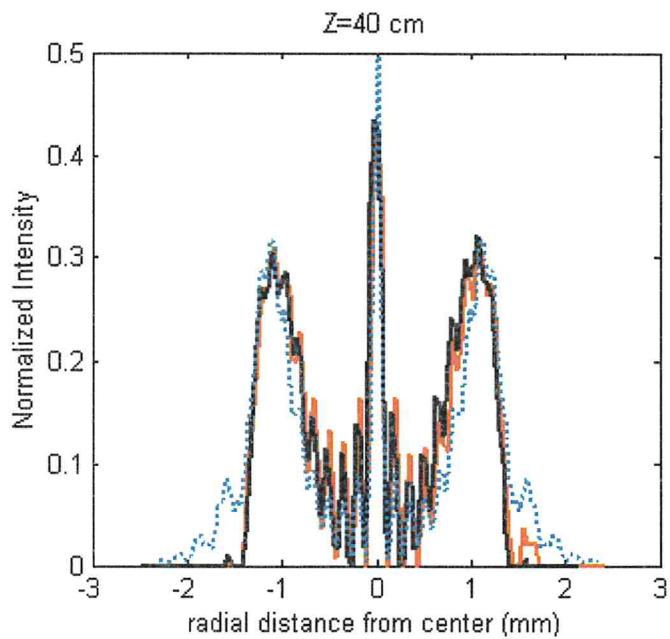


Figure 3.3 Experimental and Numeric Profiles of Poisson's Spot. Images are pseudo-color pictures of Poisson's spot at various distances Z from the obstacle. To their left are line profiles at the same distance. The solid red and black lines are from the experimental data, while the dotted line is the numeric profile. The numeric profiles are normalized, while the experimental ones are scaled using conservation of energy. No adjustments were made for the dark level of the CCD camera used to take the images.

Chapter 4: Spherical Aberration Trap

The spherical aberration trap (SAT) is new and not yet well understood. By comparing experimental results with numerical calculations, it seems apparent that it is spherical aberration that creates the trap; an infinity-corrected lens does not trap particles. So far, only one configuration has been experimentally successful in creating an SAT. Modifications on that configuration appear to weaken the trap. The configuration used for numerical calculations, only slightly different than the experimental configuration, is depicted in figure 4.1.

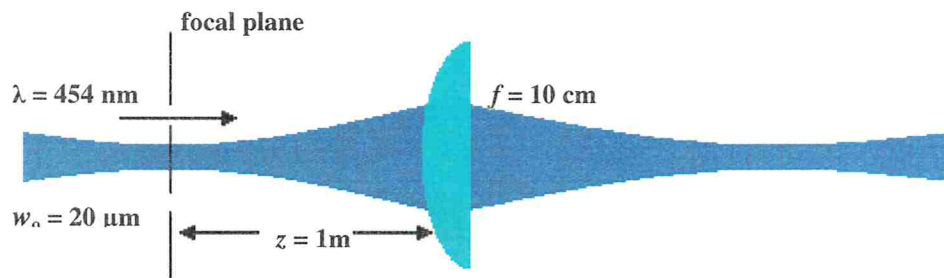


Figure 4.1 Optical Configuration for SAT

4.1 Numerical Method for Modeling the SAT

The numerical method for modeling the SAT is as follows. First, a perfect Gaussian laser is propagated to the plane of the lens following [7]

$$E(r, z) = \frac{-ikE_0}{2z \left[\frac{1}{w_0^2} - i \frac{k}{2z} \right]} e^{i \left(kz + \frac{kr^2}{2z} \right)} e^{\frac{-(kp)^2}{4z^2 \left[\frac{1}{w_0^2} - i \frac{k}{2z} \right]}} \quad (4.1)$$

Then the electric field is modified by the equation for a spherical lens, derived in the next section, which makes a radius-dependant phase modification on the existing electric field. Finally, the diffraction integral is performed on the modified electric field for each point near the focus (section 1.4 and appendix A). Using this method with the configuration depicted in figure 4.1 yields the intensity profile shown in figure 4.2. It has a very regular pattern of interference peaks approaching a primary focus in rows. The location of peaks alternates between rows so that a particle can be trapped between peaks, on each rows. The primary focus occurs axially sooner and is larger axially while smaller radially for a spherical lens than for a parabolic one. Figure 4.3 shows the intensity along the laser axis normalized to the maximum intensity a parabolic lens would have produced.

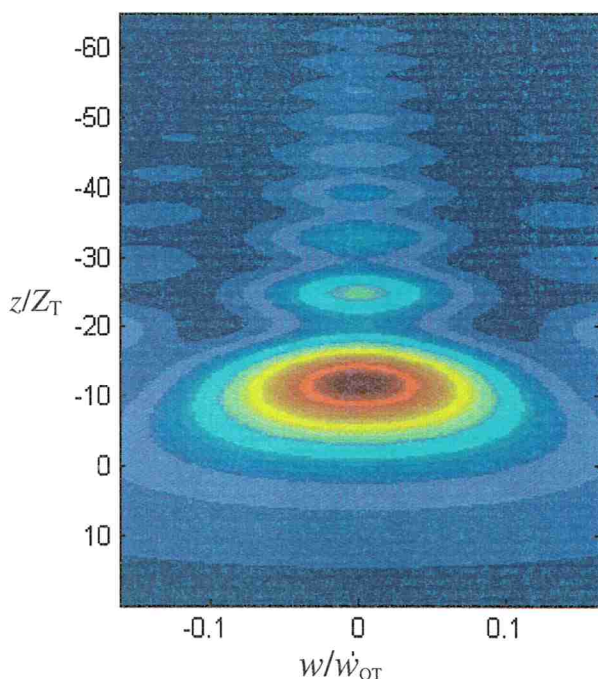


Figure 4.2 Spherical Aberration Trap

This is a numerically generated color-scale plot of laser intensity. It is scaled relative to the rayleigh range, Z_T and waist, w_{OT} , that would result from a parabolic lens.

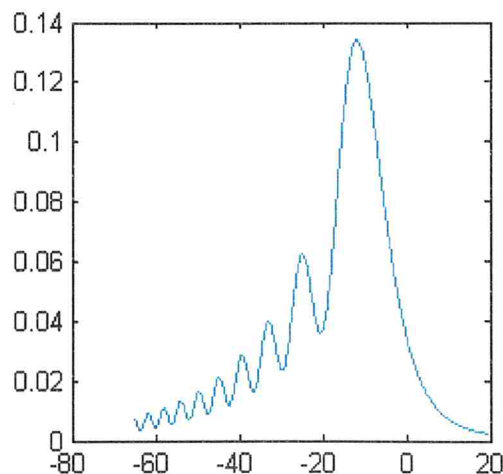


Figure 4.3 On-axis Intensity. The intensity is normalized to the maximum intensity that a parabolic lens would have produced. It is plotted according to distance from the focus using the same scale as figure 4.2.

4.2 Derivation of the Spherical Lens

The formula for a spherical lens is derived in the same manner as for a parabolic lens [7]. Ignoring the light's angle of incidence on the lens surface, a thin lens can be treated as a phase shift, $\Delta\phi$, at a plane. Thus, the mathematical effect of the lens is

$E_{afterLens} = e^{i\Delta\phi} E_{beforeLens}$. This phase shift is approximated as the optical path difference between rays parallel to the z-axis as they pass through the lens at various radii; see figure 4.4. This is essentially the paraxial-approximation, but the light is not necessarily treated as being close to the z-axis, just parallel.

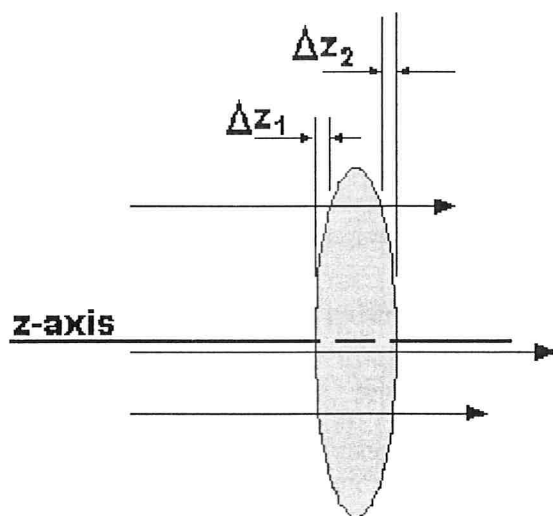


Figure 4.4 Axis-Parallel Rays.
Each ray travels a different optical path length. The relative difference is determined by how much of the lens it does not go through.

The relative phase shift of each ray is determined by how much lens material it does *not* go through when compared with the middle, Δz . The relative phase shift is,

$$\Delta\phi = -k(n-1)(\Delta z_1 + \Delta z_2) \quad (4.2)$$

where k is the wave number. The term $(n-1)$ is the difference in index of refraction

between the lens, n , and the surrounding. If each side of the lens is a portion of a sphere with radius of curvature, R , then the distance Δz is defined by

$$R^2 = r^2 + (R \pm \Delta z)^2 \quad (4.3)$$

where r is the radial distance from the z -axis. This is depicted in Figure 4.5.

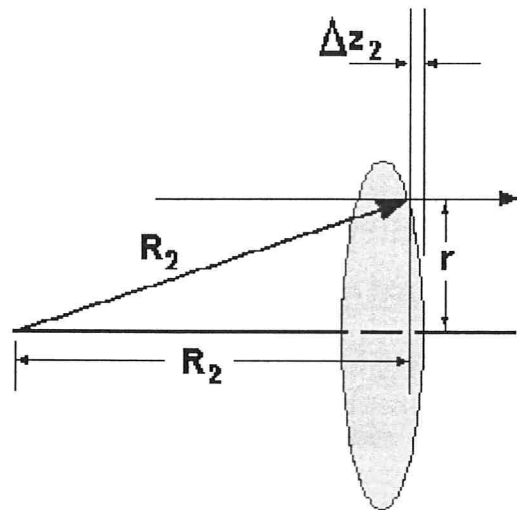


Figure 4.5 Distances used to determine a function for dz , the path difference through the lens.

The sign on Δz depends on which side of the lens the equation is being written for. By convention, R_2 is negative which forces Δz_2 to be positive. On the other side, the radius of curvature, R_1 , is defined to be positive which makes Δz_1 negative. Solving Eq 4.3 for Δz yields

$$\begin{aligned} \Delta z_1 &= R_1 - \sqrt{R_1^2 - r^2} \\ \Delta z_2 &= R_2 + \sqrt{R_2^2 - r^2} \end{aligned} \quad (4.4)$$

The square root is somewhat troubling. To simplify things, we divide out R and make a Taylor series expansion of the square root to get

$$\Delta z_1 = R_1 - \left(R_1 - \frac{r^2}{2R_1} - \frac{r^4}{8R_1^3} - \dots \right). \quad (4.5)$$

Doing the same process for Δz_2 , truncating both series to include only the terms shown, and substituting them into Eq 4.2 yields

$$\Delta\phi = -k(n-1) \left[\frac{r^2}{2} \left(\frac{1}{R_1} - \frac{1}{R_2} \right) + \frac{r^4}{8} \left(\frac{1}{R_1^3} - \frac{1}{R_2^3} \right) \right]. \quad (4.6)$$

The perfect parabolic lens ignores the R^3 terms and substitutes in the lens maker's formula $\frac{1}{f} = (n-1) \left(\frac{1}{R_1} - \frac{1}{R_2} \right)$ to get $\Delta\phi = -\frac{kr^2}{2f}$. The R^3 terms represent the spherical aberration and need to be kept in this case, which means that there is not a simple substitution with f that will make the expression "look pretty." Eq 4.6 can be simplified for specific kinds of lenses. In the case of a plano-convex lens R_2 is *infinity*. In the case of a convex-convex symmetric lens, R_1 is $-R_2$. For these cases Eq 3.5 simplifies to

$$\begin{aligned} \Delta\phi &= -\frac{kr^2}{2f} - \frac{kr^4}{2f^3} \quad (\text{plano-convex}) \\ \Delta\phi &= -\frac{kr^2}{2f} - \frac{kr^4}{8f^3} \quad (\text{convex-convex}) \end{aligned} \quad (4.7)$$

with the added assumption that the index of refraction of the lens, n , is 1.5. It is worthwhile to note that there is approximately four times the spherical aberration in a plano-convex lens than in a convex-convex symmetric lens.

4.3 Conclusion on SAT

This is a unique and interesting new trap. Experimentally, there are several distinct locations where a particle can be trapped, which matches the modeling. However, more detailed methods need to be used to obtain a more complete

understanding of the trap. The most tenacious simplification in this model is ignoring the light's angle of incidence on the lens surface. As spherical aberration becomes more pronounced, the angle of incidence becomes more significant. In the case presented here, the angle of the lens surface has changed by 8.279° between the center and the incident beam waist; the radius of curvature of the beam arriving at the lens is 1 m.

Chapter 5: Conclusion

Of the three traps discussed, the Poisson trap is best prepared for experimental use. Chaloupka's trap, while clever, requires the somewhat difficult modification of a half-wave plate. The resulting trap does not have very sharp intensity feature (see figure 5.1). This is contrary to expectation since none of the laser is blocked in the process of making Chaloupka's trap. The SAT is not well characterized at this time, but further development could prove it to be far more effective than the other two. The Poisson trap is theoretically well developed and ready for use, and may prove advantageous for particles larger than 20 microns. The wide opening between the center spike and the primary maximum limits the size of particles that can be trapped for a given Poisson trap.

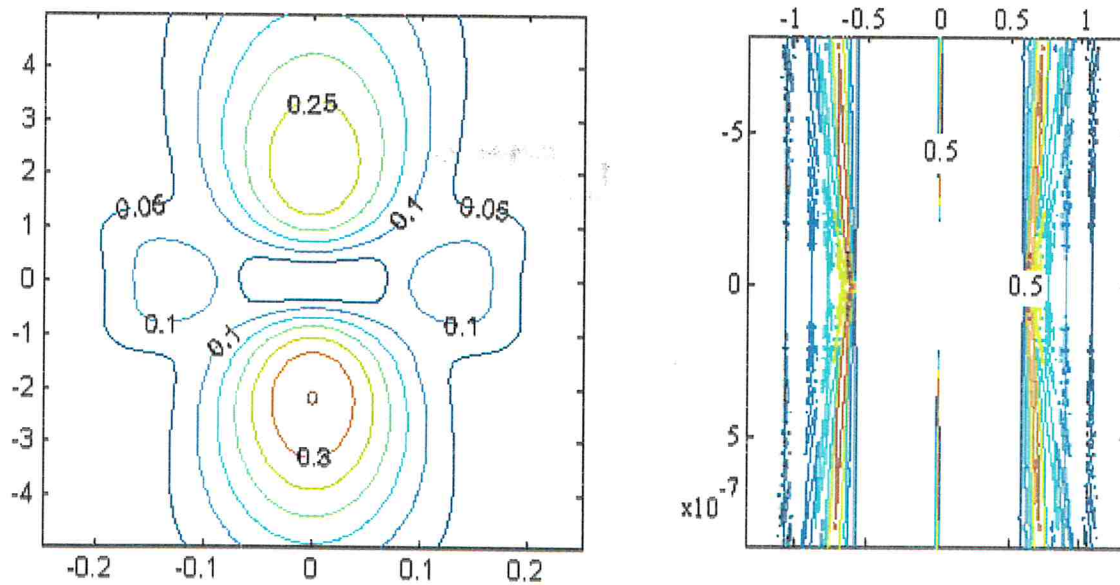


Figure 5.1 Contour Plots of Chaloupka and Poisson. Both plots are scaled to their rayleigh range vertically and beam waist horizontally. The intensities are labeled at various points, and normalized to the maximum intensity the focus would have if the beam were not modified to create the trap. Left: Chaloupka's trap is the optimized design. Right: Poisson Trap uses one-to-one imaging with an obstacle radius that is 59% of the beam waist.

References

- [1] W. H. Wright, G. J. Sonek, and M. W. Berns, "Parametric study of the forces on microspheres held by optical tweezers," *Applied Optics* **33**, 1735-1748 (1994).
- [2] Kenneth S. Anderson, "Single-Beam Optical Trapping of Micron-Sized Particles," Senior Thesis, Brigham Young University (1999)
- [3] Justin Peatross, "Physics of Light and Optics," On-line text 2001, Chapter 10, *optics.byu.edu*.
- [4] J. L. Chaloupka, Y. Fisher, T. J. Kessler, and D. D. Meyerhofer, "Single-beam, ponderomotive-optical trap for free electrons and neutral atoms," *Optics Letters* **22**, 1021-1023 (1997)
- [5] Eugene Hecht, *Optics*, 3rd ed., (Addison Wesley Longman, Inc., Reading, Massachusetts, 1998) pp 485-487
- [6] Cody Bliss and Justin Peatross, Department of Physics and Astronomy at Brigham Young University (continuing research)
- [7] Justin Peatross, "Physics of Light and Optics," On-line text 2001, Chapter 11, *optics.byu.edu*
- [8] Eugene Hecht, *Optics*, 3rd ed. (Addison Wesley Longman, Inc., Reading, Massachusetts, 1998) pg 44
- [9] Mikhael Federov (Private communication).

Appendix A Programs for Numerical Profiling

All of the numeric data was generated using Matlab code for this thesis. At least 15 programs and functions were written to test numerical methods, profile the laser, and evaluate the profiles generated. Not all of that code is included here. Some discussion of numerical methods is included in appendix B, and none of the code designed to test the numeric methods is included here. Each of the three traps has a unique optical configuration and application of the numerical methods that minimizes the computation time. However, each one also uses the same functions and was written by modifying a copy of the program for the trap before it. Only the code written to find the radial mass-center of energy for the Poisson trap is included here with all of its functions. The radial mass-center of energy of Poisson's trap is the most difficult to work with because of the sharp edge in the laser profile that needs to be propagated very small distances. This pushed the numerical limits of the techniques and computer used. Table A.1 shows all of the programs with their functions that are listed below.

PoissonTrap2	intPoly4	Shuffle
		intFresnel
	poly10	
	mirror	
	mirrorN	

Table A.1 Programs
Listed with their sub-functions to the right

A.1 PoissonTrap2.m

```
%PoissonTrap2.m
%By Benjamin Bellville, BYU 2002
%
%This is designed to do the fresnel integral for the experimental situation we did for the
%Russians. It is similar to PoissonTrap.m, but it steps behind the focus first so that
%the fresnel approximation doesn't get us in trouble. It takes about 60 times longer.
%
%There was a 1.56mm diameter ballbearing placed near the focus of a 1.31mm radius HeNe laser.
```

```

%
%It uses the Fresnel diffraction integral in cylindrical coordinates.
%See chapters 10 and 11 of "Physics of Light and Optics" by Dr. Justin Peatross 2002.
%It is available on the web at "www.optics.byu.edu"
%
%It uses two simultaneous iterations of Simpson's 1/3 integration
%And does simultaneous error estimate for the first stage of integration, AKA Boole's rule
%
%See Also intPoly4.m mirror.m mirrorN.m simpsons.m poly10.m
%

clear
tic
%Set variables in terms of the wavelength
%All measurements are done in microns
L=.633; %wavelength
k=2*pi/L; %wave number
w = 1330*k; %beam waist
zz=w^2/2; %rayleigh-range.
z=-zz/12; %distance before the lens that we want to step.
Eo=1; %normalized amplitude

a = 790*k; %obstacle diameter, Limit of integration
b= 3e4*k;

%Since integration is additive, we will subtract the integral for what the ball-bearing takes
%out from the analytical solution.
rhosteps=1000;
rhomin=0; %limits of integration
rhomax=a;
rhostep=(rhomax-rhomin)/rhosteps;
rho=rhomin:rhostep:rhomax;

%r represents the variable of radial location at the early plane.
rmin=0;
rmax=b;
%here begins the process of breaking up the vector of the variable of integration
%so that we can have different step sizes where the function is more oscillatory.
rslices=5;
rslice=rmin:(rmax-rmin)/(rslices):rmax;
rsteps=[4000,8000,12000,16000,20000]; %these must be divisible by 4 for this to work.
r=1;
y(1)=0;
for x=1:rslices
    rmin=rslice(x);
    rmax=rslice(x+1);
    rstep(x)=(rmax-rmin)/rsteps(x); %set step size
    rtemp=rmin:rstep(x):rmax; %The radius at the plane of integration
    r=cat(2,r,rtemp); %compile it as one vector

```

```

    y(x+1)=length(r)-1; %carries the information on how to break up the r-vector again.
end
r=r(2:length(r));
%done dividing it all up
clear rtemp rstep x rmax rmin

%create the array for the E-field that goes into integration
Ein=exp(-rho.^2/w^2);
Emid=intPoly4(rho,Ein,z,r);
errorMid=Emid(1,:);
Ein=-i/z/2*exp(i*z)/(1/w^2-i/z/2)*exp(i*r.^2/z/2).*...
    exp(-(r/z).^2/4/(1/w^2-i/z/2));
Emid=Ein-Emid(2,:);

figure(1)
subplot(2,1,1)
plot(r,real(Emid))
subplot(2,1,2)
plot(r/k,real(errorMid))
pause(1)
time(1)=toc
%save hopeful

%Set variables where the Efield is desired.
Rsteps=1800;
Rmin=0*k;
%Rmax is chosen so that the graph's scale will
%match the spiricon screen at maximum display size.
%4665, 23325, 1166.2 microns are the display sizes available
Rmax=5e3*k; %this is 5mm
Rstep=(Rmax-Rmin)/Rsteps;
R=Rmin:Rstep:Rmax;

%the Z-vector matches places where we took the pictures of the data.
Z=0:3:75; %milimeters
Z1=(Z)*k*1e3-z; %This is the distance from the ballbearing

for n=1:length(Z)
    Traptemp=0;
    for m=1:rslices

Traptemp=Traptemp+intPoly4(r(1+y(n):y(m+1)),Emid(1+y(m):y(m+1)),Z1(n),R);
    end
    Trap(n,:)=Traptemp(2,:);
    errorT(n,:)=Traptemp(1,:);
end
clear Traptemp m n Rsteps

```

```

%Find the mass center of energy
TrapI=Trap.*conj(Trap);%We need to use the intensity
simp=poly10(R); %array of coefficients for simpsons 1/3 integration, or poly10 integration
temp1=0;
temp2=0;
for k1=1:length(Z)
    intvar=simp.*R*Rstep;
    temp1=sum(intvar.*TrapI(k1,:).*R);
    temp2=sum(intvar.*TrapI(k1,:));
    masscenter(k1)=temp1/temp2;
end
masscenter=masscenter/w;

%mirror vectors so that they look like the line profile from the Spiricon
TrapI=mirror(TrapI);
errorT=abs(errorT);
errorT=mirror(errorT);
R1=mirrorN(R)/w; %this is in mm
Z=Z/zz;
for nn=1:length(R)
    if R(nn)<a
        TrapCompare(nn)=0;
    else
        TrapCompare(nn)=exp(-R(nn)^2/w^2);
        TrapCompare(nn)=TrapCompare(nn).*conj(TrapCompare(nn));
    end
end
TrapCompare=mirror(TrapCompare);
figure(2)
subplot(2,1,1)
title('Numeric vs Ideal')
plot(R1,TrapCompare,R1,TrapI(1,:))
subplot(2,1,2)
plot(R1,TrapCompare-TrapI(1,:));
xlabel('error in the model')

figure(3)
imagesc(R1,Z,TrapI);
title('The Wash-in After an Obstacle')
xlabel('Radius in mm')
ylabel('Distance from obstacle in mm')

figure(4)
plot(Z,masscenter)
title('Radial Mass Center of Energy in mm')
xlabel('Distance from focus in mm')

```

```

%clean up extra variables and save the calculations
clear rhostep intvar k1 simp temp1 temp2 name Z1 Z2 Rstep rho Ein

time(2)=toc
save poissontrap2
break

```

A.2 intPoly4.m

```

%intPoly4.m
%By Benjamin Bellville, BYU 2002
%
%This function is designed to work as a parent to intFresnel.m, or another integration technique
%The child function must be a left side rectangular integration technique excluding the
%factor of the step size.
%
%It uses a fourth order polynomial fit to do the integration just like simpson's 1/3 rule uses
%a second order polynomial fit. It has the advantage of making an error estimate at the same
%time it does the integration with virtually no cost in computation time. The integration
%technique is sixth order. The error estimate is for a fourth order error. The fourth order
%error estimate is used partly because real error tends to be much larger than sixth order
%error estimate.
%
%call: intPoly4(rho,Ein,z,r)
%
%rho= vector containing the variable of integration
%Ein= vector containing the Efield at the plane of integration
%z= distance from the plane of integration to the plane where the Efield is desired
%r= radial locations for the desired Efield.
%
%Vectors rho and Ein must be of (length-1) evenly divisible by 4. For example, the length could
%be 21: (21-1)/4=5 but not 22: (22-1)/4=5.2. If vectors rho and Ein do not meet this length
%requirement there is no return error. The integration will simply fail.
%
%This returns a 2 by n matrix. The first row contains the error estimates for integration
%at locations (z,r). The second row contains the results of integration for locations (z,r).
%
%see also shuffle.m intFresnel.m
%For more details on the numeric technique,
%see appendix B of Benjamin Bellville's senior thesis, BYU 2002

function[result] = intPoly4(rho,Ein,z,r)
%This integration technique creates 4 vectors of values. Each vector is integrated individually
%using a left side rectangular integration technique. The results are then combined to find the
%error estimate and the sixth order integration results. The function shuffle.m reorganizes the
%input vectors for this function. To change the child integration technique used by intPoly4.m,
%put the new child's integration call on lines 59 through 62 of this function. Also be sure to
%change the names of functions or variables passed into intPoly4.m as appropriate.

%Set vectors rho
rhostep=rho(2)-rho(1);%find step size, delta-rho
dims=(length(rho)-1)/4;
rho=shuffle(rho);

```

```

Ein=shuffle(Ein);

%Deriving the formula gives four logical rho vectors plus the ends,
%but vectors 1 and 3 behave identically, they are combined.
rhoab=rho(1:2);
Eab=Ein(1:2);

rho1=rho(3:2*dims+2);
E1=Ein(3:2*dims+2);

rho2=rho(2*dims+3:3*dims+2);
E2=Ein(2*dims+3:3*dims+2);

rho4=rho((3*dims+3):(4*dims+1));
E4=Ein(3*dims+3:4*dims+1);

%Call out the integration function
intab=intFresnelKirchhoff(rhoab,Eab,z,r);
int1=intFresnelKirchhoff(rho1,E1,z,r);
int2=intFresnelKirchhoff(rho2,E2,z,r);
int4=intFresnelKirchhoff(rho4,E4,z,r);

%get error
result=rhostep*(-intab+4*int1-6*int2-2*int4)/45;

%Get actual integral
result2=(14*intab+64*int1+24*int2+28*int4)*rhostep/45;

%compile the integral and the error estimate to send back as the result.
result=cat(1,result,result2);

```

A.3 shuffle.m

```

%shuffle.m
%
%This function shuffles the input vector much like you would shuffle a deck of cards. However,
%it places the new vector in a very specific order as can be interpreted from the call-return.
%
%call: shuffle(1:17)
%return: [1,17,2,6,10,14,4,8,12,16,3,7,11,15,5,9,13]
%
%This function is written specifically for intPoly4.m. This order is chosen because it groups
%entries from the old vector together in the new vector that behave identically in Poly4
%integration. The above return is really the concatenation of the sub-vectors:
%[1,17], [2,6,10,14], [4,8,12,16], [3,7,11,15], [5,9,13]
%
function [y]= shuffle(x)

```

```

chunk=(length(x)-1)/4; %determine the base length for each sub vector
y=zeros(size(x)); %create a vector frame to place the new order for the values.
y(1)=x(1);
for m=1:chunk-1 %set in all of the values that fit the repeatable pattern
    y(2+m)=x(4*m-2);
    y(2+2*chunk+m)=x(4*m-1);
    y(2+chunk+m)=x(4*m);
    y(2+3*chunk+m)=x(4*m+1);
end

y(chunk+2)=x(chunk*4-2); %set in the remaining values
y(chunk*3+2)=x(chunk*4-1);
y(chunk*2+2)=x(chunk*4);
y(2)=x(chunk*4+1);

```

A.4 intFresnel.m

```

%intFresnel.m
%by Benjamin Bellville, BYU 2002
%
%This program performs the integration for the Fresnel diffraction formula
%in cylindrical coordinates. This includes the Fresnel approximation of the
%obliquity factor.  $(1 + \cos(a,b))/2 = 1$ 
%
%For questions on the Fresnel diffraction formula see section 10.2 of
%"Physics of Light and Optics" by Dr Peatross, 2001. Available at "www.optics.byu"
%
%There is no higher order numerical method inherent to this function. It will do a straight
%left-side rectangular integration using the points given. The factor delta-rho has been
%omitted so that this can be used as a child program for higher-order integration techniques.
%To change this into a higher order method, then multiply the points on the Efield by the
%coefficients for that method. See simpsons.m and poly10.m
%
%call: intFresnel(rho,Ein,z,r)
%
%rho= vector containing values for the variable of integration
%Ein= vector containing values for Efield at the plane of integration
%z= distance from the plane of integration to the plane where you desire to know the Efield.
%r= vector containing values for the radial locations where you desire to know the Efield.
%
%All variables with dimensions of length must be given in terms of the wave number, k.
%k=2*pi/wavelength If Z is 1 meter, then pass Z=1*k into this function.
%
%This returns a vector containing the values of the Efield at points (z,r).

```

```
function[Eout]=intFresnel(rho,Ein,z,r)
```

```

%This integration technique temporarily creates a matrix the size of (rho x r').
%This matrix can easily surpass the memory limits of the computer and more commonly
%cause an exponential decay in the processing speed by over taxing the hard drive.
%(The hard drive slows down as it fills up.) Therefore, it is expedient to break up

```

```

%the integration before it fills up the hard drive.
%
%It assumes that the computer can handle 96 megabytes comfortably.
%Each entry in the (Erhor x Erho) array is complex, it takes 16 bytes.
%16*3e6*2 = 96 megabytes. The factor of 2 is in there because the processing speed
%decays exponentially just after the available memory is half used.
%These section can be adapted for each computer. The line to change is flagged** below.
%Notice that it is the factor of 3e6 that is put into the program.

%Determine how to beak up the integration. The vector r gets divided.
m=length(r);
rslices=ceil(m*length(rho)/3e6); %This is the line to adapt.*****
rslice=floor(m/rslices);
rrem=rem(m,rslices);

%Prepare two vectors and constants for integration
Erho=exp(i/2/z*rho.^2).*Ein.*rho;
constants=-i*exp(i*z)/z;

%Do the integration
rstart=1;
rend=rslice;
for n=1:rslices; %this integrates for one segment of r at a time.
r2=r(1,rstart:rend);
%%%%%%
Emain(n,:)=Erhor(r2,rho,z,Erho); %Erhor is a function below, it is the heart of integration
%%%%%%
rstart=rend+1;
rend=rstart+rslice-1;
end

%Compute Eout for the remainder of r (if any)
%and compile Eout from Emain.
if rrem~=0
r2=r(1,(m-rrem)+1:m);
Eout=Erhor(r2,rho,z,Erho);

%Compile the Emain onto Eout
for k=1:rslices;
Eout=cat(2,Emain(rslices+1-k,:),Eout);
end
else %compile Eout from Emain
Eout=Emain(1,:);
for k=2:rslices;
Eout=cat(2,Eout,Emain(k,:));
end
end

Eout=Eout*constants;

```



```
return
```

```
%This function does the bulk of the integration.  
%Excluding the time to clear the unnecessary variables,  
%it would be faster to do this in the body of the function.
```

```
function matrix = Erhor(r,rho,z,Erho)
```

```
    Erhor=BESSELJ(0,r'*rho/z);
```

```
    Er=exp(i/2/z*r.^2);
```

```
    matrix=Er.*(Erhor*(Erho))';
```

```
return
```

A.5 poly10.m

```
%poly10.m
```

```
%Benjamin Bellville, BYU 2002
```

```
%
```

```
%This program creates an array of multiplication values
```

```
%for tenth-order-polynomial-fit integration.
```

```
%
```

```
%It is done using the same theory as simpson's 1/3 rule,
```

```
%which fits a second order polynomial.
```

```
%
```

```
%The output array is as follows:
```

```
%[x1,x2,x3,x4,x5,x6,x5,x4,x3,x2,2*x1,...
```

```
% ...,2*x1,x2,x3,x4,x5,x6,x5,x4,x3,x2,x1]*5/299376=poly10
```

```
%
```

```
%The input array length must be 10*n+1. (See function "factor10")
```

```
%This function will not return an error if the number is not of 10n+1 length.
```

```
%This program also assumes that length(input array)>=21
```

```
%
```

```
%See Also simpsons.m
```

```
function [s1]=poly10(b);
```

```
n=(length(b)-1)/10;
```

```
m=factor(n);
```

```
L=length(m);
```

```
s1=[16067*2,106300,-48525,272400,-260550,427368,-260550,272400,...  
    -48525,106300];
```

```
for y=1:L
```

```
    s2=s1;
```

```
    for x=1:m(y)-1
```

```
        s2=cat(2,s2,s1);
```

```
    end
```

```
    s1=s2;
```

```
end
```

```
s1(1)=s1(1)/2;
```

```

s1=cat(2,s1,16067);
s1=s1*5/299376;
break

```

A.6 mirror.m

```

%mirror.m
%
%This function mirrors the matrix about the first column. It does not repeat the first entry.
%This changes the vector from an even to an odd number of columns, or vice versa.
%
%call: mirror([1,2,3,4,5],1)
%return: [4,3,2,1,2,3,4,5]
%
%see also mirrorN.m

function [x]=mirror(x)
x2=fliplr(x);
n=size(x);
x2=x2(:,2:n(2)-1);
x=cat(2,x2,x);

```

A.7 mirrorN.m

```

%mirrorN.m
%
%This function mirrors the matrix about the first column. It does not repeat the first entry.
%This changes the vector from an even to an odd number of columns, or vice versa. It also changes
%the sign of the new portion of the vector.
%
%call: mirror([1,2,3,4,5],1)
%return: [-4,-3,-2,1,2,3,4,5]
%
%see also mirror.m

function [x]=mirror(x)
x2=fliplr(x);
n=size(x);
x2=x2(:,2:n(2)-1);
x=cat(2,-x2,x);

```

Appendix B Numerical Integration Method

Choosing a numerical method for integration is very important for programming, especially with oscillatory, ill-behaved, or difficult to evaluate functions and when many iterations of integration are required. Such is the case with diffraction integrals. There are a few questions that are important to ask when choosing the integration technique.

1. How fast will it converge? (number of data points necessary)
2. How long will it take?
3. Does it give you an error estimate? Do you need one?
4. Does it throw out any points or allow you to add more later?
5. How hard is it to program?

Several integration techniques were tested for diffraction applications, Simpsons' 1/3 rule, a tenth-order polynomial fit, a fourth-order polynomial fit, and an adaptive recursive fourth-order polynomial fit. One or two experiments quickly established that adaptive-recursive routines take too much time unless they start with enough data points. Of course, the higher order polynomial fits will converge faster, but they grow exponentially more tedious to derive and then program. The fourth-order polynomial fit seems to be the happiest balance between ease of derivation and programming and convergence rate. It also can give the error if it is derived in the correct way.

Derivation of the Fourth-Order Polynomial Fit Integration, Boole's rule.

There are two ways to derive the fourth-order polynomial fit integration technique, sometimes called Boole's rule, the first is more tedious and less informative.

It involves symbolically solving for the coefficients of a fourth-order polynomial that will fit any five evenly spaced data points, use analytic Lagrangian interpolation. With that nightmarish conglomeration of equations, integration and simplification would yield

$$\left[\frac{14}{45} f_0 + \frac{64}{45} f_1 + \frac{8}{15} f_3 + \frac{64}{45} f_4 + \frac{14}{45} f_5 \right] \Delta x \approx \int_{x_0}^{x_5} f(x) dx.$$

Where f_n is the value of the function at the n^{th} value of x , and Δx is the step size. This same formula results from an analytic error adjustment to Simpson's 1/3 rule.

Simpson's 1/3 rule has error proportional to Δx^4 . So, performing Simpson's 1/3 integration twice, once with N points and once with $(2N-1)$ points, should allow us to get an error estimate for the integration. Dividing the proportional error estimates gives an approximate error equality as follows,

$$\frac{\mathbf{E}_N \propto \Delta x_N}{\mathbf{E}_{2N-1} \propto \Delta x_{2N-1}} \rightarrow \frac{\mathbf{E}_N}{\mathbf{E}_{2N-1}} \cong \frac{\Delta x_N}{\Delta x_{2N-1}},$$

where E is the error of the numeric integration. This can be combined with the statement of numeric integration and the equality of step size,

$$\int f(x) dx = \mathbf{I}_N + \mathbf{E}_N = \mathbf{I}_{2N-1} + \mathbf{E}_{2N-1}$$

$$\Delta x_N = 2\Delta x_{2N-1},$$

to yield the following error estimate

$$E_{2N-1} \cong \frac{1}{15} (\mathbf{I}_{2N-1} - \mathbf{I}_N)$$

$$= \frac{\Delta x}{15} \left[\left(\frac{1}{3} f_0 + \frac{4}{3} f_1 + \frac{2}{3} f_2 + \frac{4}{3} f_3 + \frac{1}{3} f_4 \right) - \left(\frac{1}{3} f_0 + \frac{4}{3} f_2 + \frac{1}{3} f_4 \right) \right]$$

where \mathbf{I} is the result of numeric integration. Adding this to \mathbf{I}_{2N-1} and multiplying it out gives exactly the fourth-order polynomial fit integration formula above, which has error proportional to Δx^6 . It also has the benefit of providing an error estimate without evaluating any extra data points! Although that error estimate is based on an error proportional to Δx^4 , it tends to estimate an error slightly smaller than actual error, as can be shown by comparing numeric and analytic integration. To sum up,

$$\mathbf{E} \cong \frac{\Delta x}{45} (-1f_0 + 4f_1 - 6f_2 + 4f_3 - 1f_4)$$

$$\mathbf{I} = \frac{\Delta x}{45} (14f_0 + 64f_1 + 24f_2 + 64f_3 + 14f_4).$$

Extension to a Recursive Technique

Changing Boole's rule into a recursive technique is an easy change to make, if desired. It is a matter of compiling the values of the function into different arrays that correspond with each coefficient it will be multiplied by for the final integration. With those values, an estimation of the error can be made, and you can determine if you want to double the data points. If you want to double the data points, then make the values that are currently associated with the coefficient 24 to be associated with 28 (a junction from multiple iterations $28=14*2$). The old 64s will be associated with 24 and the new points will be associated with 64. With a while loop and all the other necessary additions to a program, it shouldn't add ten lines of code, but those ten lines can easily double or triple the time it takes to complete the integration as compared to starting with enough data points.