Development of a computer model to simulate wavefront propagation

D.Rafferty[1,2], U.H. Wagner[1], C. Rau[1,3], P. Chang[1], S. Alcock[1], R. Dockree[1,4], I.R. Robinson[1,2]

1. Diamond Light Source Ltd., Harwell Science and Innovation Campus, Didcot, Oxfordshire.OX11 0DE U.K.
2. University College London, Gower Street, London, WC1E 6BT
3. Northwestern University 633 Clark Street Evanston, IL 60208 Evanston: 847.491.3741 Chicago USA
4. University of Southampton, University Road, Southampton SO17 1BJ, U.K

Abstract

A computer model has been developed using MATLAB that allows the user to simulate the propagation of a wavefront through an optical system consisting of lenses, and diffracting structures, and to observe the form of the wavefront in the output plane. The simulation was developed with the intention of being able to predict the effects of optics used in synchrotron beamlines, such as those at the Diamond Light Source. The model is also capable of predicting the transforming effects of small scale surface structures on focusing mirrors in a simplified manner. A method for modelling such features has been suggested, and implemented in this simulation. A complimentary study of small scale surface irregularities was also conducted, details of which can be found in ref [15].

1. Introduction

In experiments involving synchrotron radiation, such as those conducted at the Diamond Light Source, beams of radiation need to be manipulated to fulfil certain specifications that make them suitable for the purposes of the experiment. An example is X-ray diffraction experiments, where a particular wavelength of radiation may be required in order to resolve structures on a certain scale, whilst in other situations a certain beam spot size or degree of coherence may be required. In order to achieve particular properties such as these, the beam is passed through an optical system, the effects of which can be accurately predicted, before being used in a scattering experiment. An optical system may be composed of any number of optical elements, such as monochromator crystals or focusing mirrors, with each being carefully selected to alter the beam characteristics in a desired fashion.

Fourier optics is a very effective method of analysis of optical systems, involving the use of fourier transforms. In contrast to the Huygen's-Fresnel principle, in which a plane wavefront is regarded as a superposition of an infinite number of spherical wavefronts, Fourier Optics does the opposite- an arbitrarily shaped wavefront can be constructed from an infinite number of planar wavefronts. Fourier analysis thus provides an incredibly useful mathematical apparatus with which to evaluate such systems, as the methods allow one to break up a complicated arbitrary waveform into a (possibly infinite) series of simpler complex exponential functions. The inverse fourier transform, conversely, then enables the original complicated function to be reconstructed from these simple functions.

Fourier Transforms in 2 Dimensions [1]

The Fourier transform of a complex valued function f of two independent spatial variables, x and y, is defined as:

$$G(f_x,f_y) = \mathcal{F}(f(x,y)) = \iint_{-\infty}^{\infty} f(x,y) exp\,[-i2\pi(f_x x + f_y y)]dxdy \qquad (1)$$

The transform itself is also a complex valued function of two independent variables, f_x and f_y, which are spatial frequencies. The inverse fourier transform is then defined by:

$$f(x,y) = \mathcal{F}^{-1}\{G(f_x,f_y)\} = \iint_{-\infty}^{\infty} g(f_x,f_y) exp\,[i2\pi(f_x x + f_y y)]df_x,df_y \qquad (2)$$

As such, the functions f(x,y) and $G(f_x,f_y)$ are said to form a Fourier transform 'pair'.

Fourier transforms and inverse Fourier transforms will exist for a function f when they conform to a set of 'existence conditions':
Fourier transforms and inverse Fourier transforms will exist for a function f when they conform to a set of 'existence conditions'. There are several, but it is normally, and sufficient for our purposes to say that for the Fourier transform of a function f(x,y) to exist, it must be absolutely integrable, or square integrable [9].
There are several other conditions that are known to exist [1]. A full discussion of these conditions is not necessary here.
The fourier transform holds a lot of interesting mathematical properties, that have led to several 'Fourier transform theorems', that allow them to be easily manipulated in a number of ways. These will be discussed where they are used in this report.

Diffraction [1]

The pattern produced by a diffracting structure will depend on several factors, most importantly being the geometry of the diffracting structure- such as shapes and sizes of apertures. Independent of this geometry, the wavelength of the incident radiation, the shape of incident wavefronts, and the distance at which an observation screen is placed from the diffracting structure will affect the pattern. In this derivation of the mathematical structure of the diffraction processes, we need to make a number of assumptions:

1. All rays travel in the Paraxial regime (i.e. at small height and angles to the optic axis).
2. Across the surface of any diffracting aperture, the field distribution and its spatial derivative are exactly as they would be in the absence of the diffracting structure, with the field being zero at all opaque points on the structure.
3. The observation distance is many wavelengths from the aperture.

Huygens-Fresnel Principle [4][6][1]

Consider a diffracting aperture in the (ξ,η) plane, as shown below in figure 1 [1]:

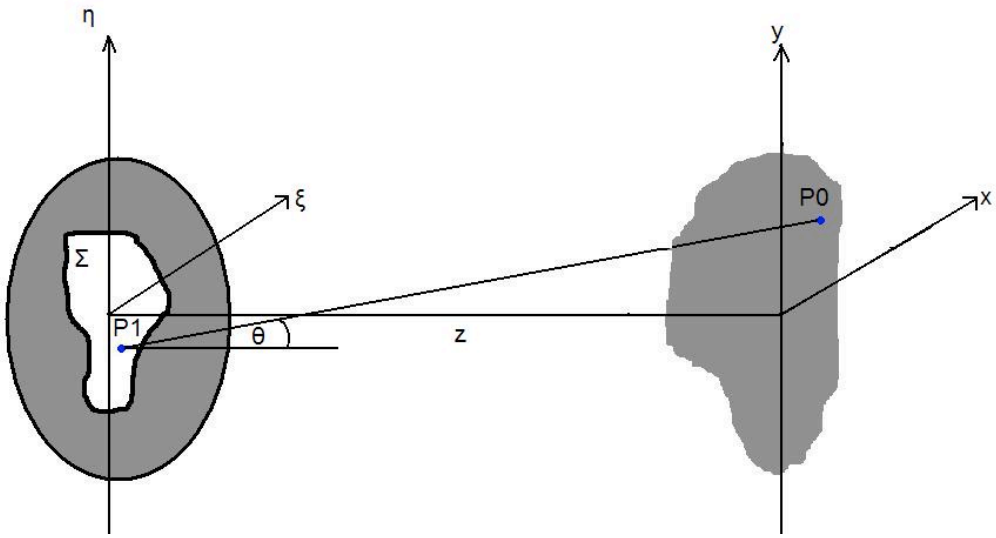


Figure 1: Diffraction geometry

The aperture is illuminated in the positive z direction, and we wish to determine the wavefield in the (x,y) plane, parallel to and a distance z from the (ξ,η) plane.
The Huygens-Fresnel principle can be expressed as:

$$U(P_0) = \frac{1}{i\lambda}\iint_{\Sigma} U(P_1)\frac{exp\ (ikr_{01})}{r_{01}} cos\theta ds \qquad (3)$$

Where, in this case, the angle θ is that between a vector perpendicular to the (ξ,η) plane and the vector $\mathbf{r_{01}}$ joining the points P1 and P0. By substituting the exact expression for cosθ using simple trigonometry, we find:

$$U(x,y) = \frac{z}{i\lambda}\iint_{\Sigma} U(\xi,\eta)\frac{exp\,(ikr_{01})}{{r_{01}}^2}d\xi d\eta \qquad (4)$$

The distance between the points P1 and P0 being given by:

$$r_{01} = \sqrt{z^2 + (x-\xi)^2 + (y-\eta)^2} \qquad (5)$$

To convert (4) to a more usable form, we expand $\mathbf{r_{01}}$ as a Taylor series by factoring a z outside the square root in (5). Keeping only the first two terms of the expansion we obtain the Fresnel approximation:

$$r_{01} = z\sqrt{1 + \left(\frac{x-\xi}{z}\right)^2 + \left(\frac{y-\eta}{z}\right)^2} \approx z\left[1 + \frac{1}{2}\left(\frac{x-\xi}{z}\right)^2 + \frac{1}{2}\left(\frac{y-\eta}{z}\right)^2\right] \qquad (6)$$

Now, we can approximate further by observing that, in the denominator of the integrand, the second two terms in this expression contribute very little, and little error arises from neglecting them. As such, we can approximate the denominator simply as z. In the exponent, however, these terms will be significant, as phase changes of even a fraction can alter the value of this exponential term by a large amount. Thus we decide to keep the full expression, leaving us with an expression for the field at (x,y) of:

$$U(x,y) = \frac{e^{ikz}}{i\lambda z}\iint_{-\infty}^{\infty} U(\xi,\eta)exp\left\{\frac{ik}{2z}[(x-\xi)^2 + (y-\eta)^2]\right\}d\xi d\eta \qquad (7)$$

This expression is known as the Fresnel diffraction integral. It has the form of a convolution, and can be expressed as:

$$U(x,y) = \iint_{-\infty}^{\infty} U(\xi,\eta)h(x-\xi, y-\eta)\,d\xi d\eta \qquad (8)$$

Where the convolution Kernel is given by:

$$h(x,y) = \frac{e^{ikz}}{i\lambda z}exp\left[\frac{ik}{2z}(x^2+y^2)\right] \qquad (9)$$

Alternatively, the Fresnel diffraction integral, by factoring the term $exp\left(\frac{ik}{2z}(x^2+y^2)\right)$ outside the integral in (7), can be formulated as:

$$U(x,y) = \frac{e^{ikz}}{i\lambda z}exp\left(\frac{ik}{2z}(x^2+y^2)\right)\iint_{-\infty}^{\infty}\left\{U(\xi,\eta)e^{\frac{ik}{2z}(\xi^2+\eta^2)}\right\}e^{-i\frac{2\pi}{\lambda z}(x\xi+y\eta)}d\xi d\eta \qquad (10)$$

This is the Fourier transform of the product of the complex field just to the right of the aperture and a complex phase exponential. This will be referred to hereon as method 2, and the convolution formulation as method 1.

There is now one further approximation that can be made to this expression- the Fraunhoffer approximation, valid when:

$$z \gg \frac{k(\xi^2+\eta^2)_{max}}{2} \qquad (11)$$

Here, k is the wavenumber of the incident wave. In this case, the quadratic phase factor within the integral in (10) is approximately equal to one over the entire aperture, and thus the field distribution

can be found directly from a Fourier transform of the aperture distribution. Thus, in the Fraunhoffer regime:

$$U(x,y) = \frac{e^{ikz}}{i\lambda z} exp\left(\frac{ik}{2z}(x^2+y^2)\right)\iint_{-\infty}^{\infty} U(\xi,\eta)\, e^{-i\frac{2\pi}{\lambda z}(x\xi+y\eta)} d\xi d\eta \qquad (12)$$

Which can be identified as the product of multiplicative phase factors and the Fourier transform of the aperture distribution, evaluated at frequencies $f_x = x / \lambda z$, $f_y = y / \lambda z$.
The condition for the validity of the Fraunhoffer approximation is given by the 'antenna designer's formula', which gives the valid region for an aperture of linear dimension D illuminated by light of wavelength λ as:

$$z > \frac{2D^2}{\lambda} \qquad (13)$$

Fresnel number [5]

Fresnel number is given by $$F = \frac{a^2}{\lambda z} \qquad (14)$$

When F is greater than or equal to one, the diffraction event occurs in the Fresnel regime. When F is very small (very much less than one), the Fraunhoffer approximation can be used.

Angular spectrum approach to diffraction [1]

A useful approach to diffraction events is to conduct the Fourier analysis at a given plane so that the different Fourier components of the field distribution are identified as plane waves propagating away from that plane in different directions. As such, the Fourier transform of the field gives the angular spectrum of the secondary wavelets, and the field amplitude at other points can then be calculated by summing the contributions of these plane wave components, taking into account their relative phase shifts which arise as a result of the fact that they travel different distances to a later parallel plane. This is seen in figure 2.

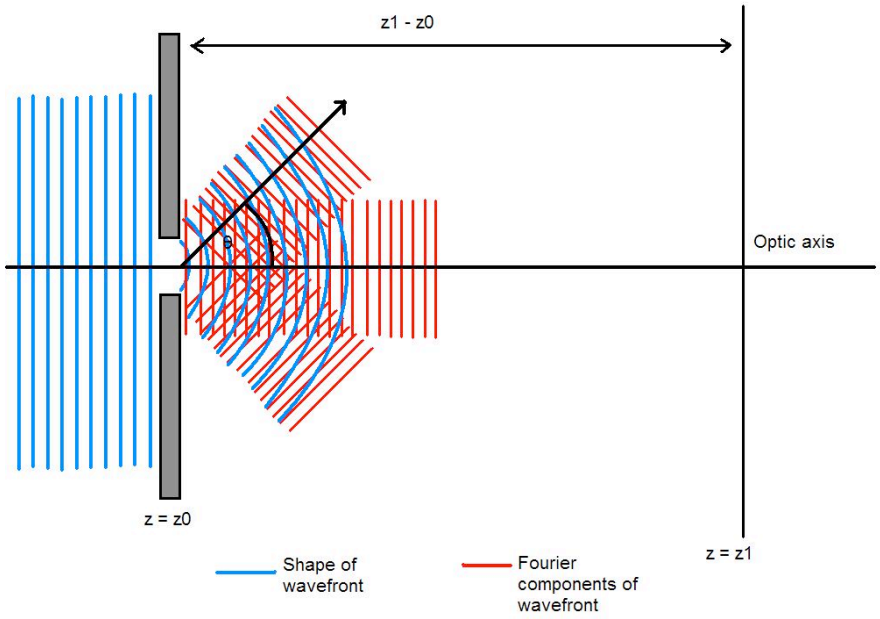


Figure 2: Angular spectrum of diffracted plane wave

It is clear that, for example, the components of the wave which travel at larger angles to the optic axis will travel a greater distance to a subsequent parallel plane. Path differences between different secondary wavelets mean that there are phase differences. It is important to remember that such emission of secondary wavelets will occur at all points on the diffracting aperture- this gives rise to

the characteristic diffraction 'fringes', which arise as a result of constructive interference between secondary wavelets at different points on the observation screen. Thus, by regarding the Fourier series as a distribution of different angular components of the wavefront, it is fairly simple to determine the field amplitude after the disturbance has propagated to a subsequent parallel plane by considering the angular field distribution and the distance to the subsequent plane.
The angular spectrum is obtained by taking the Fourier transform of the complex field across each plane. The angular spectrum of the disturbance U(x,y,0) (i.e. that across the z=0 plane) is given by:

$$A\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda},0\right)=\iint_{-\infty}^{\infty}U(x,y,0)exp\left[-i2\pi\left(\frac{\alpha}{\lambda}x+\frac{\beta}{\lambda}y\right)\right]dxdy \qquad (15)$$

Where $\alpha = f_x\lambda$, $\beta = f_y\lambda$, $\gamma=\sqrt{1-\alpha^2-\beta^2}$ are the direction cosines relating to the wave vector $\overrightarrow{K}=\frac{2\pi}{\lambda}(\alpha\hat{x}+\beta\hat{y}+\gamma\hat{z})$, as shown in figure 3.

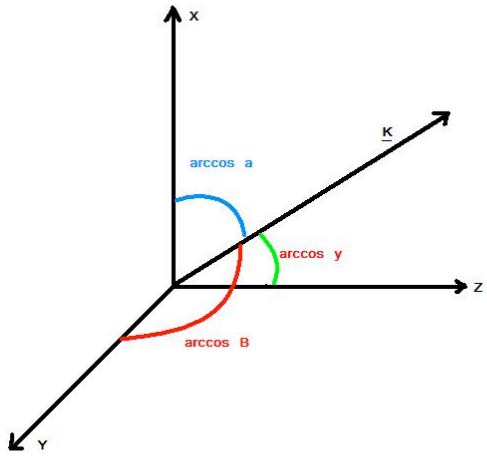


Figure 3: Direction cosines of wave vector K

In order to determine the angular spectrum at a parallel plane, a distance z from the z=0 plane, we consider the angular spectrum A(α/λ,β/λ,z) of the field U(x,y,z):

$$A\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda},z\right)=\iint_{-\infty}^{\infty}U(x,y,z)exp\left[-i2\pi\left(\frac{\alpha}{\lambda}x+\frac{\beta}{\lambda}y\right)\right]dxdy \qquad (16)$$

Then we write U(x,y,z) as: $U(x,y,z)=\iint_{-\infty}^{\infty}A\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda},z\right)exp\left[i2\pi\left(\frac{\alpha}{\lambda}x+\frac{\beta}{\lambda}y\right)\right]d\frac{\alpha}{\lambda}d\frac{\beta}{\lambda}$ (17)

U must satisfy the Helmholtz equation at all source free points: $\nabla^2U+k^2U=0$ (18)

This leads to a differential equation, an elementary solution of which is:

$$A\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda};z\right)=A\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda};0\right)exp\,(i\frac{2\pi}{\lambda}\sqrt{1-\alpha^2-\beta^2}z \qquad (19)$$

This leads to the condition that $\alpha^2 + \beta^2 > 1$, i.e. that the effect of propagating the angular spectrum over a distance z results only in relative phase changes of the different components of the angular spectrum, resulting from the different plane wave components travelling at different angles, and thus over different path lengths to the plane at a distance z.

In order to consider the effect of a diffracting aperture on the angular spectrum, we must define the amplitude transmittance function t_A- this is the ratio of the transmitted field amplitude U_t to the incident field amplitude U_i:

$$t_A(x,y)=\frac{U_t(x,y;0)}{U_i(x,y;0)} \qquad (20)$$

Here we must introduce the convolution theorem, which states that the convolution of two functions in the space domain is equivalent to the inverse Fourier transform of the product of their individual

Fourier transforms. This allows us to relate the angular spectrum of the incident field A_i and that of the transmitted field A_t:

$$A_t\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda}\right) = \left[A_i(\frac{\alpha}{\lambda},\frac{\beta}{\lambda}) * T(\frac{\alpha}{\lambda},\frac{\beta}{\lambda})\right] \quad (21)$$

Where:

$$T\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda}\right) = \iint_{-\infty}^{\infty} t_A(x,y) exp\left[-i2\pi(\frac{\alpha}{\lambda}x + \frac{\beta}{\lambda}y)\right] dxdy \quad (22)$$

The transmitted angular spectrum is thus seen to be the convolution of the angular spectrum of the incident field, and one that is defined by the diffracting structure.
In the case of a plane wave at normal incidence, this becomes the simple case:

$$A_i\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda}\right) = \delta(\frac{\alpha}{\lambda},\frac{\beta}{\lambda}) \quad (23)$$

Which gives the transmitted spectrum:

$$A_t\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda}\right) = \delta\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda}\right) * T\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda}\right) = T\left(\frac{\alpha}{\lambda},\frac{\beta}{\lambda}\right) \quad (24)$$

Thus, the transmitted angular spectrum is simply the Fourier transform of the amplitude transmittance function. As such, a large simplification is present in the case of a normally incident plane wave.

Discrete Fourier Transform (DFT) [3][7]

Unfortunately, only simple functions can be Fourier transformed analytically with relative ease. For more complicated functions we need to resort to numerical methods with which to compute a Fourier transform. By designing the program to calculate the necessary Fourier transforms by such methods, it will be able to handle arbitrary functions, maximising its applicability. The DFT is so-called as the transform is evaluated at discrete points of the function. As such, we must adapt equations 1 and 2 to account for this:

$$G(p\Delta f_x, q\Delta f_y) = \mathcal{F}(f(x,y)) = \sum_{m=0}^{N_x-1}\sum_{n=0}^{N_y-1} f(m\Delta x, n\Delta y) exp\left[-i2\pi(\frac{pm}{N_x} + \frac{qn}{N_y})\right] \quad (25)$$

$$f(m\Delta x, n\Delta y) = \mathcal{F}^{-1}\left(G(p\Delta f_x, q\Delta f_y)\right) = \frac{1}{N_x N_y}\sum_{p=0}^{N_x-1}\sum_{q=0}^{N_y-1} G(p\Delta f_x, q\Delta f_y) exp\left[i2\pi(\frac{pm}{N_x} + \frac{qn}{N_y})\right] \quad (26)$$

Here p and q, and m and n are indices to represent different points in real and Fourier space. N_x and N_y are the number of data points, or samples, in the x and y directions. Δx and Δy represent sampling period in real space, Δf_x and Δf_y represent sampling period in Fourier space. These sampling periods define the resolutions with which our object is sampled, and our image is rendered respectively- a high resolution corresponds to low sampling period. We can see that these sampling periods are not independent of each other. In fact, an increase in resolution in real space will cause a reduction in resolution in Fourier space, and vice versa. For example- if it were required to find the far-field (i.e. Fraunhoffer) diffraction pattern produced by illumination of a aperture of a given size and shape, we would sample the transmittance of the aperture in real space, and would obtain the angular spectrum in Fourier space by performing a DFT on our sample. The shape of the aperture must be accurately sampled in order to produce an accurate angular spectrum, and so it is desirable to sample it in a high resolution. However, this will then lead to a low resolution in Fourier space, so the fine details of the angular spectrum can be difficult to identify. This will not be of significant inconvenience if the aperture is of simple shape, but for more complicated apertures, or near-field diffraction, this lack of resolution in Fourier space can mean that important details in the amplitude spectrum are missed (see later).
The solution to the problem is that when the resolution is increased in real space, the sample size must also increase in order to maintain the same resolution in Fourier space. However, this will then increase the number of samples, and thus increase the computation time. The limiting factors

in resolving the details of the angular spectra and spatial distributions are the computational resources available- memory and processing speed. The calculations quickly become intractable with increasing sample sizes, so care must be taken to select a sampling resolution appropriate for the given circumstances.
Whilst the resolution to which the aperture transmission function can be sampled is limited, the Nyquist-Shannon sampling theorem provides an upper limit on the resolution necessary to which it is necessary to sample:

"*If a function x(t) contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced* $\frac{1}{2B}$ *seconds apart."*

Whilst the above statement is given in terms of times and temporal frequencies, the same can be said in terms of space and spatial frequencies. Thus we can identify an optimum sampling resolution- if we know the maximal values of the spatial frequencies of a given spectrum, we can choose a resolution that will ensure that the spectrum is rendered as accurately as possible. Any higher resolution will result in a waste of computing resources.

The Nyquist limit allows calculation of the real space resolution necessary to effectively obtain frequencies up to a maximum value: $\alpha_{max} = \frac{\lambda}{2\Delta x}$

2. Design of simulation program

Since the case of normally incident plane waves has been shown to simplify matters considerably, it was decided that the simulation should initially be developed only to account for this special case, and then once this simple case could be effectively simulated, to extend the model to allow for arbitrary angles of incidence and waveforms.
The model was developed by initially considering the very simple case of diffraction from a one dimensional rectangular slit, with the observation distance in the far-field in order that the Fraunhoffer approximation be applicable. The model was then subsequently developed to perform an equivalent analysis in two dimensions, allowing the effects of the shape of the diffracting structures to be taken into account.
As mentioned previously, these programs were developed using MATLAB. MATLAB, standing for MATrix LABorotory, is a numerical computing environment that was developed by MathWorks. It works on a scripted programming language, and is primarily concerned with dealing with matrices and data arrays. It is a powerful mathematical tool, widely used throughout the academic and industrial world in the fields of science, engineering and economics [2]. The reader is asked to refer to the appendix for details of the code used in the programs.

1D Fraunhoffer and Fresnel diffraction simulation

In one dimension it is not possible to define a particular shape of diffracting structure, only to specify its principal (largest) dimension. As such, the one dimensional case serves no real practical purpose, although it is a useful way of examining the mathematical structure of the diffraction events in a simple manner, and is quite easily extended to two dimensions. Another useful aspect of the one dimensional calculation is that it can render the diffracted field amplitude in a given direction to much greater resolution than the 2D calculation, due to samples being restricted to one dimension, and thus being able to have a greater sample size in that single dimension. Whilst this may not be of much use for complicated aperture distributions, for simple shapes where symmetric patterns are expected to be produced, it will be able to generate a high resolution cross section of the 2D diffracted field distribution.
The program developed for the one dimensional case was designed to be applicable to a range of input conditions, and to respond by analysing these conditions and calculating whether the full Fresnel treatment of diffraction is necessary, or if in fact the Fraunhoffer approximation will suffice. Therefore, the first part of the program was written to ask the user to input the wavelength of light used to illuminate the diffracting structure (in nm), the principal dimension of the diffracting structure (in mm), and the distance at which the diffraction pattern is observed (in m). After these

basic parameters are defined, the program was then coded to calculate the Fresnel number (using equation 14), and to decide which treatment is necessary to calculate the diffracted field amplitude. This is all of the input required of the user in order to calculate the diffraction pattern. The diffracting screen was defined in the MATLAB language as being a one dimensional array of data with a zero value at all points except in the location of the slit, where it was valued unity. This is an important approximation- in reality this may not be the case (the transmittance over an aperture often has a more complicated distribution, such as a Gaussian), but for simplicity, the model was designed thus, with the opportunity for development later. This in effect is the amplitude transmittance function seen in equation (20). As we have made the simplification that the incident light is of planar wavefronts and normally incident, the Fraunhoffer case is easily computed as the Fourier transform of this function. In MATLAB, a discrete Fourier transform (DFT) is performed using the FFTW (Fastest Fourier Transform in the West) algorithm [2]. There is a pre-programmed function in MATLAB by which this can be evaluated.
Once the FFT has been performed, the co-ordinates of the Fourier transform must be re-scaled in order to display the correct angular spectrum subject to the values of the input parameters. This rescale factor arises as a result of the transform from real to fourier space, and is dependent on the wavelength of the incident light, the sampling frequency and the number of samples. It is given in equation (25) below:

$$\Delta\alpha = \frac{1}{N\Delta x} \tag{25}$$

Here $\Delta\alpha$ represents the rescaled spacing of transformed data points (in radians), N is the total number of data points, Δx is the spacing between them. The original x data (before FFT performed) is multiplied by this scaling factor to give the x co-ordinates against which the FFT data should be plotted. In the 2D case, a similar scaling factor must be applied to the y data. Also, a factor of λ needs to be included in the case of electric field distributions such as those we are considering, arising from the Fourier transform of the wave equation.
The amplitudes also must be rescaled. In order to do this, we consider Parseval's theorem, another of the Fourier transform theorems. This theorem states that if $\mathcal{F}\{g(x,y)\} = G(f_x, f_y)$, then:

$$\iint_{-\infty}^{\infty} |g(x,y)|^2 \, dxdy = \iint_{-\infty}^{\infty} \left|G(f_x, f_y)\right|^2 d\, f_x d f_y \tag{26}$$

i.e. The integral of the field intensity must be the same before and after transformation. This is a statement of conservation of energy, and allows a simple way of finding the appropriate factor by which to rescale the coordinates. By taking the ratio of left to right, we find the scaling factor by which the intensity spectrum of the diffracted field must be multiplied. Figures 4 and 5 show the aperture transmittance function and intensity spectrum produced by a 1mm aperture illuminated by 500nm light, with observation in the far field (100m). This corresponds to a Fresnel number of 0.02.

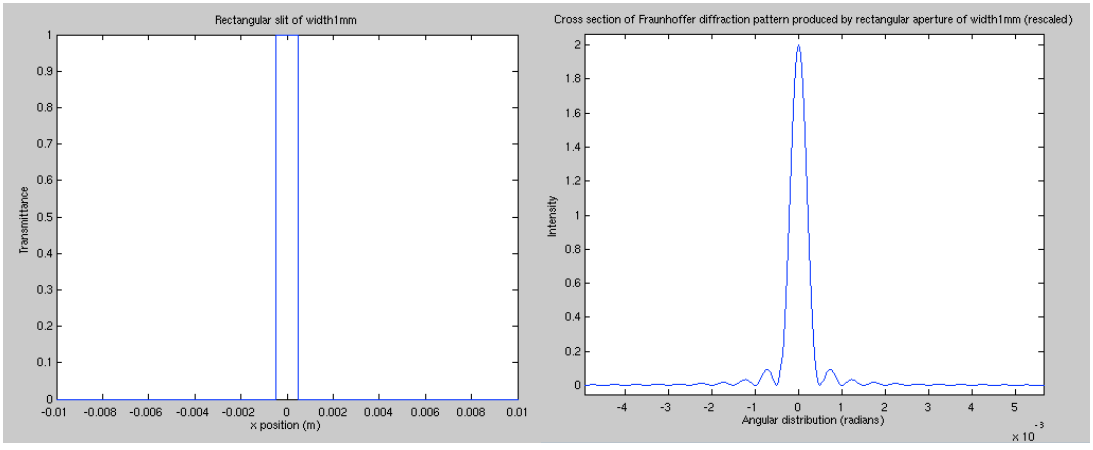


Figure 6: Amplitude transmittance function **Figure 7: Re-scaled FFT of aperture distribution**

In this case we observe the expected sinc function as the Fourier transform of the rectangular transmittance function [1].

For Fresnel diffraction, the diffracted field distribution is obtained through a more complicated analysis. As seen in equations 7-12, the field amplitude distribution produced by diffraction in the near-field can take the form of a convolution, or as the Fourier transform of the product of the aperture transmittance function and a quadratic phase exponential, with some extra multiplicative factors. As a means of verification, it was proposed that the program should be able to calculate the diffraction patterns using both formulations of the Fresnel diffraction integral, which could then be compared. Unfortunately, whilst many efforts were made to allow this, the convolution proved very difficult to compute using MATLAB, and subsequently had to be neglected. The problems encountered are discussed later in this report. The formulation given in equation 12 however, is easily programmable, and was found to work as expected. Figure 8 shows the intensity spectrum produced by illuminating a 1mm aperture with 500nm light, with an observation distance of 0.3m. This corresponds to a Fresnel number of 6.6667, far outside the expected limits of validity of the Fraunhoffer approximation. In order to illustrate the failure of the Fraunhoffer approximation in the near field, figure 9 shows the Fraunhoffer calculation of the intensity spectrum according to the given parameters. For ease of comparison, the angular distribution of the field has been calculated for the Fresnel case using simple trigonometry.

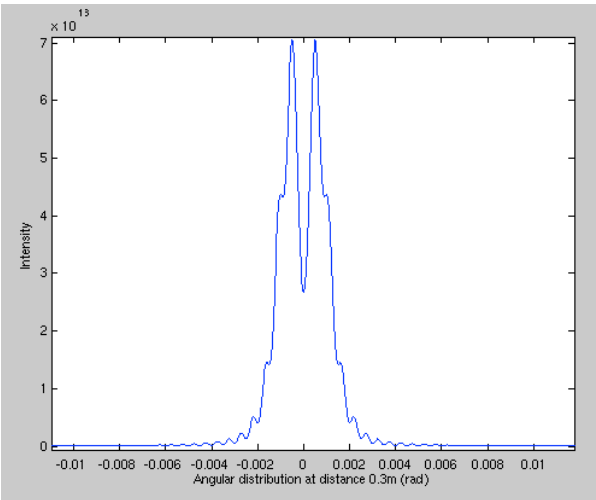


Figure 8: Fresnel calculation at F= 6.6667

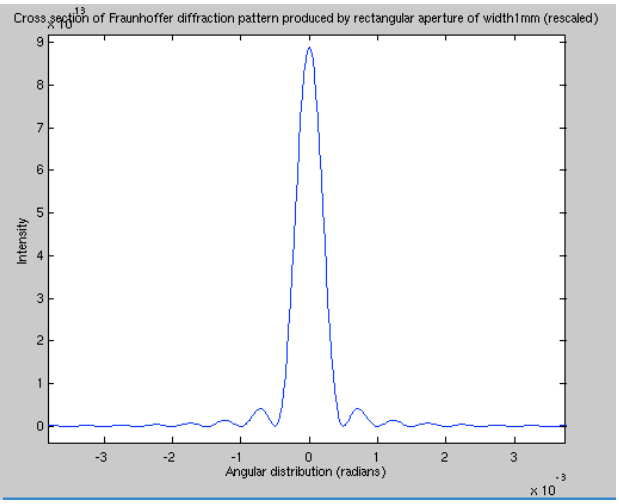


Figure 9: Fraunhoffer calculation at F=6.6667

As can be seen, the Fraunhoffer calculation again produces a sinc function- an approximation that misses many important details of the spectrum. As expected, it is found that the Fraunhoffer approximation is invalid in the near field.

We should, however, expect the Fresnel and Fraunhoffer approximations to produce similar results at large distances. Figures 10 and 11 show the spectra given by the Fresnel and Fraunhoffer calculations respectively, for a 1mm slit illuminated by 500nm light, with observation at 100m. This corresponds to a Fresnel number of 0.02. Again, the Fresnel case has been displayed as an angular distribution for ease of comparison of the two calculations.

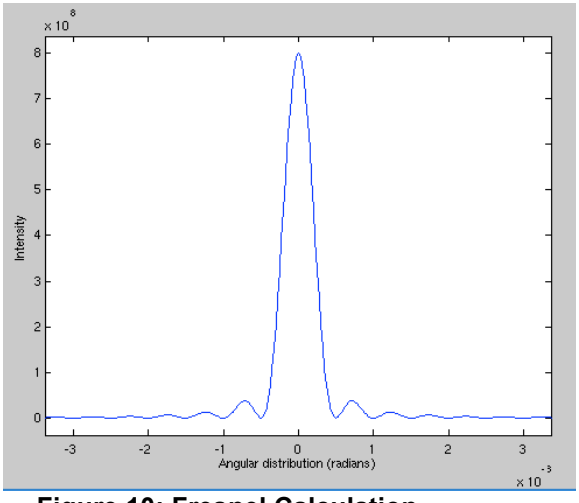


Figure 10: Fresnel Calculation

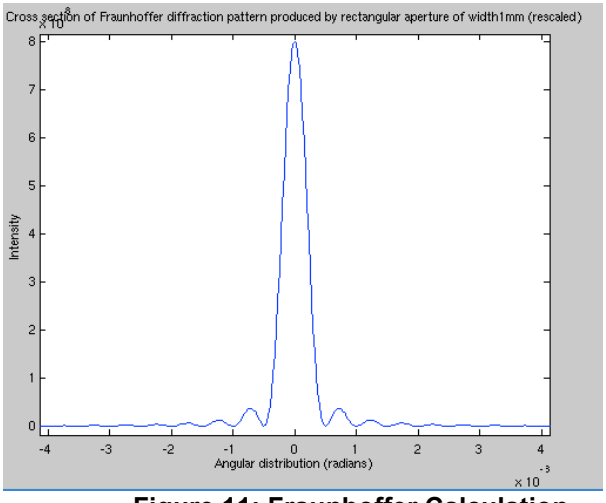


Figure 11: Fraunhoffer Calculation

It can be seen that the two methods of calculation become entirely equivalent in the Fraunhoffer regime, as we should expect from equations 10 and 12. The distance at which there is a transition between the Fresnel and Fraunhoffer regimes is not properly defined- for decreasing Fresnel number (F) the Fraunhoffer formulation becomes an increasingly accurate description of the Fresnel case. Where F becomes very small, we see that the spectra produced by the two different methods become indistinguishable.

2D Fraunhoffer diffraction simulation

The Fraunhoffer case is easily extended to two dimensions. To achieve this, the amplitude transmittance function is defined in two dimensions by creating a 2D array of data, and defining the slit again by setting the amplitude transmittance to zero everywhere except in the location of the aperture. In 2D we are now also able to define the shape of the aperture- the program developed allows for apertures of circular and rectangular shapes, the dimensions of which are input by the user. Another extension that has been made to the program is that it has been made possible to define a rectangular array of apertures. This is useful in being able to predict the effects of, for example, Young's two-slit experiment [3]. Limitations now become apparent due to the limited computational power available. The FFT algorithm works most effectively with data arrays whose dimensions are of powers of two. In order to define the amplitude transmittance function, a square array is defined- zero everywhere but in the location of the aperture(s), where it is valued unity. Again this is an approximation to the way in which an aperture would transmit light, which we use for simplicity. MATLAB includes a feature whereby the programmer can record the time taken for a given set of operations to be performed. As the dimensions of the array representing the amplitude transmittance function are increased, the time taken for the array to be analysed, and an FFT to be performed, increases exponentially. Larger dimensions means a better resolution of the aperture distribution or a wider field of view, which can mean that larger aperture arrays can be analysed, or angular spectra can be resolved in greater detail. Unfortunately, with dimensions of greater powers of two than 11, it was found that the computation time quickly becomes larger than is practically suitable for the hardware available to us. In the Fraunhoffer case it is not strictly necessary to resolve the angular spectra in great detail, as the Fourier transforms of the aperture distributions are fairly simple functions, which are known from the theory. However, we will see that a problem presents itself in the Fresnel regime. Whilst these limitations persist, a feature has been coded into the program that allows for user-defined resolution of the aperture distribution. No limit is actually set on the resolution that can be achieved, and the program is widely applicable to different sets of circumstances. The degree to which the aperture distribution can be resolved is, however, limited in that it must be low enough that all of the important features of the distribution can be included within the field of view. The dimensions of the aperture distribution array have been set to be of length 2^{11} data points. Having tested the program, this corresponds to a computation time of around 50 seconds. 2^{12} points was also investigated, taking around 4 minutes to perform the calculation, which, for our purposes, is a bit too long.
As an example, figure 12 shows the Fraunhoffer diffraction pattern produced by diffraction of 500nm light through a screen with two 1mm radius circular apertures separated by 2mm. Figure 13 shows the relative intensity spectrum of this aperture. The reader is referred to the appendix for details of the code used to program these features, and those used for the 2D Fresnel case.

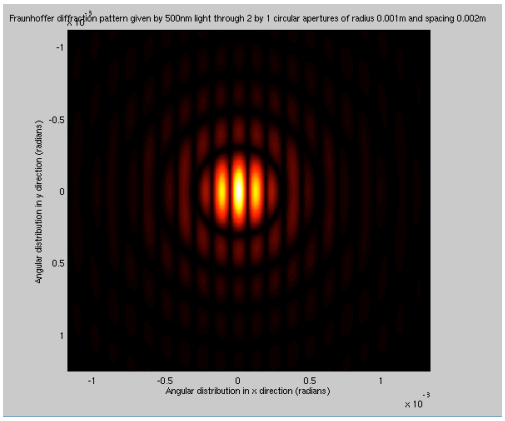


Figure 12: Fraunhoffer diffraction pattern

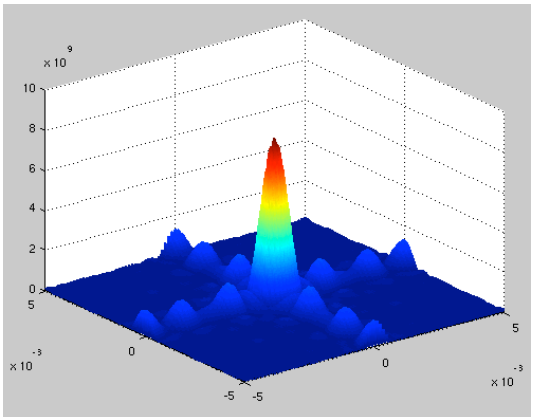


Figure 13: Intensity spectrum

2D Fresnel diffraction simulation

In order to adapt the program to cope with diffraction events in the Fresnel regime, it was necessary to make use of the mathematics described in equations 7 to 12. MATLAB includes a convolution function, which convolves two arrays of data to produce a larger array. As it had already been found that the 2^{11} by 2^{11} arrays already used to define amplitude transmission functions were already very demanding of computer time, it was decided that this method of analysing the Fresnel diffraction integral was unsuitable. The convolution theorem allows a way around this problem. This involved calculating the Fourier transforms of the convolution kernel (equation 8), and the amplitude transmission or aperture distribution function, multiplying them together, and then working out the inverse transform of the product. The convolution formulation of the Fresnel diffraction integral proved very difficult to program. Some of the problems found with the convolution are discussed in the next section.

In the Fresnel case, the sampling frequency becomes of the upmost importance. Fine details of the diffracted field distribution can only be identified when it is resolved in high detail. However, as mentioned before, we cannot maintain high resolutions in both real (sampling the aperture distribution) and Fourier (angular spectrum) space- there is a trade off involved, and the degree of detail to which we can examine a given spectrum is limited by our computer power. A higher sampling resolution in real space leads to a lower resolution of the spectrum in Fourier space. Unless the sample size is increased with the real space resolution, it also causes a wider field of view in Fourier space. This doesn't immediately sound like a problem, but this extra field of view is essentially wasted as it will be beyond that necessary. Due to the limitations on computer power available, this is unavoidable. Due to these problems, it is necessary to make use of the Nyquist-Shannon sampling theorem in order to decide on a suitable resolution with which to sample the diffracting structure, one which, preferably, will be able to render all of the frequency components of the diffracted field. Figures 13a-13d show the effects of increasing real space resolution on rendered field distribution.

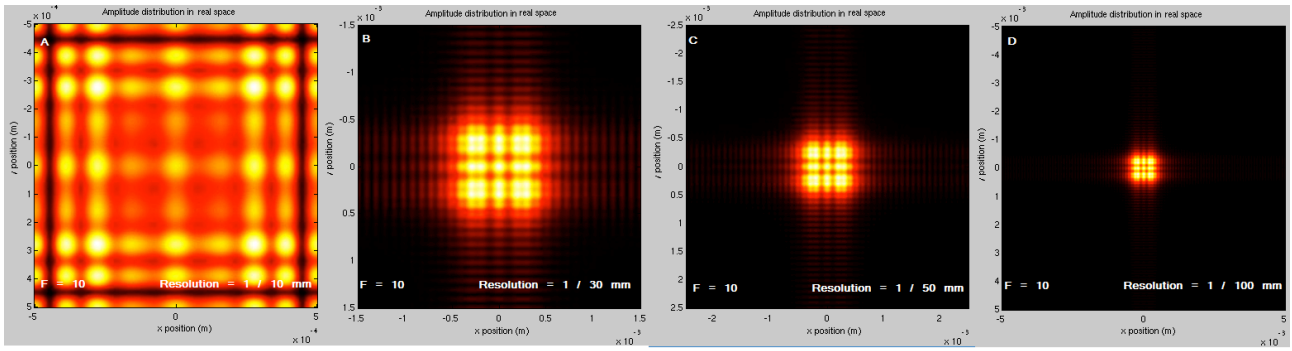


Figure 13- Effect of resolution on rendered angular spectrum

Of course the resolution that we are able to use is limited by the circumstances of the diffraction event we wish to calculate- the sampling frequency we choose must of course still leave a large enough field of view that the entire aperture distribution is sampled. As such, the program was

coded with a feature that allows the user to define the degree of resolution to no upper limit. The program needs to be adaptable in this manner so the user can view the features he or she wishes. Also, if the program is run on a more powerful computer, the user may wish to increase the sample size, so being able to use higher resolutions.
This feature thus allows the user to define the resolution, as is suitable to the pattern that they wish to observe. It would be possible to hard-code the program so that an acceptable resolution is automatically decided upon based on the user's given input parameters. However, in the interests of widening applicability, it was decided that each time the program is run the user should be able to define the resolution according to their own personal objectives.
Using the model, it is possible to observe the transition from the Fresnel regime into the region of validity of the Fraunhoffer approximation, as the distance to the observation point is increased. Figure 14 shows the angular spectra observed for a circular aperture of radius 1mm illuminated by 500nm light for a range of observation distances z, which are displayed on the images, as well as the Fresnel number F. The images clearly show a transition from Fresnel to Fraunhoffer behaviour that occurs somewhere between F = 0.4 and F = 0.2, which is as we would expect from the conditions mentioned in the literature [1]. These images were rendered using a real-space resolution of 1/100mm.

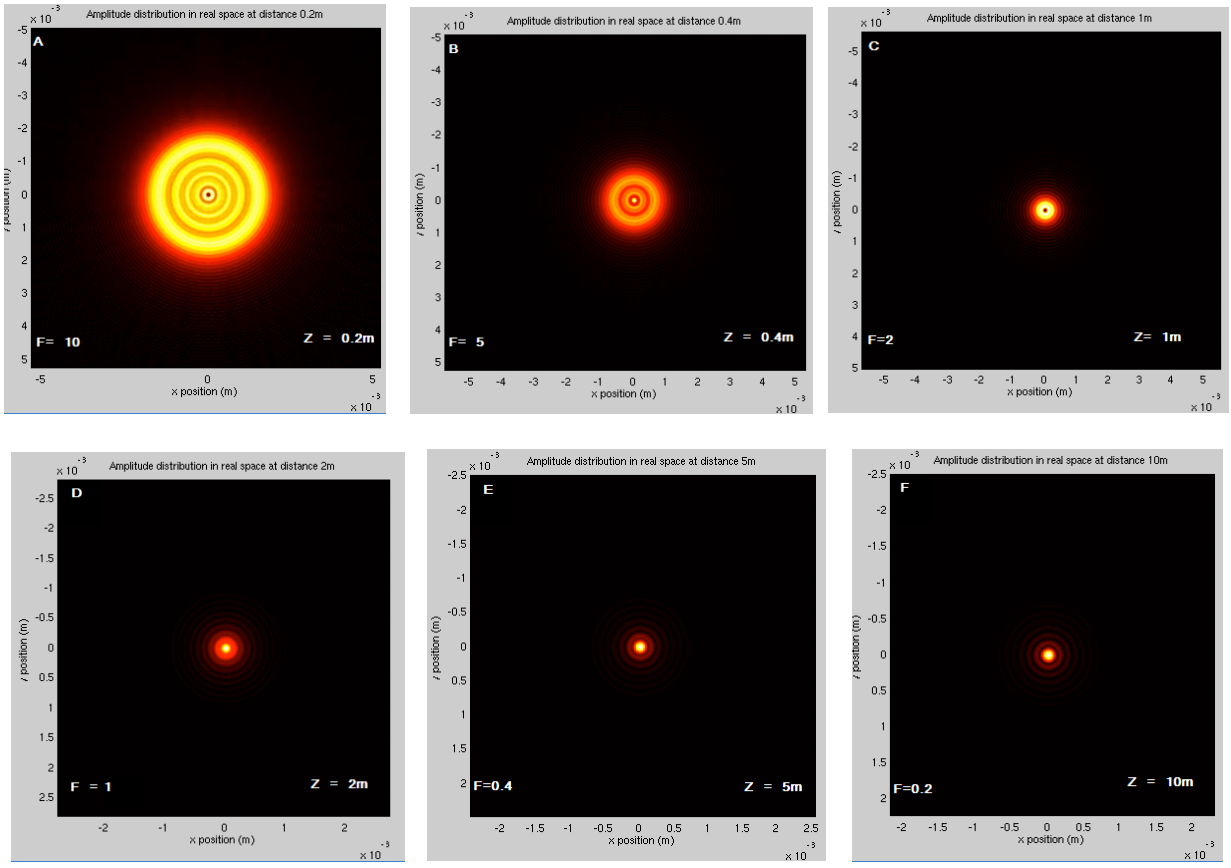


Figure 14: Transition from Fresnel to Fraunhoffer regimes with increasing distance

Experiment to test validity of program calculations

To test the validity of the calculations performed by the computer simulation, a simple experiment was conducted. This involved examining the diffraction patterns produced at various distances as a result of illuminating a circular aperture with laser light. Laser light is coherent and monochromatic, making it suitable for the experiment, since the simulation was designed initially to only account for this simple case. The arrangement of apparatus is seen in figure 15.

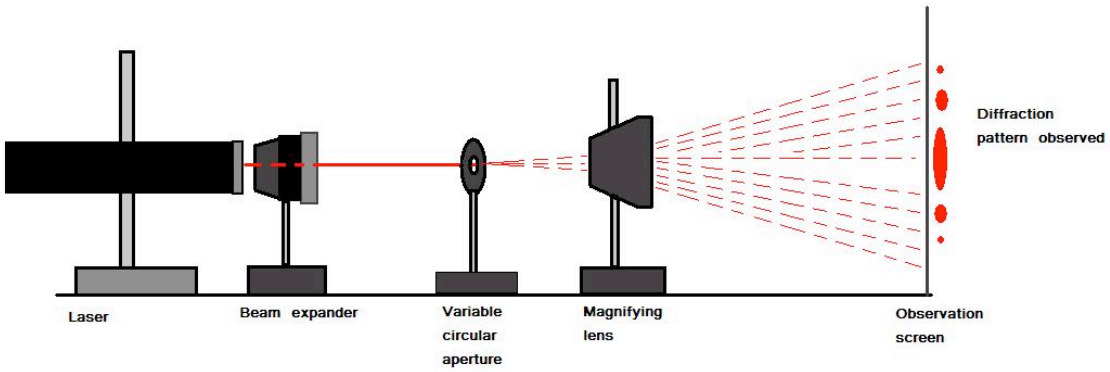


Figure 15: Experimental Set-up

The Laser used in the experiment was manufactured by Thor Labs Inc. (model HRROIS), emitting light with a wavelength of 633nm, at a power of 1.5MW. This model is a class IIIa Laser, so safety precautions had to be taken whilst in use- such as avoiding direct eye contact with beam, not wearing jewellery etc.

To ensure that the wavefronts of light incident on the aperture were planar (as in the simple case the program is able to simulate), the beam was first passed through a 4x beam expander, manufactured by LINOS photonics [11].

After passing through the aperture, the beam passed through a magnifying lens- this was simply to project the field in the plane in which the lens is positioned onto an observation screen, which could then be photographed to capture the diffraction pattern. Thus, by moving the magnifying lens, we are able to observe the diffracted field at different points behind the aperture.

The aperture used was circular, and of variable size. By moving a small lever on the aperture, the experimenter can adjust the radius as required. Unfortunately, the device used does not feature a scale, so the radius of the aperture had to be measured quite crudely using a ruler. As the required aperture size was of around a millimetre, using a ruler with only millimetre precision means there is likely to be a fairly large error in the aperture dimension used in the simulation. Due to this problem, it is sensible to conduct only a qualitative evaluation of the diffraction patterns observed. For now, we will not concern ourselves with the length scales of the diffracted fields- the effects of divergences and the magnifying lens are troublesome to account for. Also, with a CCD array unavailable with which to capture the fields, the patterns had to be photographed from an angle so as not to obstruct the beam. This makes the distance scales in the photograph difficult to make out. In order to perform a full quantitative analysis, it would be necessary to use a high resolution CCD array, arranged in a plane perpendicular to the optic axis.

It was decided to use an aperture radius of 0.5mm, and to examine the diffracted fields at intervals of 6cm behind it, up to a distance of 60cm. This corresponds to Fresnel numbers in the range ~0.6 up to ~6.5. In each case where a diffracted field was photographed, the program was run to simulate the field, and a comparison was made of the two images produced.

Figures 16-25 show photographs of the real diffracted field, and rendered images of the simulated field. In each case F denotes Fresnel number, calculated using equation 14. Each figure shows the simulated field on the left, and the real diffracted field on the right.

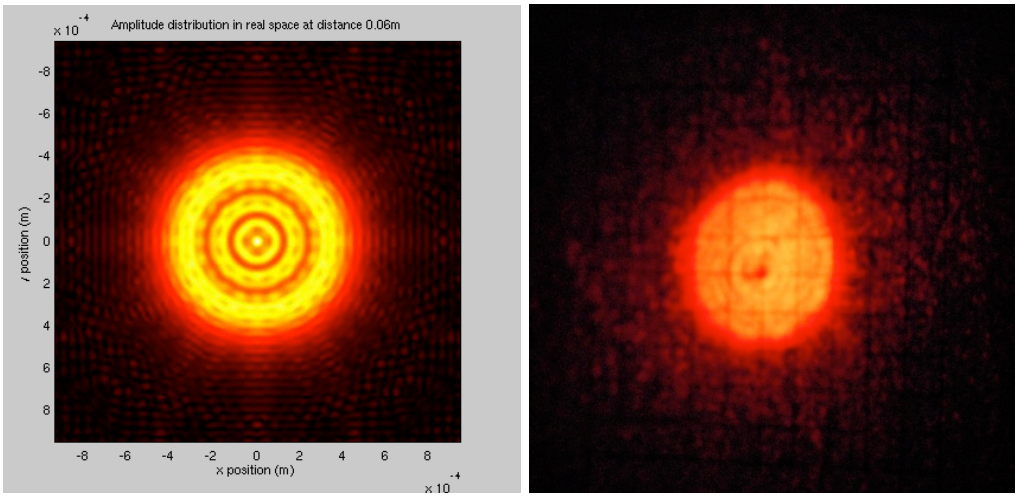


Figure 16: Fields at a distance of 6cm from aperture. F=6.5824

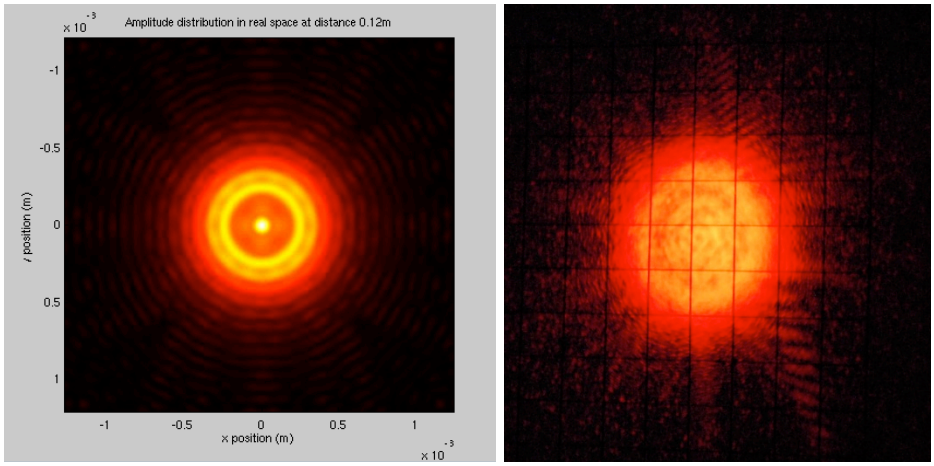


Figure 17: Fields at a distance of 12cm from aperture. F= 3.2912

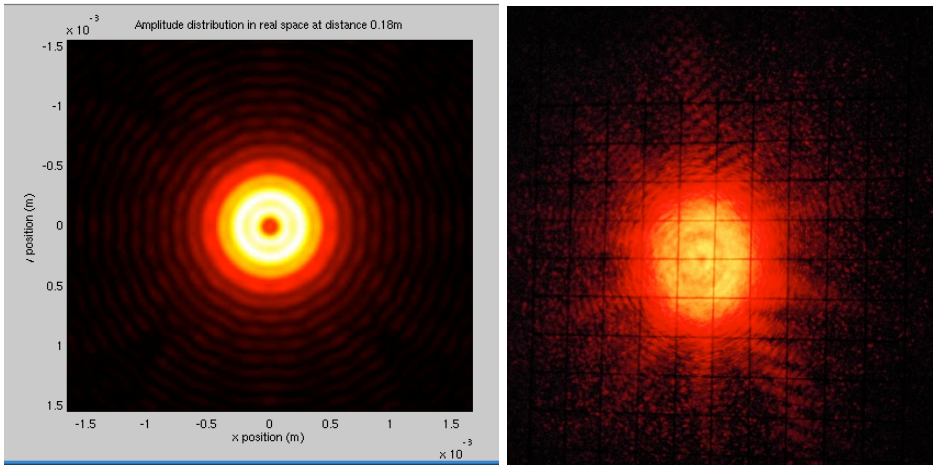


Figure 18: Fields at a distance of 18cm from aperture. F= 2.1941

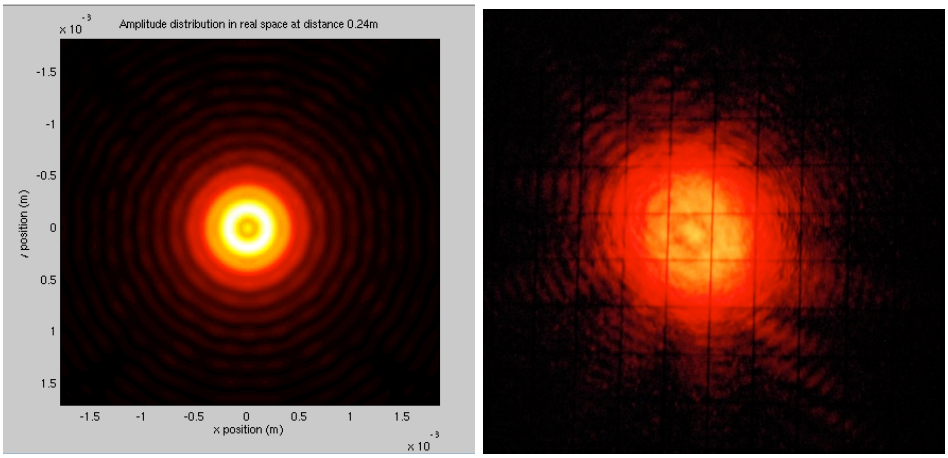


Figure 19: Fields at a distance of 24cm from aperture. F= 1.6456

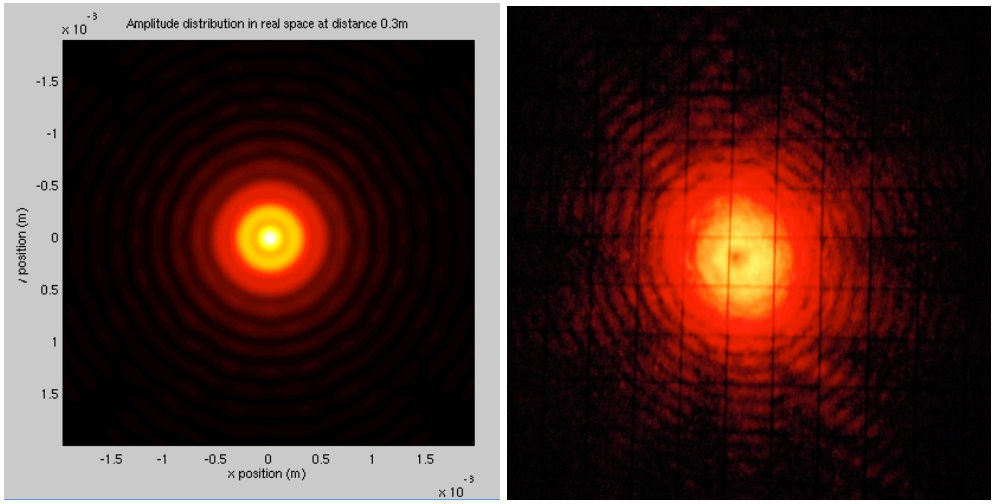


Figure 20: Fields at a distance of 30cm from aperture. F= 1.3165

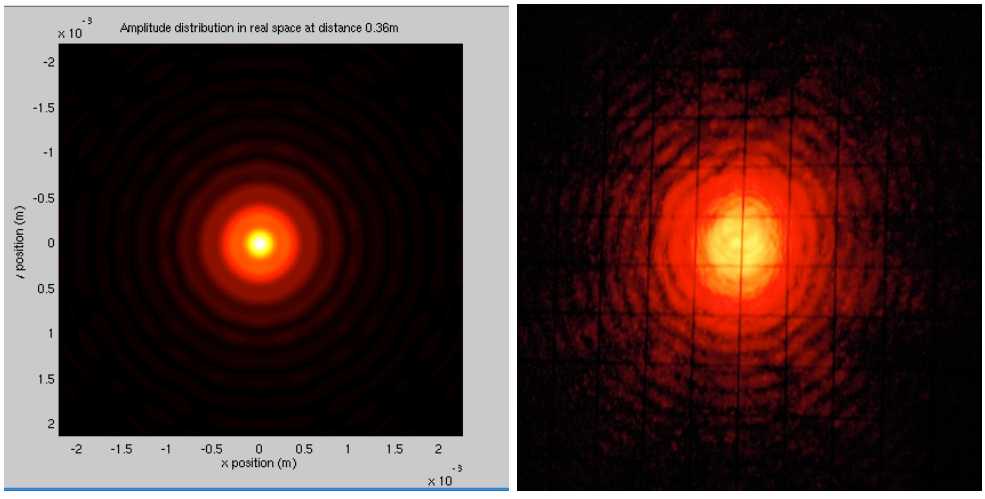


Figure 21: Fields at a distance of 36cm from aperture. F= 1.0971

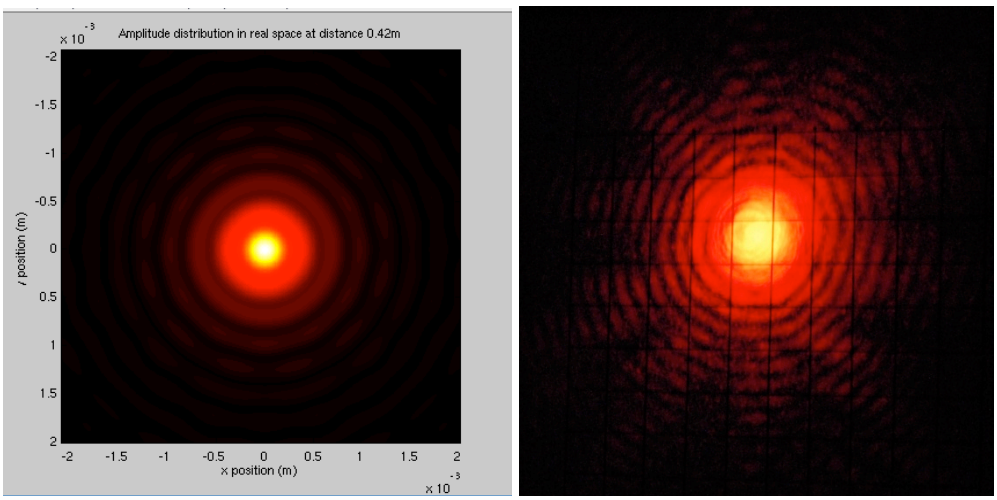


Figure 22: Fields at a distance of 42cm from aperture. F= 0.94034

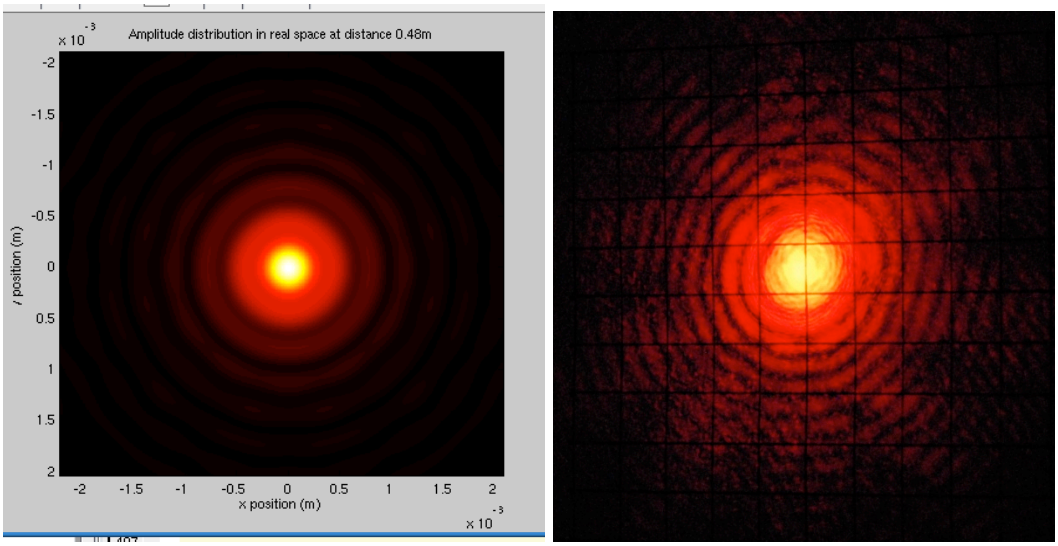


Figure 23: Fields at a distance of 48cm from aperture. F= 0.8228

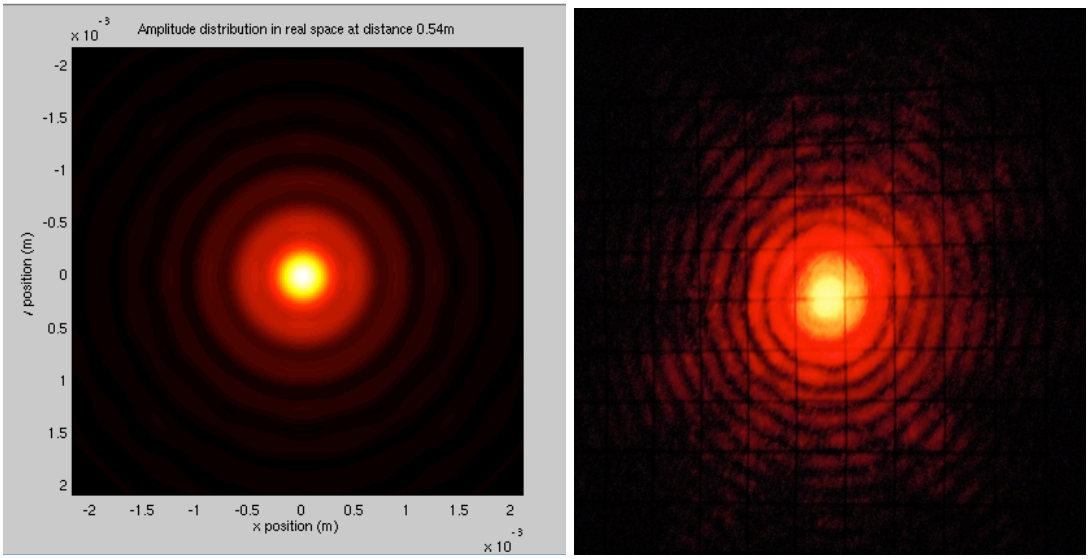


Figure 24: Fields at a distance of 54cm from aperture. F= 0.73138

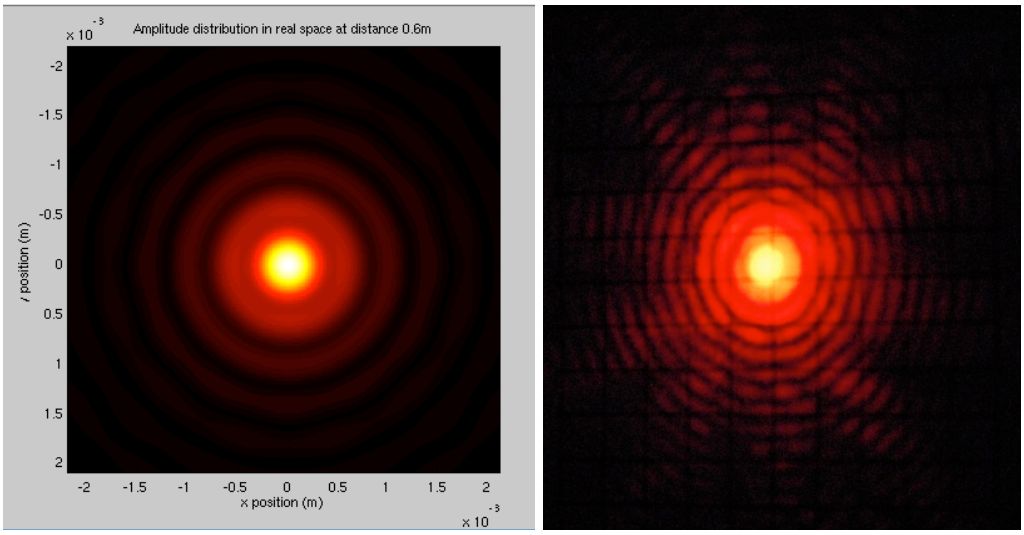


Figure 25: Fields at a distance of 60cm from aperture. F= 0.65824

It can be seen from the images that in a qualitative sense, the simulated fields closely resemble those observed in the laboratory with the same parameters. There are a few slight discrepancies that can be seen- for example in figure 20 (corresponding to the diffracted field being propagated to a distance of 30cm), we observe a dark spot in the centre of the pattern produced in the lab, whilst the simulated case predicts a bright central spot. Otherwise, however, the simulated fields are fairly accurate. Some slight discrepancies are to be expected, due to the high uncertainty in the aperture size, and it is likely that the numerical methods of analysis used to generate the simulated field present additional sources of error. This is discussed in greater depth later.

Despite some small inconsistencies between the real and simulated cases, the computer model appears to work well, and as such we can conclude that the Fresnel propagation simulated in the program gives an accurate representation of the physical process. As such, we can now work to develop the computer model to account for additional types of optical element.

It may be necessary at a later stage to conduct a full quantitative analysis of the observations described here. If so, a more effective method of capturing an image of the diffraction pattern is required- as mentioned, a high resolution CCD array would be suitable for this purpose, if arranged to capture the field in a plane perpendicular to the optic axis. In using such a device, the length scales of the features of the pattern can be recorded to a high precision, which will make a quantitative evaluation possible.

Convolution formulation of Fresnel Integral (method 1)

We have seen that there are two different ways of evaluating the Fresnel diffraction integral. Whilst method 2 has been shown to produce expected diffracted field amplitudes over a wide range of different input parameters, method 1 has proved very difficult to program.

The Fresnel diffraction integral is in the form of a convolution. Using MATLAB, there are two different ways that we can compute this. MATLAB has a built in function conv(x,y) which is able to convolute two arrays of data x and y. However, convoluting two arrays in this way results in an array whose dimensions are equal in length to the sum of the lengths of the dimensions of the convoluted arrays. As we have already seen, there are limitations on the sizes of arrays that it is practical to evaluate in our calculations, due to limited computational resources. As such, this method is not particularly useful, as a smaller sample size must be used so that the calculation can be performed. The alternative is to use the convolution theorem. By doing so, we can keep the

product of the convolution to the same size as our sample- thus the Fresnel Integral can be resolved in greater detail.

As always, the diffraction integral is easiest evaluated in one dimension, so to begin a computer program was written to make these calculations. In order to determine the equivalence of MATLABs convolution function, and an analysis using the convolution theorem, the program was written to perform both calculations for user-defined inputs of the wavelength, slit size, and propagation distance.

Figure 26a shows the diffracted field amplitude, obtained using the MATLAB conv() function, for a slit of width 1mm, illuminated by 500nm light, with a subsequent propagation distance of 1m. This corresponds to a Fresnel number of 2. Figure 26b shows the equivalent result of the convolution theorem. Note that amplitudes here are to be taken as relative- they have not yet been rescaled.

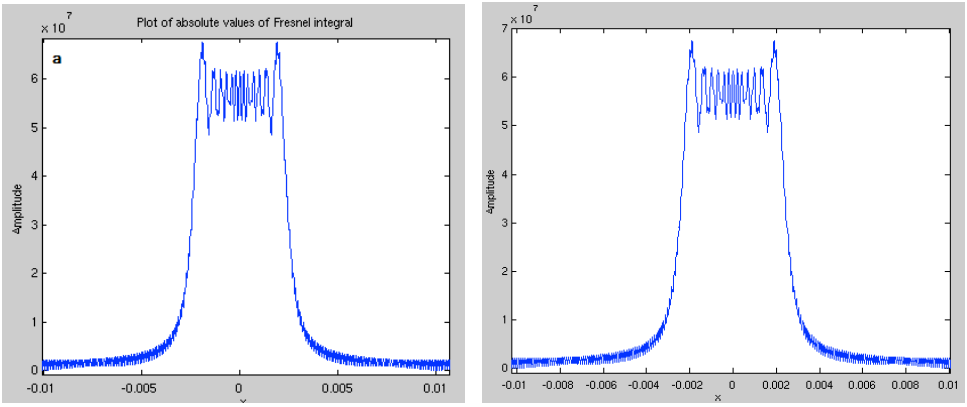


Figure 26: Fresnel diffraction integral calculations in 1D for F=2 using method 1

We can see that indeed the two different approaches to the convolution produce identical results. In both cases the convolution produced a periodic function, with the sections shown above being one period of each approach. We do indeed recognise these as 'Fresnel-like' diffracted field profiles, however, upon comparison with the calculation made using method 2 for an identical slit, illuminated and observed in the same way, we notice a discrepancy. Figure 27 shows the field distribution calculated using method 2 for the same input parameters.

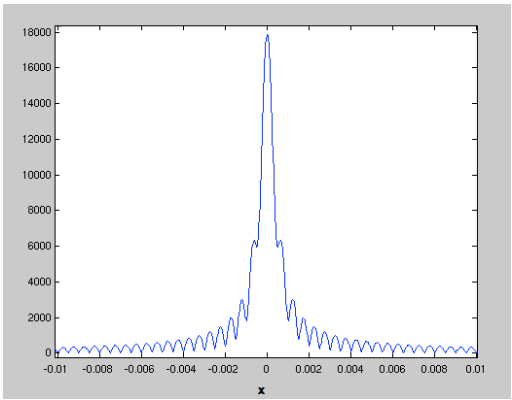


Figure 27: Fresnel calculation for F=2 using method 2

Clearly these results are contrary, and so we are left to determine which is valid. To aid with our decision, it is useful to look at the propagated field distributions for a range of distances, and to observe the behaviour of the calculation as Fresnel number changes.

It will be useful here to extend the model into two dimensions, so that the images can be compared easily with those produced using method 2. Again, this is fairly easily done, by simply using the two dimensional forms of the aperture transmission function, convolution kernel, and using the 2D forms of the corresponding MATLAB functions. Figures 28a-28e show propagated field

distributions to distances z increasing from 0.3m to 4m, where the diffracting aperture is circular of radius 1mm, and illuminated by 500nm light. In each case the original aperture was sampled with a resolution of 1/50 mm.

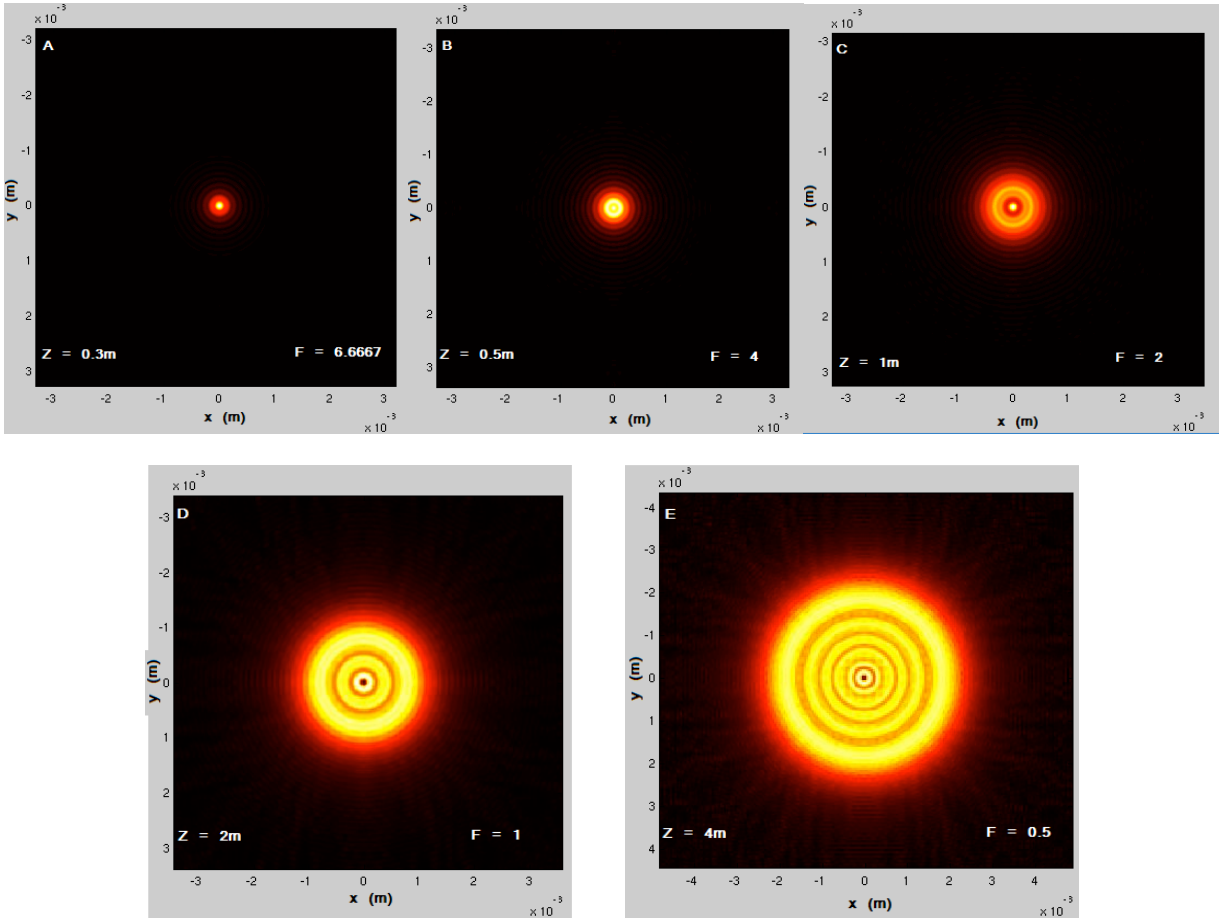


Figure 28: Variation of propagated field distribution with Fresnel number

These calculations are clearly going wrong somewhere- we are observing the patterns to be approaching the Fraunhoffer form as Fresnel number becomes larger! This is in contradiction to the predictions of the theory, and so we must conclude that there is some unforeseen problem that is arising from the way in which the Fresnel diffraction integral is being calculated. It is possible that there may be a numerical error arising from the numerical analysis of the convolution- the Fourier transforms involved in the convolution theorem approach are of course DFTs, and so the normal problems arise from the discreteness of data points. Problems with resolution also become apparent using this method of analysis. Figure 29 a-e shows the field distributions calculated for a 1mm slit, illuminated by 500nm light, with the diffracted field propagating distance of 1m, where the original aperture has been sampled with different resolutions. These correspond to a Fresnel number of 2, so we should in theory expect to see a Fresnel-like distribution.

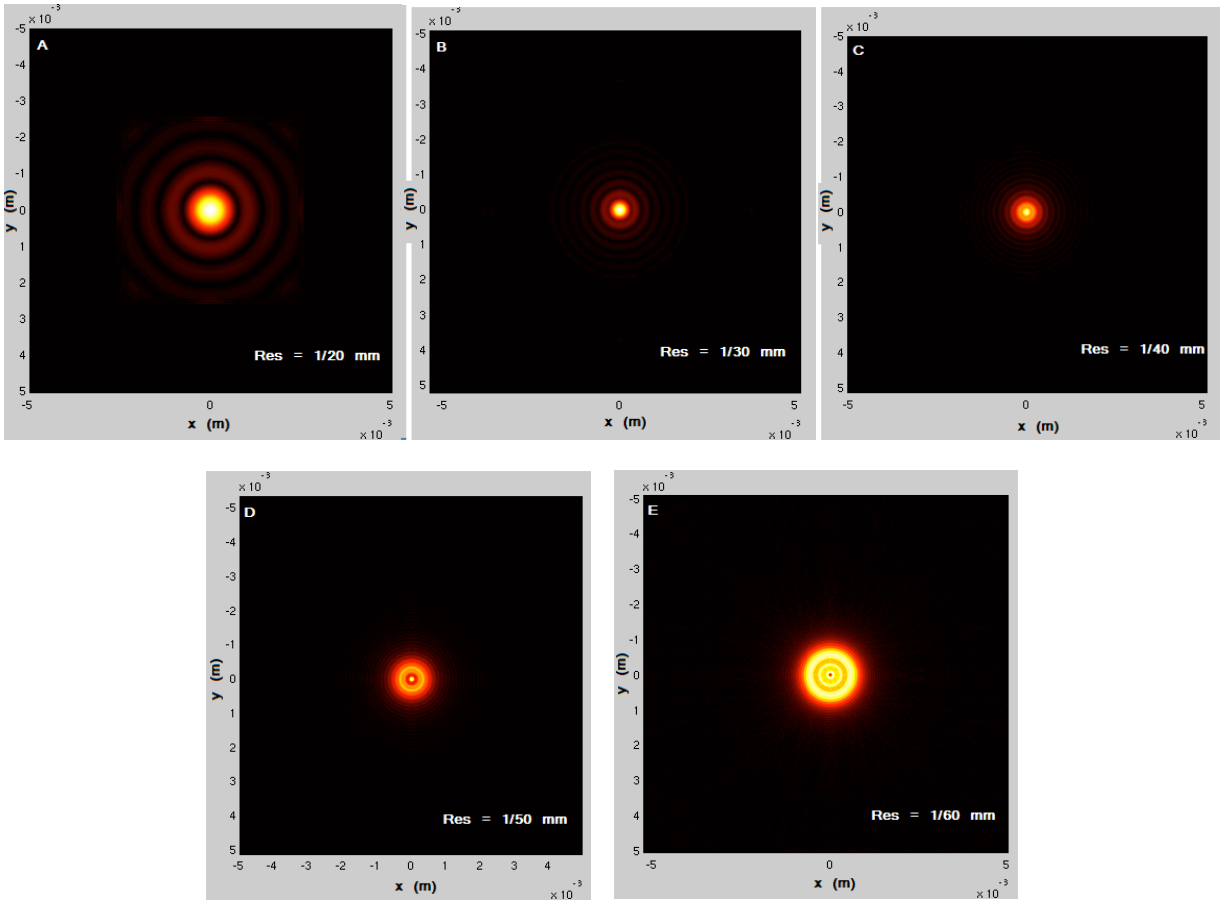


Figure 29: Variation of rendered field distribution with increasing resolution

This clearly shows that by formulating the Fresnel diffraction integral in this way, the resolution at which the aperture (or transmittance function) is sampled affects the rendered image. Therefore, we cannot rely on any of the calculations using this method. Clearly the propagated field distribution should not vary in this way. We should expect that with lower resolutions, we should observe the same distribution as that calculated with a higher sampling resolution, but rendered by our calculations in lower resolution- it should simply occupy a smaller fraction of the field of view in the propagated plane.

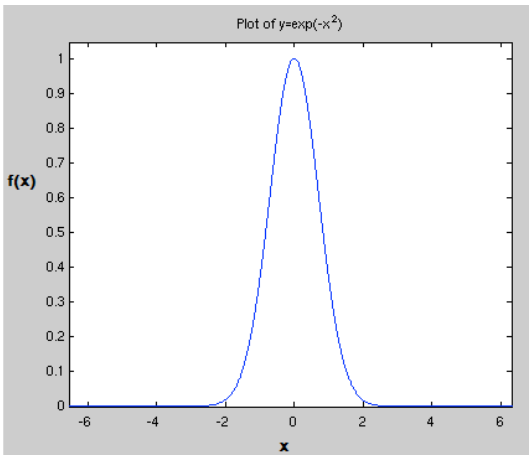
As a result of the discrepancies mentioned here, it is sensible to discard the convolution formulation of the Fresnel diffraction integral for our purposes, pending further investigation as to the reasons why it is seen to produce these strange results. It is likely that some of the problems may arise from the numerical analysis of the convolution. This seems a possibility, as we have seen that the degree of resolution is having a significant adverse effect on the calculations of the propagated field.

Possible sources of numerical error

Numerical methods of analysis are useful for evaluating continuous functions to which there are no analytic solutions. Though useful, however, they can only at best provide an approximation to the true solutions of the continuous function. Through discretization, numerical errors become apparent. These can arise from several sources, including round-off and truncation.

Since we are dealing with complex quantities, the phase of the different functions involved in the calculations is also important. With numerical analysis, phase errors occur in the data, and these can cause problems. An excellent example of how, for example, the discretization of the Fourier transform affects phase, is given by considering the Gaussian function: $f(x) = Ae^{\frac{-(x-B)^2}{2C^2}}$ for some real constants A,B,C. The Fourier transform of a Gaussian function is another Gaussian function.

For simplicity, consider the case of A=1, B=0, C=$\sqrt{0.5}$, so we get $f(x) = e^{-x^2}$. Figure 30 shows a plot of this function, and a plot of its phase.



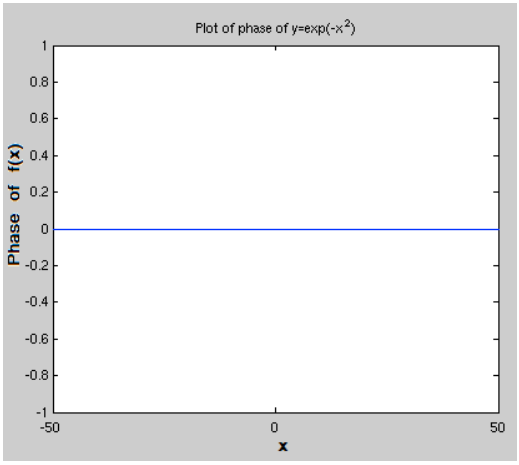
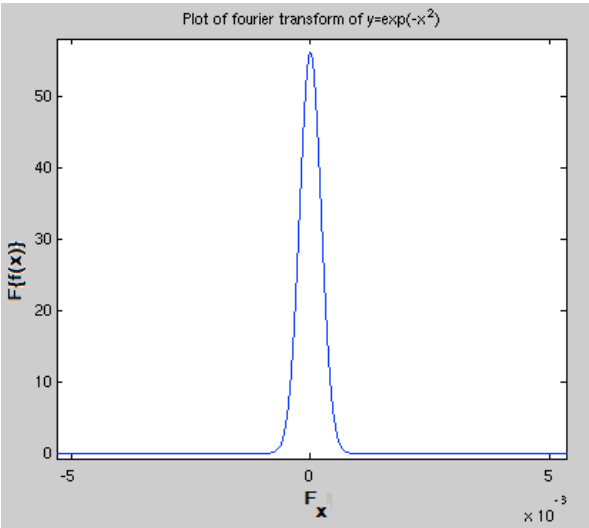


Figure 30a: Plot of f(x) **Figure 30b: Plot of phase pf f(x)**

It is seen that the Gaussian function has a constant zero phase. This is to be expected- there is no complex part of the function. However, if we now compute the DFT of f(x), we observe something different. A plot of this is shown in figure 31, along with the phase of the transformed function. The transform of the function can be worked out analytically, and is found to be: $\mathcal{F}(f(x)) = ACe^{-\frac{x^2}{2C^{-2}}}$.

For the f(x) we defined, this becomes $f(x) = s\sqrt{0.5}e^{-\frac{x^2}{4}}$, where s is a scaling factor equal to $1/N\Delta x$, arising from the DFT.



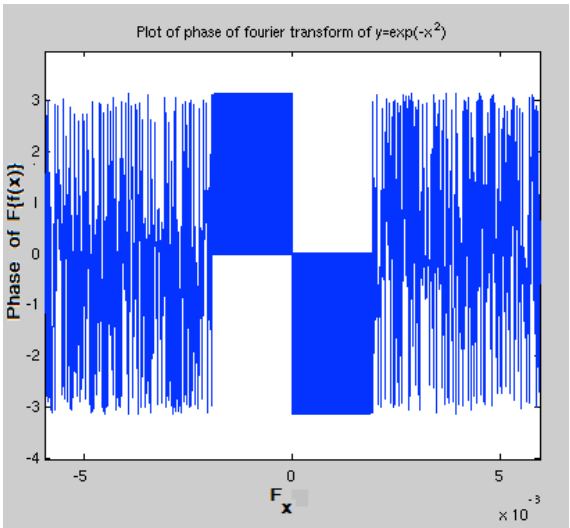


Figure 31a: Plot of DFT of f(x) **Figure 31b: Plot of phase of DFT of f(x)**

The DFT results in a complex values in Fourier space, which contain a phase. When the absolute values are evaluated, we find that we retain a Gaussian function in Fourier space. Its phase has a few distinct parts- we see that at frequencies corresponding to zero values of the transformed function, the phase oscillates very quickly between ±π. Where the transform is non-zero, we see that the phase oscillates between zero and ±π, with the phase flipping at the turning point of the Gaussian. Most of the noise seen in the signal is due to the FFT algorithm. This can be removed using the unwrap() function in MATLAB. The unwrapped phase distribution is seen in figure 32.

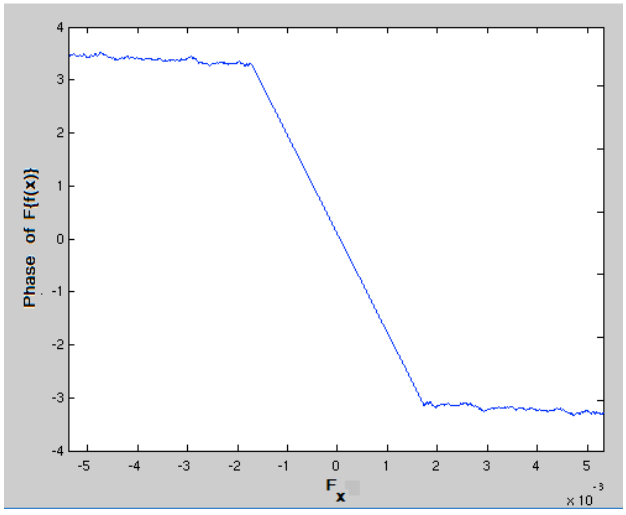


Figure 32: Unwrapped phase

The unwrap() function works by adding or subtracting multiples of 2π to the radian phase angles of a data array, wherever the phase jump between adjacent elements is greater than π. Through its use we are able to remove the noise in the original signal, however, we still observe a linear phase shift across the non-zero points of the function. This remains over every resolution and sample size tested- this shows us that this represents a phase shift of π between successive pixels in our generated image, and so is simply an artefact of the DFT.

Wave-optics analysis of coherent optical systems

Wave optics techniques are useful for predicting the effects of optical systems, a useful feature of the analysis being that diffraction effects are fully accounted for. In order for the program to be well adaptable to different systems, it is necessary that the program is designed so that the user is able to define an arbitrary combination of optical elements, and that it can calculate the effects of the total system on the wave field distribution in the output plane.

The important points to note on this subject are that different optical elements have particular effects on an incident wave field- for example a thin lens can be treated as a phase transformation of the incident field. The lens introduces phase factors to the wavefront, which represent the differential delay caused to it by the variation of the thickness of the lens at each point on its surface. This phenomenon is explained in detail by Ref[1].

Ref[1] provides a comprehensive introduction to this approach, and the reader is referred to this resource as the theory on which this aspect of the program development has been based.

Operator Notation

For mathematical simplicity, a useful method of analysis is through the use of an operator notation. The advantage of this method is that the effects of different optical elements can be represented by different operators, and the effects of an optical system on an incident wavefront can be represented by a combination of applications of these operators on the incident field. This is extremely useful to us in creating a program such as ours, in that the different operators can be programmed as functions in MATLAB, and we can design our simulation to allow the user to simply select a particular sequence in which these operators act on the incident wavefront to represent the effects of an arbitrary optical system.

Whilst there are many different operations that may be necessary to account for, here are outlined the basic and most important operations useful to us in our simple simulation program. In MATLAB, the programmer defines functions as follows:

Function [output arguments] = functionname(input arguments)

The input and output arguments can be numerous, and it is often useful in repeated application of these operators for the output of one to be the input of another that is subsequently applied. For example, as we are designing this program to simulate the evolution of an initially planar wavefront, we will want each operator to give the transformed wavefront as its output. This output will then be the input of the next operator applied to examine further evolution resulting from other elements.

As the wavefront propagates through the system, it will, as before, be represented by a matrix, which will be subject to the operations described below. This will be referred to from here on as the 'wave field matrix'. The output of each function defined will be this matrix, along with, if necessary, rescaled coordinates.

In this program we will be, and have been, in all cases considering the simple case of an incident monochromatic plane wavefront. For explicit details of the MATLAB code used to program the different functions, the reader is referred to the appendix.

1. Define Gaussian Beam profile [5][4]

Until now, for the sake of simplicity, we have been working with unit amplitude, planar wavefronts. It is now useful to define a function that allows the user to simulate the propagation of a beam with a Gaussian profile, since this is an accurate representation of the beam amplitude distribution in most practical situations. The form of the field distribution of a Gaussian beam is given by:

$$E(r,z) = E_0 \frac{w_0}{w(z)} exp\left(\frac{-r^2}{w^2(z)}\right) \exp(-ikz) \exp\left(\frac{-ikr^2}{2R(z)}\right) \exp(iG(z)) \quad (27)$$

Where E_0 is the amplitude (=E(0,0)), r is the distance from the beam axis (with $r^2 = x^2 + y^2$), z is the distance along the beam axis (z axis) from the beam waist, k is the wavenumber, w_0 is the waist (smallest beam size- e.g. from source) and we have the beam parameters:

Spot size (beam radius) at z: $$w(z) = w_0\sqrt{1 + \left(\frac{z}{z_R}\right)^2} \quad (28)$$

Where $z_R = \frac{\pi w_0^2}{\lambda}$ is the Rayleigh range (distance from waist at which spot size is $w_0\sqrt{2}$). w(z) is the distance from the beam axis at which the amplitude falls to 1/e its maximum value.

Wavefront radius of curvature: $$R(z) = z\left[1 + \left(\frac{z_R}{z}\right)^2\right] \quad (29)$$

Gouy Phase: $$G(z) = \arctan\left(\frac{z}{z_R}\right) \quad (30)$$

The required input arguments by the user are the wavelength, waist size, distance from waist, and the resolution. Figure 33a shows a Gaussian beam profile defined using this function- with a waist size of 3mm, resolution of 1/10mm, at a distance from waist of 100m. Figure 33b shows a beam profile using the same parameters, but at a distance from waist of 1000m, exhibiting the divergence of the beam with propagation distance.

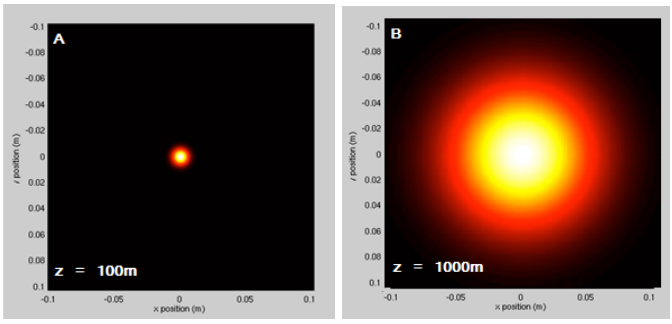


Figure 33: Gaussian beam profile

2. Pupil and aperture functions

It is essential for the program to be able to calculate the effects of apertures on an incident wavefront. The mathematics involved has already been discussed in length in earlier sections in this report. However, aperture transmission functions can also be used to represent the finite extent of lenses- in this case we would refer to them as pupil functions. With aperture and pupil functions we retain the use of the simplification that the function is valued unity at all points within the aperture or pupil, and zero elsewhere. In the case of pupil and aperture functions, the necessary inputs will be the wave field matrix and the characteristics of the aperture (e.g. radius/ width, height).

Other operations of interest are those representing the effects of amplitude and phase gratings- these are easily programmable as sinuosoidal phase and amplitude transmission functions. For such gratings the inputs will be the minimum and maximum transmission, and the number density of lines per unit length (spatial frequency). The amplitude transmittance functions are as follows:

Sinusoidal Phase grating: $$t_A(\xi,\eta) = \exp\left(\frac{im}{2}\sin(2\pi f_0 \xi)\right) rect\left(\frac{\xi}{2w}\right) rect\left(\frac{\eta}{2w}\right) \quad (31)$$

Sinusoidal Amplitude grating: $$t_A(\xi,\eta) = \left[\frac{1}{2} + \frac{m}{2}\cos(2\pi f_0 \xi)\right] rect\left(\frac{\xi}{2w}\right) rect\left(\frac{\eta}{2w}\right) \quad (32)$$

Notation is as has been used throughout. Here f_0 is the spatial frequency of the grating, m is the peak-to-peak deviation of the phase delay/ change of amplitude, and the factor of $rect\left(\frac{\xi}{2w}\right) rect\left(\frac{\eta}{2w}\right)$ is included to describe the finite extent of the grating [1]. This finite extent will be determined by the resolution the user chooses in which to render the grating. Figure 34 shows the transmittance function representing a sinusoidal amplitude grating defined in this way, with 100 lines per metre, peak-to-peak transmittance of 0.6, resolved to 1/10mm.

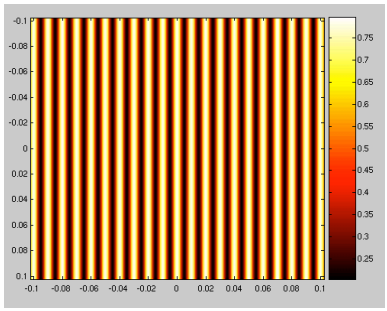


Figure 34: Transmittance function for sinusoidal amplitude grating

3. Multiplication by quadratic-phase exponential

Ref[1] discusses the effect of thin lenses on an incident wavefield- the effects can be represented by multiplication by a quadratic-phase exponential, which can accurately represent the differential phase delays incurred by the wavefield according to its point of incidence on the lens. This operation is useful in other situations too- for example a Fresnel diffraction operation is equivalent to multiplication by a quadratic-phase exponential, then a scaled Fourier transform, then another multiplication by a quadratic-phase exponential. We will see that there are relationships between many of these operators. It will prove easier for our purposes to define operators to include these composite effects for common events such as Fresnel diffraction; however, it is interesting to see how each can be made up of others.

Here, we will retain the notation used by ref[1]. The operator described above is denoted by $\mathbb{Q}$, and is defined as:

$$\mathbb{Q}[c]\{U(x,y)\} = e^{i\frac{kc(x^2+y^2)}{2}} U(x,y) \quad (33)$$

The parameters necessary for input are denoted in square brackets. Here c is an inverse length. For example, in the case of the phase factor introduced by transformation by a thin lens, the c parameter would be 1/f where f is the focal length of the lens. The inverse of this operation would be Q[-c].

4. Scaling by a constant

This operation is necessary, for example, where we wish to calculate the Fraunhoffer diffraction pattern produced by an aperture. We have seen that it is necessary to rescale the x and y coordinates when a Fourier transform is performed in order to retain quantitative physical significance in the results. This operator is denoted as $v[b]$, where b is a dimensionless constant. v is defined as:

$$v[b]\{U(x,y\} = |b|^{\frac{1}{2}}U(bx,by) \tag{34}$$

The inverse of $v[b]$ is $v[\frac{1}{b}]$.

5. Fourier Transformation

Fourier transforms are of course the core of Fourier Optics analysis. MATLAB of course has its own built-in FFT function, so we need not define an operator to perform this task. In order to make the computation easier, it is however useful to define an operator that will automatically perform a scaled Fourier transform by making use of 3.

6. Free space propagation

It is of great benefit to define such an operator, as our main interests here are the propagation of the wavefront through the optical system. This operator is denoted as $\mathcal{R}$ and defined by:

$$\mathcal{R}[d]\{U(x_1,y_1)\} = \frac{e^{ikd}}{i\lambda d}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}U(x_1,y_1)\exp\left(\left(\frac{ik}{2d}\right)[(x_2-x_1)^2+(y_2-y_1)^2]\right)dx_1dy_1 \tag{35}$$

Here d represents the distance of propagation, (x_1,y_1) are the coordinates in the initial plane, (x_2,y_2) are the coordinates in a plane a distance d to the right of the initial plane. The inverse of $\mathcal{R}$[d] is $\mathcal{R}$[-d].

With this collection of operators, we will be able to simulate the effects of a wide variety of optical systems. There exist a number of relations that connect these operators with one another, which can be useful when making analytic evaluations. However, as in this case we will be computing the operators, this isn't strictly relevant to our discussion. As a matter of interest, the reader is referred to ref[1] p.117 for further details.

Error Accumulation due to numerical methods

We have seen previously that errors arise in the wave field calculations upon performing Fourier transformations due to the discrete nature of our data. It seems likely then that repeated application of operators that make use of numerical methods of analysis, such as the discrete Fourier transform, will cause numerical errors to accumulate. Figures 35a-35d show the 2D wave field distributions calculated at distances d 1-4m with only one propagation step from z=0 to z=d. These calculations were based on a circular aperture of radius 1mm, resolution 1/50mm, illuminated by light of wavelength 500nm.

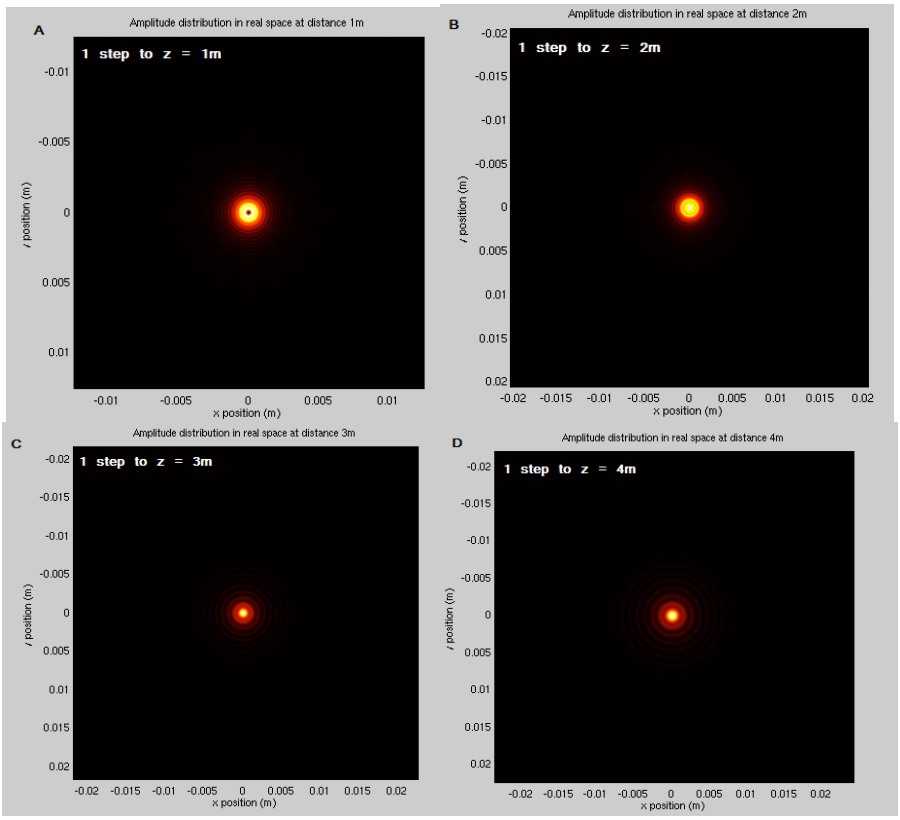


Figure 35: Single step propagation to z=dm.

Figure 36a-36d shows the fields calculated at d under the same circumstances but that the field has been propagated using operator 5 in *d* 1m steps.

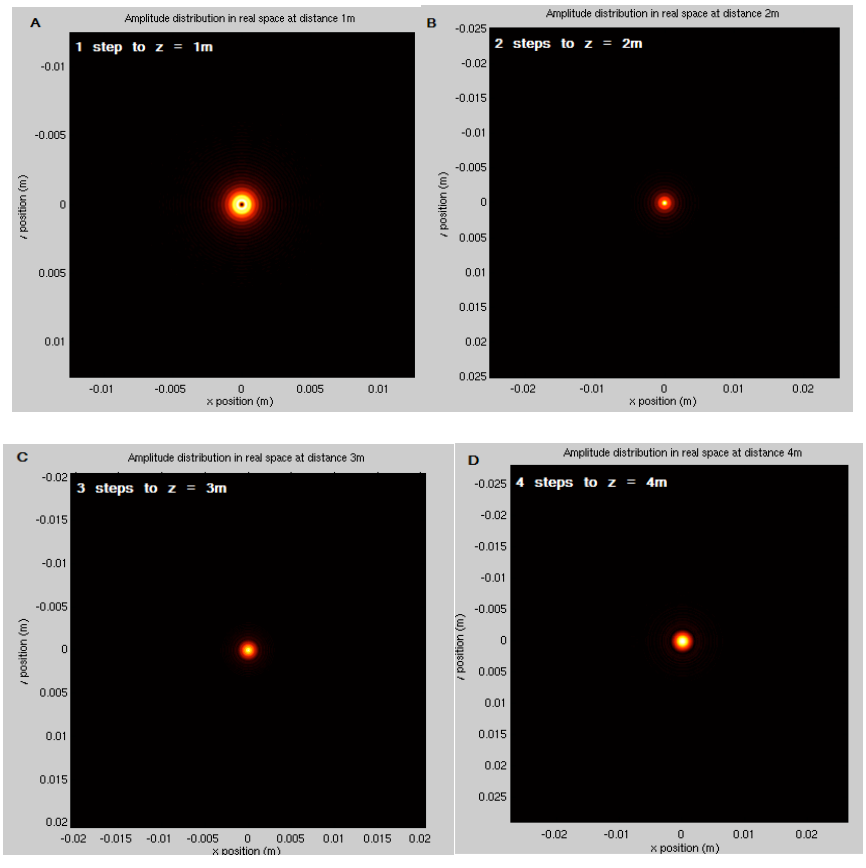


Figure 36: Propagation to z = dm in 1m steps

It can be seen that the field calculated in the step-by-step propagation departs from that calculated using single step propagation. In particular the distribution seems to tend faster towards a Fraunhoffer-like pattern, and the high frequency components of the distribution do not appear in the calculated wave field.

In order to investigate this effect more fully, a simulation in one dimension is necessary, as this allows us a larger sample size in one dimension with which we can look at the effects of resolution. A comparison was made of the wave field calculated at a distance d with one application of $\mathcal{R}$ and that calculated at a distance d by operating on the incident field d times, each time propagating 1m. In this case the aperture was of principal dimension 2mm, and again was illuminated by 500nm light.

In order to quantify the discrepancy between the one-step and step-by-step cases, it is useful to look at the difference between their normalised square integrals. We should expect, from Parseval's theorem, that in the absence of numerical error these ought to be the same, so the discrepancy between them gives a good indication of the degree to which their amplitude distributions are different.

It was thought that both the number of propagation steps would likely be an important factor in the resulting discrepancy. Figure 37 shows the variation in the discrepancy between one-step and step-by-step propagation at a resolution of 0.0001mm up to 50 propagation steps of 1m.

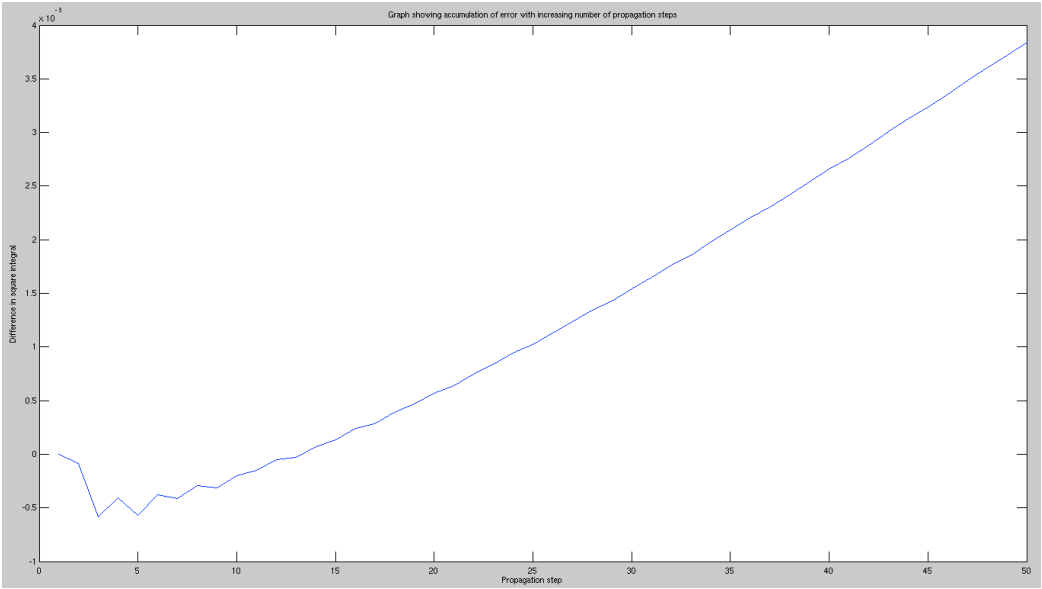


Figure 37: Accumulation of error with number of propagation steps

Figure 37 shows clearly that the size of the discrepancy becomes larger for a greater number of propagation steps. The discrepancy is calculated as one-step minus step-by-step- it can be seen that there is an initial discrepancy with the step-by-step case having a larger square integral. With repeated application of operator 5 we then find that the step-by-step case becomes smaller than the one-step square integral. The discrepancy is generally small, however, it must be noted that this discrepancy has been calculated on the basis of Parseval's theorem- a statement of conservation of energy. We can see that the error is highly significant, but more importantly is the shape of the propagated field distribution.

A useful method of comparison between an obtained and a theoretical, or ideal, data set is the Chi-square test [8]. This gives an indication as to the degree to which the obtained data set agrees with the expected set. The Chi-square statistic χ^2 is given by:

$$\chi^2 = \sum_{i=1}^{n} \frac{(y_i - \bar{y}_i)^2}{\bar{y}_i} \qquad (36)$$

Where n is the number of data points, y_i are the obtained data, $\overline{y_i}$ are the expected data. The Chi-square statistic has several statistical uses [8], but for our purposes can be used as a measure of goodness-of-fit. Figure 38 shows the variation of χ^2 with number of propagation steps.

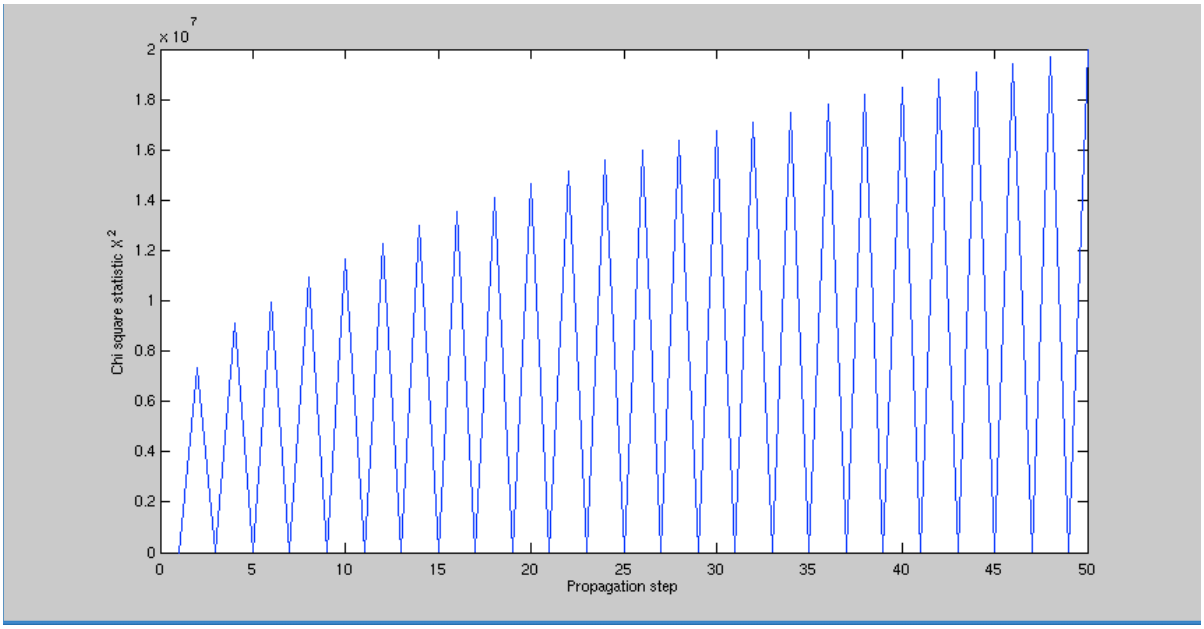


Figure 38: Graph showing variation of χ^2 with number of propagation steps

It can be seen that the χ^2 value oscillates between high and low values. This is due to the repeated application of the DFT to the wave field. Consider a sin function- figure 39a shows the sin function, and 39b twice Fourier transformed.

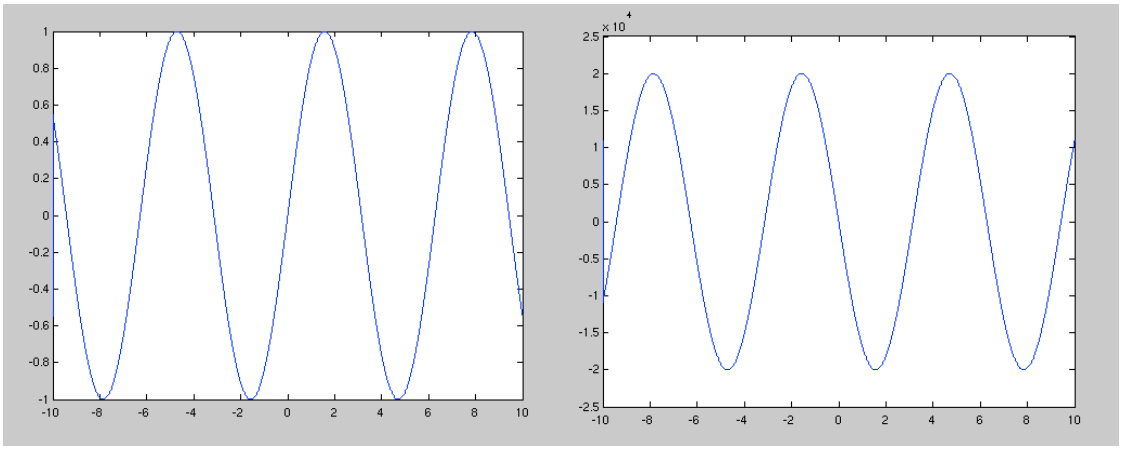


Figure 39a: Plot of sin(x) **Figure 39b: Plot of $\mathcal{F}[\mathcal{F}\{sin(x)\}]$**

It can be seen that the sin(x) function twice transformed yields a sin(-x) function, together with a scaling factor introduced by the transform. Fourier transforming a function of real space co-ordinates twice so that it ends up back in real space co-ordinates results in reflection in the real space co-ordinate axes. As such we can see that we cannot compare adjacent propagation steps, as the field distribution is reflected in the y axis in this way. If we then consider only the odd or even propagation steps- as shown in figures 40a and 40b:

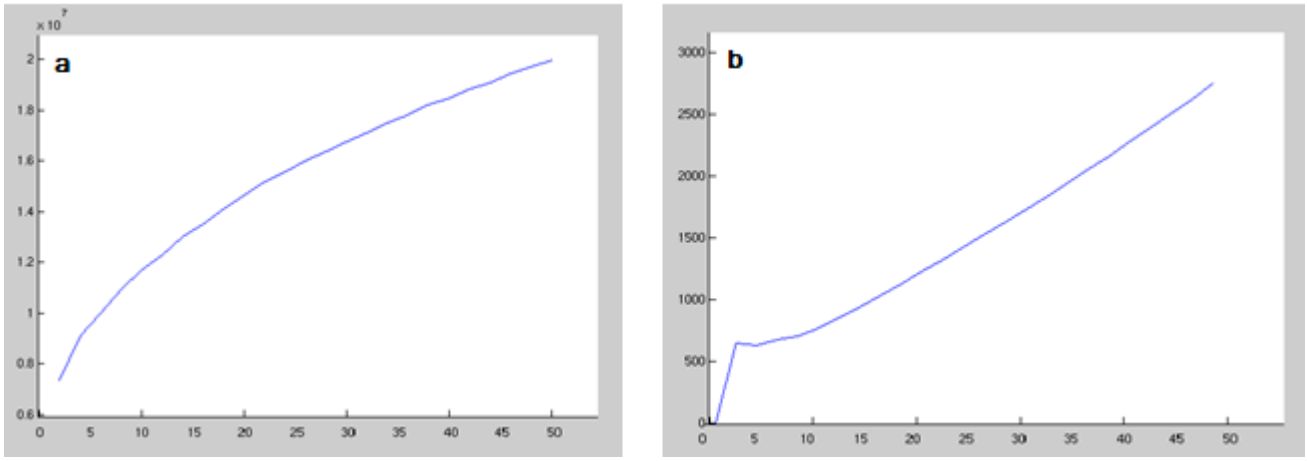


Figure 40: χ^2 variation with even steps (a), odd steps (b)

It can be seen that the χ^2 value increases with propagation steps in each case. Thus we clearly see that the field distribution accumulates numerical errors with greater number of operations, and this causes it to depart from its theoretical form. This presents an important limitation of the simulation we are able to perform. In practice, we would not require our simulation to propagate the wave field one step at a time- if we know that the beam has to propagate a certain distance, it is simple to calculate its expected field distribution at that distance by using single step propagation. However, in synchrotron beamlines, the beam often traverses a complex optical system, comprised of many different elements. We may find that with a more complicated optical system, the accuracy to which we can predict the characteristics of the beam profile in the output plane will be reduced.

Focusing mirrors

One of the main obstacles in synchrotron beamline design is predicting the scattering effects of focusing mirrors on a reflected beam. There is not a well established theory to account for the transformations introduced by small surface structures of a mirror. It is therefore of great interest to attempt to model the effects of focusing mirrors on a reflected wave field.

Several techniques have been developed to map the surface structure of high quality optics, such as those used in synchrotron beamlines [12-15]. Many such techniques are able to record variations in surface height profile to nanometre precision and micrometer resolution. No surface is perfectly smooth, and such height profiling techniques reveal surface irregularities in even the highest quality optical elements. Whilst these irregularities are extremely small (on the nanometre scale), the short wavelengths of synchrotron light mean that even tiny surface structures can cause the beam to strongly scatter.

Reflection can cause the phase of the incoming wave to be transformed, depending on the nature of the media that the reflecting interface separates. If the interface separates a dielectric and conductor, the phase of the wave is unaltered. If the interface separates two dielectrics, with the second dielectric being of higher refractive index, the phase of the reflected wave will be inverted. On the other hand, if the wave is propagating from a high refractive index medium to one of lower refractive index, the phase will be retained [18].

In order to make an attempt to model the effects of such small-scale height deviations, we need to consider the effects of reflection on the propagation of light. Reflection occurs at an interface between two different media- the ray changes propagation direction so that it remains in the original medium. Reflection of a wave at an interface normally results in a *specularly* reflected component and a *diffusely* reflected component- though some surfaces may exhibit only one type of behaviour. In specular (mirror-like) reflection, the light is observed to obey the three ‘laws of reflection’ [18].

1. The incident ray, reflected ray, and surface normal all lie in the same plane.

2. The angle of incidence equals the angle of reflection.

3. Light paths are reversible.

Diffuse reflection results in a scattering of the wave in all directions, meaning the image is lost. It is a common misconception that diffuse reflection is caused generally by surface roughness- surface structures can contribute, but diffuse reflection is normally a result of different parts of the incoming wave being scattered by different centres, at different depths in the surface [18]. This causes the observed variable angle reflection. Some materials do not exhibit diffuse reflection in this manner- for example light cannot penetrate through a metal surface, so any diffuse component of a reflected wave is likely due to small scale surface structures. Figure 41 shows how height deviations can cause light to be reflected diffusely. Consider two initially parallel rays incident upon some irregular surface- both travelling at an angle **a** with respect to the normal to the mean surface height. If the small scale structures were to be ignored, we would expect the reflected rays to propagate away from the mean surface at an angle **b**, equal to angle **a**, again measured with respect to the normal to the mean surface height. However, when the small structures are taken into consideration, it can be seen that the angles of incidence must in fact be taken into account relative to the normal to the surface at the point of reflection- thus rays striking the surface at different points will likely be reflected in different directions. This will cause a wavefront to become distorted upon reflection.

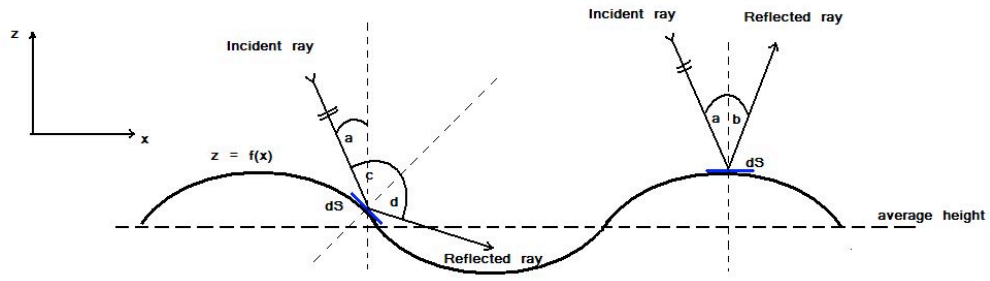


Figure 41: Effects of surface irregularities on reflected wavefront [17].

For our purposes, however, since surface irregularities are very small in the high quality optics used in synchrotron beamlines, we can neglect the effects of diffuse reflection and consider only the specular case. Slope deviations on the mirrors' surfaces are normally found to be on the microradian scale, and as such their scattering effects are negligible.

Despite the apparent simplicity of specular reflection, height deviations do prove troublesome. As different parts of the incident wavefront are reflected from different points, and therefore heights relative to each other, an initially planar wavefront becomes distorted as different rays incur relative path and phase differences. Figure 42 shows how path differences between adjacent rays can be adopted upon reflection from an irregular surface.

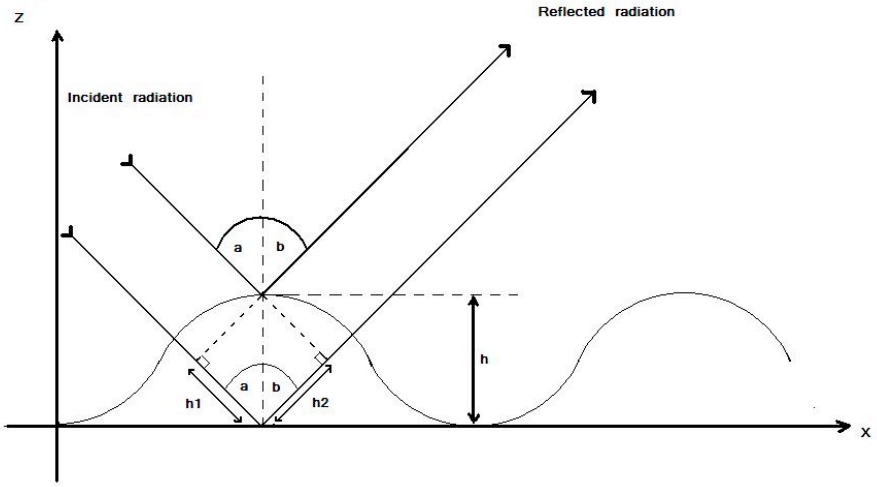


Figure 42: Accumulation of path differences as a result of reflection from irregular surface [16]

The path difference adopted by two rays striking the surface at different heights is given by equation 37:

$$\Delta x = \Delta h(\cos(a) + \cos(b)) \qquad (36)$$

In the case of specular reflection this of course reduces to $\Delta x = 2\Delta h \cos a$. The effect of this path difference on the relative phase difference between the two rays is dependent on their wavelength. These phase differences cause a loss of image quality upon reflection of the wavefront.

By compiling a surface profile of the mirror, it is possible therefore to predict the transformation effects of the small scale height deviations.

Modelling the transformation effects of a focusing mirror

We have seen that the phase transformation effects of a mirror with small scale surface deviations are dependent upon the magnitude of the height variations, and the angle of incidence of the wavefront. Each individual component of the collective wavefront will adopt a different relative phase delay, dependent on the position at which it strikes the optical surface. As part of this investigation, a 1D slope profile was measured of a high quality synchrotron optic- details of which can be found in ref [15]. We now consider how the obtained surface map of the optic can be used to predict its scattering properties.

Upon specular reflection from a surface, the amplitude of the incident wave components will be unchanged, but they are likely to adopt relative phase changes. As such, we can consider the effect of the mirror on the wavefront as causing a phase transformation. We are considering the case of specular reflection relative to the nominal shape of the mirror [12], and so the direction of propagation of the reflected wavefront away from the mirror can be easily determined from its known incident angle, and the nominal shape of the mirror (focusing mirrors are often torroidal or spherical in shape in order to focus the incoming beam to a particular specification). Thus, in order to fully characterise the effects of the mirror, we need only consider the phase transformation that is picked up by the wavefront.

We have seen previously that a sinusoidal phase grating, which causes an incoming wave to incur phase delays but without change in amplitude, can be characterised by a transfer function, given in equation 31. A simple but useful way of modelling the phase transformation effects of a mirror is to consider the mirror as the combined action of a number of such sinusoidal phase gratings. By taking the Fourier spectrum of the height error profile, we can obtain the combination of phase gratings necessary to model its effects- the Fourier spectrum provides the different spatial

frequencies of the gratings, along with the height deviations by which they impart phase delays on the incoming wavefront.

Figure 43 shows the height error profile measured for the optic- this was calculated as the difference between the measured height profile and an ideal profile, found by fitting a second order polynomial to the height data, and subtracting this from the real data.

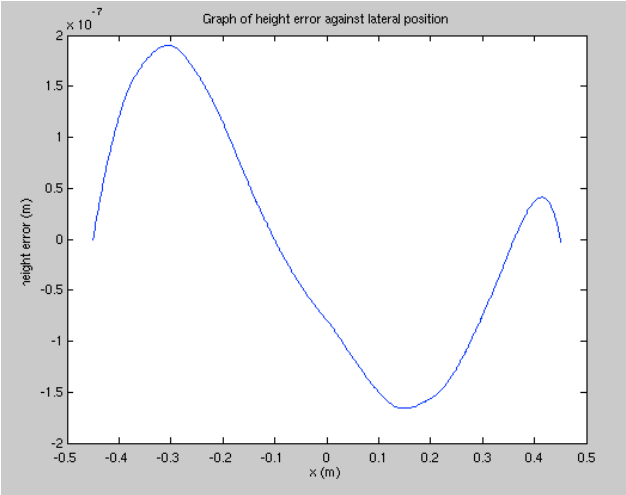


Figure 43: Height error map of measured mirror

A DFT was then performed on this height error map using the FFT routine in MATLAB, yielding the Fourier spectrum, which is shown in figure 44, plotted on a logarithmic scale.

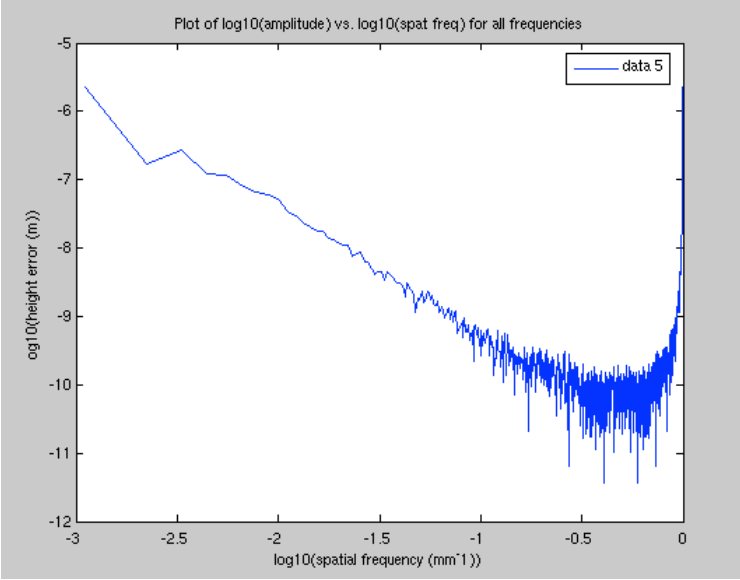


Figure 44: Fourier spectrum of height error map

It can be seen from figure 44 that the signal becomes very noisy in the mid-high frequency range. It seems that the millimetre precision afforded by the Diamond-NOM [15] used to measure the slope profile of the optic was insufficient to resolve these frequency components of the Fourier spectrum. This is a common problem, and often in order to obtain a full frequency spectrum of such a mirror, it is necessary to use different instruments to measure the low, mid, and high frequency ranges. As such, we are left with an incomplete picture of the surface profile, and are going to be incapable of accurately modelling its effects on a wavefront. However, the low frequency part of the spectrum is acceptable, so we can at least demonstrate how these can be used to model the transformation effects of the mirror. We can identify an apparently linear relationship between the logarithmic frequencies and height error magnitude at low frequencies. This is shown in figure 45:

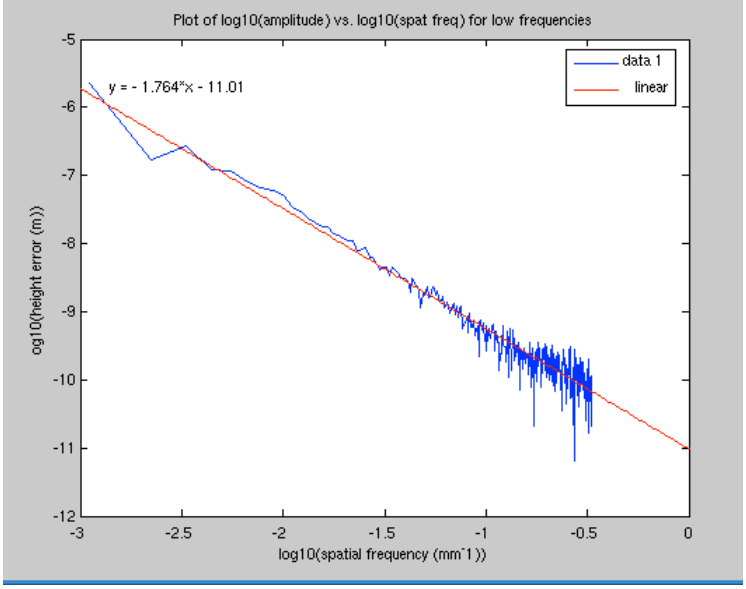


Figure 45: Low frequency spectrum

Using the basic fitting tools in MATLAB, it was also possible to determine the relationship between the height error and spatial frequencies, which will be useful in enabling us to model the behaviour of a mirror. We have now obtained the frequencies of the phase gratings that will be used in the simulation, so we can now consider their effects. Equation 31 gives the transfer function for a sinusoidal phase grating- the m parameter given in this equation, representing the peak-to-peak phase variation imparted by the grating can be formulated as:

$$m = \frac{2\pi}{\lambda} h(cos\theta_i + cos\theta_r) \qquad (37)$$

Where θ_i is the angle of incidence of the wavefront, and θ_r is the angle of reflection- these will of course be equal in the case of specular reflection from the nominal shape of the optic- where the slope effects of small scale structures are neglected. h here is the relative height of the optic at the point of reflection, consisting of the Fourier spectrum of the height profile- a series of sinusoidally varying height functions of different frequencies and amplitudes. The wavelength λ here refers to the spatial wavelengths of the Fourier components of the optic height profile, not of the incident radiation.

Thus we now arrive at the final form of our approximation of the phase transforming effects of a focusing mirror. It consists of a multiplication of the effects caused by a series of sinusoidal phase gratings, the frequencies and peak-to-peak changes being given by the Fourier spectrum of the optical height profile. Now we can attempt to extend the computer model to account for the effects of such optical elements.

For the sake of simplicity, it was decided to restrict the attempt to model the mirror to one dimension. We aim here to demonstrate the method, and this part of the simulation has no practical ambitions. To extend to two dimensions is trivial, and the common obstacles imposed by computer power limitations and numerical methods of analysis will recur again.

In order to compute the phase changes using MATLAB, the program designed for the purpose first asked the user to define the angle of incidence and the wavelength of the incident radiation. In the examples shown below, the height map of the optic measured previously [15] has been used in the calculations.

The calculation requires that the phase transformation caused by each sinusoidal phase grating is determined, in terms of its position along the optic, and then that each of these are multiplied together. It was most appropriate to compute this problem by constructing a matrix whose dimensions represented the frequency of the phase grating, and position on the optic. This matrix was then transformed using the transfer function (equation 31). Each row of the matrix then

represented one phase grating. These rows could then be multiplied together to give the overall phase transformation given by the mirror. Figure 46 shows graphs of the phase transformation introduced by the different frequency components of the mirror height profile when illuminated by hard (0.1nm) and soft x rays (10nm), at an incident angle of 6.2822 radians (i.e. grazing incidence).

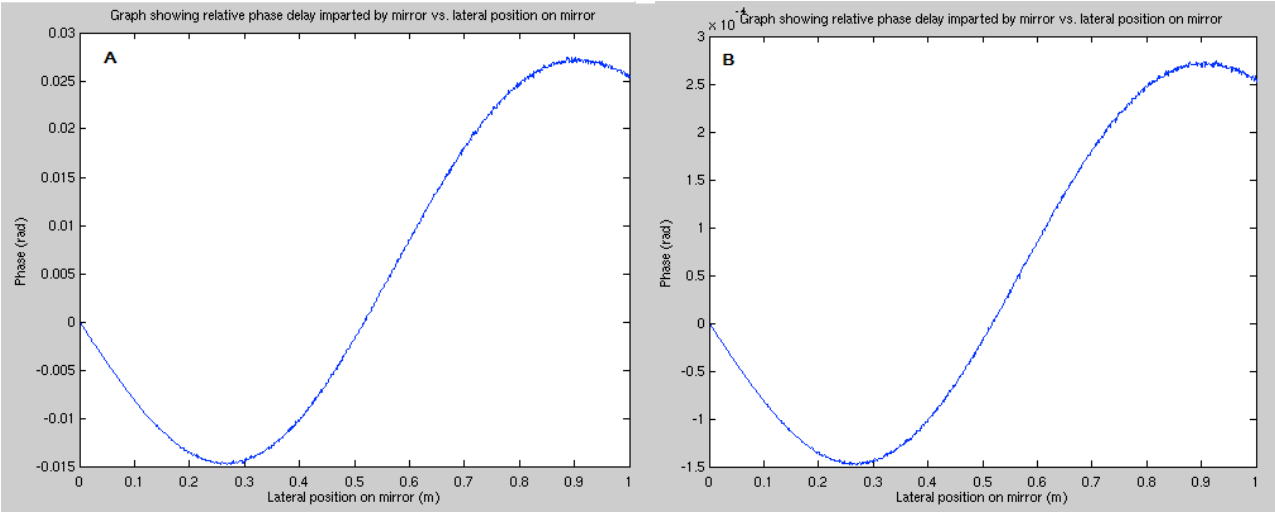


Figure 46: Phase transformations imposed for hard (a) and soft (b) X rays by mirror

In order to determine the validity of these results, we should compare with the height error profile obtained previously [15]. This is shown in figure 47.

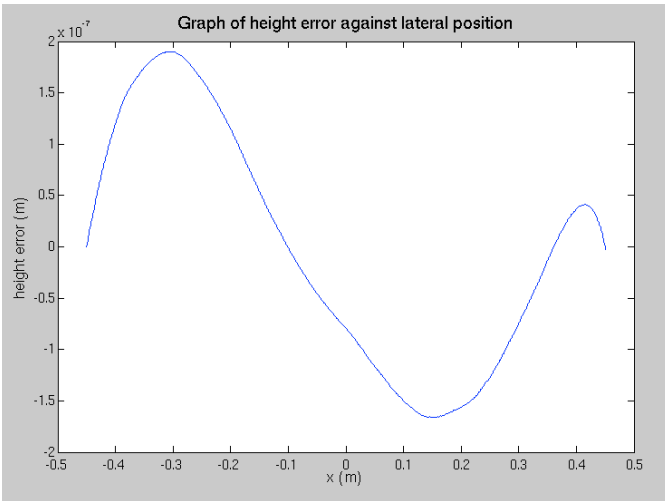


Figure 47: Height error profile of mirror

We see, as expected from the comparison, the phase varying in an almost sinusoidal manner along the optic. We do indeed expect the phase profile to be complementary to the height profile, so it looks as though the method described may be appropriate for modelling the effects of mirrors. We do see some discrepancies between the height profile and phase delay profiles- this is due to the fact that we have only analysed the low frequency components of the height Fourier spectrum. We also used a linear relationship which was fitted to the data, so strictly speaking figures 46 and 47 are not entirely equivalent. As stressed before, however, the purpose here is simply to demonstrate a method.

It can be seen that the phase delays incurred by hard X rays are greater than those incurred by soft X rays- this is to be expected, as obviously shorter wavelength radiation will pick up greater path and therefore phase differences upon reflection.

It would be interesting to see how the higher frequency components of the height profile affect the phase spectrum. Unfortunately the data obtained using the NOM [15] was not of a resolution that will permit this. An essential extension of this investigation would be to obtain a good set of mid and high spatial frequency data, and to examine their phase transformation effects. This would be

a good way to verify whether this modelling method provides a good representation of the mirror transformation. We should expect that the lower frequencies should contribute greater phase delays to the wavefront- this behaviour is something that would vindicate this modelling method if it were found to so predict.

3. Conclusions

It has been seen that Fourier optics provides useful methods by which to predict the transforming effects of different optical elements, and propagation of a wavefront through an optical system. A computer model has been developed that is able to calculate expected forms of wavefronts after propagation, or after a transformation caused by some optical element, and its predictions are found to agree well with experiment. Some discrepancies have, however, been observed. In particular, one of the diffraction experiments performed in the lab was seen to disagree with the prediction of the simulation. A dark central spot was observed, but not predicted. This warrants further investigation- the patterns produced in these practical experiments were quite noisy- some attempt may be needed to account for some of this noise, as it may well have contributed to the observed dark central spot, and indeed this may not be the famous 'Poisson's spot' [1]. Despite this discrepancy, all of the other experiments agreed well with the simulated cases.

Another problem that has presented itself in the simulation techniques is the errors that accumulate in the simulated wavefield due to the numerical methods of analysis that are used in the calculations. Unfortunately, these are problems that are impossible to eliminate. The accuracy of the simulations that can be performed using the model are strictly limited by the computing power available to us. With higher sampling resolutions we are able to more accurately predict the behaviour of the field. However, maintaining the same field of view with a higher resolution means that the time taken to compute the calculations increases exponentially. Calculations very quickly become intractable. This is an unavoidable fact. Where a very high accuracy is required, extremely powerful computers with large memories are essential.

A potential method for modelling the effects of small scale surface structures on a mirror on a reflected wavefront has also been suggested. There is room for a lot of further investigation of this method, necessary to its verification. Further investigators should focus on examining its ability to predict the phase transforming properties of mid and high spatial frequency components of the mirrors height profile. The instrument used to measure the height profile of the simulated mirror in a complimentary investigation [15] was only capable of millimetre resolution, and as such only the low frequency components could be modelled. Using other instruments to measure these other frequency components, and using stitching techniques, a high spatial resolution and large field of view can both be maintained [19]. This would provide a much better surface profile to be used with and investigate the modelling methods.

References

[1] J.W. Goodman- *Introduction to Fourier Optics. 3rd ed-* Roberts and Company Publishers 2005

[2] MathWorks website- www.mathworks.com.

[3] Double slit experiment Wikipedia page- http://en.wikipedia.org/wiki/Double-slit_experiment.

[4] E. Hecht- *Optics: International Edition 4th ed-* Addison Wesley 2002

[5] B.E.A Saleh, M.C Teich- *Fundamentals of Photonics*- Wiley 2007

[6] Online Fourier Optics guide- http://www.fourieroptics.org.uk/

[7] Nyquist-Shannon Sampling Theorem Wikipedia page- http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem

[8] Pearson's Chi-square test wikipedia page- http://en.wikipedia.org/wiki/Pearson%27s_chi-square_test

[9] Stanford University CCRMA website- https://ccrma.stanford.edu/~jos/mdft/Existence_Fourier_Transform.html

[10] Fourier Optics Wikipedia page- http://en.wikipedia.org/wiki/Fourier_optics

[11] LINOS photonics website- http://www.linos.com/pages/no_cache/

[12] *'The Diamond-NOM: A non-contact profiler capable of characterizing optical figure error with sub-nanometre repeatability'* S. Alcock et al., Nuclear Instruments and Methods in Physics Research A 616 (2010), p. 224-228

[13] '*Calculation of the power spectral density from surface profile data*' J. Elson, J. Bennett., Applied Optics, Volume 34, Issue 1, pp.201-208 (1995)

[14] '*Calculation of the Surface Topography of Optical Components from the Power Spectral Density- a Comparison of Different Approaches*' T. Owen, G.Ludbrook, S. Alcock. Diamond Light Source (Unpublished) (2009)

[15] '*Determination of optical surface profile using power spectral density'* D.Rafferty, S. Alcock, R. Dockree, 2010 Diamond Light Source (Unpublished)

[16] '*Non-paraxial scalar treatment of sinusoidal phase gratings'* J.E. Harvey, A. Krywonos, D. Bogunovic, JOSA A, Vol. 23, Issue 4, pp. 858-865 (2006)

[17] '*Polarized light reflection from strained sinusoidal surfaces'. B. Schulkin et al.,* Applied Optics, Vol. 42, Issue 25, pp. 5198-5208 (2003)

[18] Reflection Wikipedia page- http://en.wikipedia.org/wiki/Reflection_(physics)

[19] '*Large Field of View, High Spatial Resolution Surface Measurements*' J. Schmit, J.C. Wyant, Int J. Mach Tools Manufact. Vol 38, Nos 5-6, pp. 691-698 (1998)

Appendix- MATLAB code for various programs

1D Fresnel and Fraunhoffer Diffraction

```
% Program to calculate 1 dimensional Fresnel and Fraunhoffer diffraction
% patterns for single rectangular slit

clear all
close all

disp('THIS PROGRAM CALCULATES FRESNEL DIFFRACTION PATTERNS- VALID FOR ALL DISTANCE SCALES.
WHERE FRESNEL NUMBER << 1 CAN USE FRAUNHOFFER APPROX')


% Define diffraction properties

wavelength=input('What is the wavelength of the light used to illuminate the diffracting aperture (nm)?          ');
width=input('What is the aperture width (mm)?          ');
distance=input('What is the distance from aperture to screen (m)?          ');
wl=10^-9;
wdth=10^-3;

% Calculation of Fresnel number

F=((width*wdth)^2)/((wavelength*wl)*distance);

disp(['Fresnel number = ',num2str(F)])

% Determination of Fresnel or Fraunhoffer diffraction

if F<=0.1
   G=input(' Compute Fresnel (1) or Fraunhoffer case (2)?      ')
else
   G=1;
end



%%              %Fraunhoffer case

if G==2

   x=[-10:0.005:10]*wdth;
   y=abs(x)<=(width*wdth)/2;
   scaling1=(wavelength*wl)/((x(2)-x(1))*(length(x)-1));
   scaling2=(x(2)-x(1));
   x1=scaling2*x;
   endx=[-((length(x)-1)/2):((length(x)-1)/2)]*(wavelength*wl)/((length(x)-1)*scaling2);

   yscal=sum((y.^2)*(x(2)-x(1)))/sum((abs(fft(y))).^2)*(endx(2)-endx(1)));

   k=(2*pi)/(wavelength*wl);
   z=distance;

   h2=((exp(i*k*z))/(i*(wavelength*wl)*z))*(exp(((i*k)/(2*z))*(endx.^2)));

   figure(1)      % Transmittance vs x

   plot(x,y)
   title(['Rectangular slit of width',num2str(width),'mm'])
   ylabel('Transmittance')
   xlabel('x position (m)')

   figure(2)         % Unscaled

   F1=fft(y);
   FFS1=h2.*fftshift(F1);
   plot(endx,abs(FFS1*sqrt(yscal)));
```

```
    title(['Cross section of Fraunhoffer diffraction pattern produced by rectangular aperture of width',num2str(width),'mm
(unscaled)'])
    ylabel('Relative amplitude')
    xlabel('Angular distribution (unscaled)')

    figure(9)          % Rescaled


    plot(endx,abs(((FFS1).^2)*yscal))
    title(['Cross section of Fraunhoffer diffraction pattern produced by rectangular aperture of width',num2str(width),'mm
(rescaled)'])
    ylabel('Intensity')
    xlabel('Angular distribution (radians)')

    figure(10)

    posendx=z*tan(endx);
    plot(posendx,abs(((FFS1).^2)*yscal))
    title(['Cross section of Fraunhoffer diffraction pattern produced by rectangular aperture of width',num2str(width),'mm
(rescaled)'])
    ylabel('Intensity')
    xlabel('Spatial distribution (m)')



end

%%              %  Fresnel Case

if G==1
    tic
   x=[-10:0.000005:10]*wdth;
    y=abs(x)<=(width*wdth)/2;

    figure(4)       % Transmittance vs x

    plot(x,y)
    title(['Rectangular slit of width',num2str(width),'mm'])
    ylabel('Transmittance')
    xlabel('x position')

    k=(2*pi)/(wavelength*wl);
    z=distance;




    scaling1=(wavelength*wl)*z/((x(2)-x(1))*(length(x)-1));
    scaling2=(x(2)-x(1));
    x1=scaling2*x;
    endx=[-((length(x)-1)/2):((length(x)-1)/2)]*scaling1;



    h2=((exp(i*k*z))/(sqrt(i*(wavelength*wl)*z)))*(exp(((i*k)/(2*z))*(endx.^2)));

    h1=((exp(((i*k)/(2*z))*x.^2)));
%
    fres11=(ifftshift(ifft(fft(y).*(h1))));       % Convolution attempt by  computing kernel in fourier space- doesn't work!

    fres1=ifftshift(ifft((fft(y)).*(fft(h2))));     % Convolution kernel in real space- does work but lack of resolution in real
space%

    yscal=sum(((y.*abs(h1)).^2)*(x(2)-x(1)))/sum((abs(fft(y.*h1)).^2)*(endx(2)-endx(1)));


    fres2=fftshift(fft(y.*h1))*sqrt(yscal).*h2;
```

```
%%

%    h1=exp((i*pi*z*(wavelength*wl))*(x1.^2));

%    figure(4)              % Convolution attempt by using fourier space kernel
%
%    title('Method 1')
%
%    subplot(2,5,1)
%
%     plot(x,y)
%    title(['Rectangular slit of width',num2str(width),'mm'])
%
%
%    subplot(2,5,6)
%
%    plot(x,angle(y))
%    title(['Phase over rectangular slit of width',num2str(width),' mm'])
%
%
%    subplot(2,5,2)
%
%    plot(x1,fftshift(abs(fft(y))))
%    title('Fourier transform of aperture')
%
%
%    subplot(2,5,7)
%
%    plot(x1,unwrap(angle(fft(y))))
%    title(['Phase of fourier transform of aperture'])
%
%    subplot(2,5,3)
%
%    plot(x,abs(h1))
%    title(['PLot of Fresnel propagator'])
%
%    subplot(2,5,8)
%
%    plot(x,unwrap(angle(h1)))
%    title('Phase of Fresnel propagator')
%
%    subplot(2,5,4)
%
%    plot(x,fftshift(abs(fft(h1))))
%    title('Fourier transform of Fresnel propagator ignore!!!!')
%
%    subplot(2,5,9)
%
%    plot(x,unwrap(angle(fft(h1))))
%   title('Phase of Fresnel propagator ignore¬!!!!')
%
%
%    subplot(2,5,5)
%
%
%    title('Convolution of aperture field and propagator')
%    plot(endx,ifftshift(abs(ifft((fft(y)).*(h1)))))
%
%    subplot(2,5,10)
%
%    title('Phase of Convolution')
%    plot(endx,unwrap(angle(ifftshift(ifft((fft(y)).*(h1))))))
```

```
%
%

%%

%  h3=((exp(i*k*z))/(i*(wavelength*wl)*z))*(exp(((i*pi)/((wavelength*wl)*z))*endx.^2));
%    h3=(exp(((i*pi)/((wavelength*wl)*z))*endx.^2));

%    figure(5)          % Kernel in real space
%
%
%     subplot(2,5,1)
%
%     plot(x,y)
%    title(['Rectangular slit of width',num2str(width),'mm'])
%
%
%    subplot(2,5,6)
%
%    plot(x,angle(y))
%    title(['Phase over rectangular slit of width',num2str(width),' mm'])
%
%
%    subplot(2,5,2)
%
%    plot(x1,fftshift(abs(fft(y))))
%    title('Fourier transform of aperture')
%
%
%    subplot(2,5,7)
%
%    plot(x1,unwrap(angle(fft(y))))
%    title(['Phase of fourier transform of aperture'])
%
%    subplot(2,5,3)
%
%    plot(x,abs(h3))
%    title(['PLot of Fresnel propagator'])
%
%    subplot(2,5,8)
%
%    plot(x,unwrap(angle(h3)))
%    title('Phase of Fresnel propagator')
%
%    subplot(2,5,4)
%
%    plot(x,fftshift(abs(fft(h3))))
%    title('Fourier transform of Fresnel propagator ignore!!!!')
%
%    subplot(2,5,9)
%
%    plot(x,angle(fft(h3)))
%   title('Phase of Fresnel propagator ignore¬!!!!')
%
%
%    subplot(2,5,5)
%
%
%    title('Convolution of aperture field and propagator')
%    plot(endx,ifftshift(abs(ifft((fft(y)).*(h3)))))
%
%    subplot(2,5,10)
%
%    title('Phase of Convolution')
%    plot(endx,angle(ifftshift(ifft((fft(y)).*(h3)))))



%%
```

```
figure(6)          % Fourier transform of product of aperture and propagator

title('Method 2')

subplot(2,4,1)

plot(x,y)
title(['Rectangular slit of width',num2str(width),'mm'])

subplot(2,4,5)

plot(x,angle(y))
title('Phase of rect function')

subplot(2,4,2)

plot(x,abs(h1))
title('Fresnel propagator')

subplot(2,4,6)

plot(x,unwrap(angle(h1)))
title('Phase of Fresnel propagator')

subplot(2,4,3)

plot(abs(y.*h1))
title('Aperture x propagator')

subplot(2,4,7)

plot(unwrap(angle(y.*h1)))
title('Phase of aperture x propagator')

subplot(2,4,4)

plot(abs(fftshift(fft(y.*h1))))
title('Fourier transform of aperture x propagator')

subplot(2,4,8)

plot(unwrap(angle(fftshift(fft(y.*h1)))))
title('Phase of fourier transform of aperture x propagator')

figure(7)              % Rescaled intensity plot


plot(endx,abs(fres2).^2)
 ylabel('Intensity')
xlabel('x position (m)')

figure(8)

angleendx=atan(endx/z);

plot(angleendx,nonzeros(abs(fres2).^2))
ylabel('Intensity')
xlabel(['Angular distribution at distance ',num2str(z),'m (rad)'])



%    subplot(2,5,9)
%
%    plot(angle(h2))
%
%    subplot(2,5,5)
%
```

```
%     plot((abs((fft(y).*h1))))
%
%     subplot(2,5,10)
%
%     plot(angle(fftshift(fft(y).*h1)))



%     figure(5)                % Amplitude spectrum
%
%     subplot(1,2,1)
%     plot(endx,absfres2)
%     title(['Cross section of Fresnel diffraction pattern produced by rectangular aperture of width',num2str(width),'mm
(rescaled) method 2'])
%     ylabel('Relative amplitude')
%     xlabel('x position (m)')
%     disp(['Fresnel number = ',num2str(F)])
%
%     subplot(1,2,2)
%     plot(x,absfres1)
%     title(['Cross section of Fresnel diffraction pattern produced by rectangular aperture of width',num2str(width),'mm
(rescaled) method 1'])
%     ylabel('Relative amplitude')
%     xlabel('x position (m)')
%
%     figure(6)                % Intensity spectrum
%
%     subplot(1,2,1)
%     plot(endx,(absfres2).^2)
%
%     title(['Cross section of Fresnel diffraction pattern produced by rectangular aperture of width',num2str(width),'mm
(rescaled) method 2'])
%     ylabel('Relative intensity')
%     xlabel('x position (m)')
%
%
%     subplot(1,2,2)
%     plot(endx,(absfres1).^2)
%
%     title(['Cross section of Fresnel diffraction pattern produced by rectangular aperture of width',num2str(width),'mm
(rescaled) method 2'])
%     ylabel('Relative intensity')
%     xlabel('x position (m)')
%
    t=toc

end
```

2D Fresnel diffraction

```
clear all
close all
% 2D Fresnel Diffraction

disp('THIS PROGRAM CALCULATES FRESNEL DIFFRACTION PATTERNS- VALID FOR ALL DISTANCE SCALES.
WHERE FRESNEL NUMBER << 1 CAN USE FRAUNHOFFER APPROX')

% Define diffraction properties

wavelength=input('What is the wavelength of the light used to illuminate the diffracting aperture (nm)?          ');
shape=input('What is the shape of the aperture? Choose 1 for rectangular, 2 for circular          ');
z=input('What is the distance from aperture to screen (m)?          ');
res=input('Select real space resolution factor 1-5 (low-high)          ');
wl=10^-9;
```

```
wdth=10^-3;
D=input('How many apertures tall is the array?        ');
N=input('How many apertures wide is the array?        ');

if shape==1
   width=input('What is the aperture width (mm)?          ');
   w=width*wdth;
   height=input('What is the aperture height (mm)?            ');
   h=height*wdth;
   F=((width*wdth)^2)/((wavelength*wl)*z);
else if shape==2
      rad=input('What is the aperture radius (mm)?            ');
      R=rad*wdth;
      F=((rad*wdth)^2)/((wavelength*wl)*z)
   end
end




if shape==1

% Calculation of Fresnel number (rectangular)




disp(['Fresnel Number = ',num2str(F)])

% Determination of Fresnel or Fraunhoffer diffraction



%%              %  Fresnel Case- single rectangular

tic




   if  N==1 && D==1
   n=2^11;
   I=1:n;
      x=(I-(n/2))*wdth/(res*10);
      y=((n/2)-I)*wdth/(res*10);
      M=zeros(n);
      [X,Y]=meshgrid(x,y);
      A=abs(X)<=(width*wdth)/2;
      B=abs(Y)<=(height*wdth)/2;
      M(A&B)=1;
      scaling=((wavelength*wl)*z)/((x(2)-x(1))*(length(x)-1));
      xx=([-((length(x)-1)/2):((length(x)-1)/2)]*(wavelength*wl))/([(x(2)-x(1))]*(length(x)-1));
      yy=([-((length(y)-1)/2):((length(y)-1)/2)]*wavelength*wl)/([(y(2)-y(1))]*(length(y)-1));


      figure(1)      % Image of diffracting aperture
      imagesc(x,y,M)
      title(['Single slit of width ',num2str(width),'mm and height ',num2str(height),'mm'])

      k=(2*pi)/(wavelength*wl);

   [XX,YY]=meshgrid(xx,yy);

   h1=(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));
%
h1=((exp(i*k*z))/(i*(wavelength*wl)*z))*(exp(((i*k)*(((wavelength*wl)^2)*z/2))*((XX/wavelength*wl*z).^2)+((YY/wavelength
*wl*z).^2)));
```

```
   h2=(exp(((i*k)/(2*z))*((X.^2)+(Y.^2))));


   fh1=(fft2(h1));

   fM=fftshift(fft2(M));

   fres=((ifft2(fftshift((fM).*(h1)))));

   fres2=fftshift(fft2(M.*h2));

   endx=[-((length(x)-1)/2):((length(x)-1)/2)]*((wavelength*wl)*z/((length(x)-1)*(x(2)-x(1))));
   endy=[-((length(y)-1)/2):((length(y)-1)/2)]*((wavelength*wl)*z/((length(x)-1)*(x(2)-x(1))));


      [XXX,YYY]=meshgrid(endx,endy);


      mult=((exp(i*k*z))/(i*(wavelength*wl)*z))*(exp(((i*k)/(2*z))*(XXX.^2 + YYY.^2)));
      yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((fres2.^2)*(endx(2)-endx(1))*(endy(2)-endy(1)))));
      fres2r=yscal*fftshift(fft2(M.*h2));

   figure(2)          % Image of diffraction pattern

%        subplot(1,2,1)

      imagesc(endx,endy,abs(mult.*fres2r))
      axis image
      colormap(hot)
      title(['Amplitude distribution in real space at distance ',num2str(z),'m']);  xlabel('x position (m)');ylabel('y position (m)')

%        subplot(1,2,2)
%
%        imagesc(endx,endy,abs(fres))
%        axis image
%        colormap(hot)
%        title('Amplitude distribution in real space (convolution)');
%        xlabel('x position (m)');ylabel('y position (m)')


      figure(3)          % Intensity spectrum
%
%        subplot(1,2,1)

      mesh(endx,endy,(abs(mult.*fres2r)).^2);shading interp;drawnow;
      title(['Intensity distribution in real space at distance ',num2str(z),'m']);  xlabel('x position (m)');ylabel('y position (m)')

%        subplot(1,2,2)
%
%        mesh(endx,endy,(abs(fres)).^2);shading interp;drawnow;
%        title('Intensity distribution in real space (convolution)');
%        xlabel('x position (m)');ylabel('y position (m)')


t=toc
   end
%%

   if N~=1 || D~=1        % Multiple rectangular apertures


    spac=input('What is the spacing of the apertures (mm)?        ');

    tic

    s=spac*wdth;
    n=2^11;
   I=1:n;
      x=(I-(n/2))*wdth/(res*10);
      y=((n/2)-I)*wdth/(res*10);
      scaling=((wavelength*wl)*z)/((x(2)-x(1))*(length(x)-1));
```

```
xx=([-((length(x)-1)/2):((length(x)-1)/2)]*(wavelength*wl))/([(x(2)-x(1))]*(length(x)-1));
yy=([-((length(y)-1)/2):((length(y)-1)/2)]*wavelength*wl)/([(y(2)-y(1))]*(length(y)-1));
M=zeros(n);
[X,Y]=meshgrid(x,y);
hight=height*res*10;
widt=width*10*res;
space=spac*res*10;

if mod(N,2)==0
    g=2;          % Even number of slits
else
      g=1;        % Odd number of slits
end
if mod(D,2)==0
   u=2;           % Even number of slits tall
else
   u=1;           % Odd number of slits tall
end

if u==1 && g==1       % Odd tall, Odd wide

    J=N-1;
O=D-1;
for G= -(J/2):(J/2);
 for H= -(O/2):(O/2);
 M(((n/2)+((-hight/2)+H*hight)+(H*space)):((n/2)+((hight/2)+H*hight)+(H*space)),((n/2)+((-
widt/2)+G*widt)+(G*space)):((n/2)+((widt/2)+G*widt)+(G*space)))=1;
 end
   end
   end

if g==2 && u==1           % odd tall, even wide

   O=D-1;
   for G= -(N/2):((N/2)-1);
for H= -(O/2):(O/2);
   M(((n/2)+((-hight/2)+H*hight)+(H*space)):((n/2)+((hight/2)+H*hight)+(H*space)),((n/2)+((G-0.5)*space)+((G-
1)*widt)+(widt+space)):(((n/2)+((G-0.5)*space)+(G*widt)+(widt+space))))=1;
end
   end
end

if g==1 && u==2           % Odd slits wide, even tall
J=N-1;
for G= -(J/2):(J/2);
 for H= -(D/2):((D/2)-1);
    M((((n/2)+((H-0.5)*space)+((H-1)*hight)+(hight+space))):((n/2)+((H-
0.5)*space)+(H*hight)+(hight+space)),((n/2)+((-widt/2)+G*widt)+(G*space)):((n/2)+((widt/2)+G*widt)+(G*space)))=1;
 end
end
end


if g==2   && u==2       % For even number of slits wide and tall

  for G= -(N/2):((N/2)-1);
    for H= -(D/2):((D/2)-1);
      M((((n/2)+((H-0.5)*space)+((H-1)*hight)+(hight+space))):((n/2)+((H-
0.5)*space)+(H*hight)+(hight+space)),((n/2)+((G-0.5)*space)+((G-1)*widt)+(widt+space)):(((n/2)+((G-
0.5)*space)+(G*widt)+(widt+space))))=1;
    end
  end
end


figure(1)      % Image of diffracting aperture
imagesc(x,y,M)
title(['Single slit of width ',num2str(width),'mm and height ',num2str(height),'mm'])

k=(2*pi)/(wavelength*wl);
```

```
   [XX,YY]=meshgrid(xx,yy);

   h1=(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));
%
h1=((exp(i*k*z))/(i*(wavelength*wl)*z))*(exp(((i*k)*(((wavelength*wl)^2)*z/2))*((XX/wavelength*wl*z).^2)+((YY/wavelength
*wl*z).^2)));

   h2=(exp(((i*k)/(2*z))*((X.^2)+(Y.^2))));


   fh1=(fft2(h1));

   fM=fftshift(fft2(M));

   fres=((ifft2(fftshift((fM).*(h1)))));

   fres2=fftshift(fft2(M.*h2));

   endx=[-((length(x)-1)/2):((length(x)-1)/2)]*((wavelength*wl)*z/((length(x)-1)*(x(2)-x(1))));
   endy=[-((length(y)-1)/2):((length(y)-1)/2)]*((wavelength*wl)*z/((length(x)-1)*(x(2)-x(1))));


      [XXX,YYY]=meshgrid(endx,endy);


      mult=((exp(i*k*z))/(i*(wavelength*wl)*z))*(exp(((i*k)/(2*z))*(XXX.^2 + YYY.^2)));
      yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((fres2.^2)*(endx(2)-endx(1))*(endy(2)-endy(1)))));
      fres2r=yscal*fftshift(fft2(M.*h2));


   figure(2)         % Image of diffraction pattern

%       subplot(1,2,1)

      imagesc(endx,endy,abs(mult.*fres2r))
      axis image
      colormap(hot)
      title(['Amplitude distribution in real space at distance ',num2str(z),'m']);  xlabel('x position (m)');ylabel('y position (m)')

%       subplot(1,2,2)
%
%       imagesc(endx,endy,abs(fres))
%       axis image
%       colormap(hot)
%       title('Amplitude distribution in real space (convolution)');
%       xlabel('x position (m)');ylabel('y position (m)')


      figure(3)         % Intensity spectrum

      subplot(1,2,1)

      mesh(endx,endy,(abs(mult.*(fres2r)).^2));shading interp;drawnow;
      title(['Intensity distribution in real space at distance ',num2str(z),'m']);  xlabel('x position (m)');ylabel('y position (m)')

%       subplot(1,2,2)
%
%       mesh(endx,endy,(abs(fres)).^2);shading interp;drawnow;
%       title('Intensity distribution in real space (convolution');  xlabel('x position (m)');ylabel('y position (m)')

t=toc


end
end
```

```
%%


if shape==2
   % Calculation of Fresnel number (circular)

   F=((rad*wdth)^2)/((wavelength*wl)*z)

   disp(['Fresnel Number = ',num2str(F)])

   % Determination of Fresnel or Fraunhoffer diffraction

   if F>=0.0001
      G=1;          % Fresnel Diffraction
   else
      G=2;          % Fraunhoffer diffraction
   end

%%                  %  Fresnel Case

tic

if G==1

   if D==1 && N==1;          % Single circular aperture

   n=2^11;
   I=1:n;
      x=(I-(n/2))*wdth/(res*10);
      y=((n/2)-I)*wdth/(res*10);
      M=zeros(n);
      [X,Y]=meshgrid(x,y);
      A=(((X.^2)+(Y.^2))<=((R)^2));
      M(A)=1;
      scaling=((wavelength*wl)*z)/((x(2)-x(1))*(length(x)-1));;

      xx=([-((length(x)-1)/2):((length(x)-1)/2)])*scaling;

      yy=([-((length(y)-1)/2):((length(y)-1)/2)])*scaling;



      figure(1)      % Image of diffracting aperture
      imagesc(x,y,M)
      axis image
      title(['Single circular aperture of radius ',num2str(rad),'mm'])

      k=(2*pi)/(wavelength*wl);

      [XX,YY]=meshgrid(xx,yy);

      h1=(exp(((i*pi)/(wavelength*wl*z))*((XX.^2)+(YY.^2))));

      h2=(exp(((i*k)/(2*z))*((X.^2)+(Y.^2))));

      fh2=((fft2(h2)));

      fM=fftshift(fft2(M));


      fres=((ifft2(fftshift((fM).*(h1)))));

      fres2=fftshift(fft2(M.*h2));

      endx=[-((length(x)-1)/2):((length(x)-1)/2)]*((wavelength*wl)*z/((length(x)-1)*(x(2)-x(1))));
      endy=[-((length(y)-1)/2):((length(y)-1)/2)]*((wavelength*wl)*z/((length(x)-1)*(x(2)-x(1))));

      [XXX,YYY]=meshgrid(endx,endy);
```

```
    mult=((exp(i*k*z))/(i*(wavelength*wl)*z))*(exp(((i*k)/(2*z))*(XXX.^2 + YYY.^2)));
    yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1))))/(sum(sum((fres2.^2)*(endx(2)-endx(1))*(endy(2)-endy(1)))));
    fres2r=yscal*fftshift(fft2(M.*h2));



    figure(2)          % Image of diffraction pattern

%       subplot(1,2,1)

    imagesc(endx,endy,abs(mult.*(fres2r)))
    axis image
    colormap(hot)
    title(['Amplitude distribution in real space at distance ',num2str(z),'m']);  xlabel('x position (m)');ylabel('y position (m)')

%       subplot(1,2,2)
%
%       imagesc(endx,endy,abs(fres))
%       axis image
%       colormap(hot)
%       title('Amplitude distribution in real space (convolution)'); xlabel('x position (m)');ylabel('y position (m)')


    figure(3)          % Intensity spectrum

%       subplot(1,2,1)

    mesh(endx,endy,(abs(fres2r.*mult)).^2);shading interp;drawnow;
    title(['Intensity distribution in real space at distance ',num2str(z),'m']);  xlabel('x position (m)');ylabel('y position (m)')
%
%       subplot(1,2,2)
%
%       mesh(endx,endy,(abs(fres)).^2);shading interp;drawnow;
%       title('Intensity distribution in real space (convolution');  xlabel('x position (m)');ylabel('y position (m)')

    figure(4)

    imagesc(endx,endy,(abs(fres2r.*mult)).^2)
    axis image
    colormap(hot)
    title(['Intensity distribution in real space at distance ',num2str(z),'m']);  xlabel('x position (m)');ylabel('y position (m)')

    t=toc

  end



%%
 if N~=1 || D~=1       % Multiple circular apertures


   spac=input('What is the spacing of the apertures (mm)?        ');

   tic

   s=spac*wdth;
   n=2^11;
  I=1:n;
    x=(I-(n/2))*wdth/(res*10);
    y=((n/2)-I)*wdth/(res*10);
    scaling=((wavelength*wl)*z)/((x(2)-x(1))*(length(x)-1));
    xx=([-((length(x)-1)/2):((length(x)-1)/2)]*(wavelength*wl))/([(x(2)-x(1))]*(length(x)-1));
    yy=([-((length(y)-1)/2):((length(y)-1)/2)]*wavelength*wl)/([(y(2)-y(1))]*(length(y)-1));
    M=zeros(n);
    [X,Y]=meshgrid(x,y);


    if mod(N,2)==0
```

```
    g=2;          % Even number of slits wide
else
      g=1;        % Odd number of slits wide
end
if mod(D,2)==0
   u=2;           % Even number of slits tall
else
   u=1;           % Odd number of slits tall
end

 if g==1 && u==1              % Odd number of circular apertures wide and tall
   J=N-1;
   j=D-1;
   for G= -(J/2):(J/2);
      for H= -(j/2):(j/2);
      A=((((X-(G*((2*R)+s))).^2)+((Y-(H*((2*R)+s))).^2))<=(R^2));
      M(A)=1;
      end
   end
 end

 if g==2 && u==2              % Even number of circular apertures wide and tall
     for G= -(N/2):((N/2)-1);
      for H= -(D/2):((D/2)-1);
      A=((((X-[(s/2)+R+[(G-1)*(s+(2*R))]+(s+2*R)]).^2)+((Y-[(s/2)+R+[(H-1)*(s+(2*R))]+(s+2*R)]).^2))<=(R^2));
      M(A)=1;
      end
     end
 end

 if g==1 && u==2              % Odd wide, even tall
    J=N-1;
   for G= -(J/2):(J/2);
      for H= -(D/2):((D/2)-1);
      A=((((X-(G*((2*R)+s))).^2)+((Y-[(s/2)+R+[(H-1)*(s+(2*R))]+(s+2*R)]).^2))<=(R^2));
      M(A)=1;
      end
   end
 end

 if g==2 && u==1              % Even wide, odd tall
    j=D-1;
   for G= -(N/2):((N/2)-1);
      for H= -(j/2):(j/2);
      A=((((X-[(s/2)+R+[(G-1)*(s+(2*R))]+(s+2*R)]).^2)+((Y-(H*((2*R)+s))).^2))<=(R^2));
      M(A)=1;
      end
   end
 end

 figure(1)      % Image of diffracting aperture
imagesc(x,y,M)
title(['Single circular aperture of radius ',num2str(rad),'mm'])

k=(2*pi)/(wavelength*wl);

[XX,YY]=meshgrid(xx,yy);

%     h1=((exp(i*k*z))/(i*(wavelength*wl)*z))*(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));

h1=(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));

h2=(exp(((i*k)/(2*z))*((X.^2)+(Y.^2))));

fh1=(fft2(h1));

fM=fftshift(fft2(M));

fres=((ifft2(fftshift((fM).*(h1)))));

fres2=fftshift(fft2(M.*h2));
```

```
    endx=[-((length(x)-1)/2):((length(x)-1)/2)]*((wavelength*wl)*z/((length(x)-1)*(x(2)-x(1))));
    endy=[-((length(y)-1)/2):((length(y)-1)/2)]*((wavelength*wl)*z/((length(x)-1)*(x(2)-x(1))));


    [XXX,YYY]=meshgrid(endx,endy);


    mult=((exp(i*k*z))/(i*(wavelength*wl)*z))*(exp(((i*k)/(2*z))*(XXX.^2 + YYY.^2)));

    yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((fres2.^2)*(endx(2)-endx(1))*(endy(2)-endy(1)))));
    fres2r=yscal*fftshift(fft2(M.*h2));


    figure(2)       % Image of diffraction pattern

%       subplot(1,2,1)

    imagesc(endx,endy,abs(mult.*(fres2r)))
    axis image
    colormap(hot)
    title(['Amplitude distribution in real space at distance ',num2str(z),'m']);  xlabel('x position (m)');ylabel('y position (m)')
%
%       subplot(1,2,2)
%
%       imagesc(endx,endy,abs(fres))
%       axis image
%       colormap(hot)
%       title('Amplitude distribution in real space (convolution)'); xlabel('x position (m)');ylabel('y position (m)')


    figure(3)       % Intensity spectrum
%
%       subplot(1,2,1)

    mesh(endx,endy,(abs(mult.*fres2r)).^2);shading interp;drawnow;
    title(['Intensity distribution in real space at distance ',num2str(z),'m']);  xlabel('x position (m)');ylabel('y position (m)')

%       subplot(1,2,2)
%
%       mesh(endx,endy,(abs(fres)).^2);shading interp;drawnow;
%       title('Intensity distribution in real space (convolution)');
%       xlabel('x position (m)');ylabel('y position (m)')

    t=toc


 end
end
end



% Convolution does not work! Second subplots are for fresnel diffraction
% integral formulated as convolution
```

2D Fraunhoffer diffraction

```
%  Program to define diffracting structures and display fraunhoffer diffraction
%  patterns. Program can handle rectangular arrays of rectangular and
%  circular apertures.
%  Co-ordinates have been re-scaled, to give angular spectrum of
%  diffracting structures in radians
%  VALID FOR FAR FIELD ONLY

clear all
close all


disp(' THIS PROGRAM IS VALID FOR FRAUNHOFFER REGIME ONLY- FAR FIELD')
```

```
tic

l=input('What is the wavelength of the light used to illuminate the screen (nm)?                ');
z=input('What is the distance to observation (m)?              ');
W=l*(10^(-9));
wdth=1*10^-3;

p=input('Are the apertures circular (1) or rectangular (2)?         ')

%%

if p==1             % Circular apertures


   N=input('How many apertures wide is the array?      ')
   D=input('How many apertures tall is the array?      ')


   if N==1 && D==1        % Single circular aperture


      Rad=input('What is the radius of the aperture? (mm)          ');

      R=Rad*wdth;
      zz=(R^2)/(W*0.1);
      disp(['The fraunhoffer approximation will be valid only for an observation distance of more than ',num2str(zz),'m for
the parameters given'])
      res=input('Select real space resolution (1-5 (1=1mm resolution, 5=1/5 mm resolution)) WARNING: HIGH
RESOLUTION IN REAL SPACE = LOW RESOLUTION IN FOURIER SPACE        ')
      wdth=10^-3;
      n=2^11;
      M=zeros(n);
      I=1:n;
      x=(I-(n/2))*wdth/(10*res);
      y=((n/2)-I)*wdth/(10*res);
      [X,Y]=meshgrid(x,y);
      A=(((X.^2)+(Y.^2))<=((R)^2));
      M(A)=1;
      xx=(W*([-((length(x)-1)/2):((length(x)-1)/2)]))/([(x(2)-x(1))]*(length(x)-1));
      yy=(W*([-((length(y)-1)/2):((length(y)-1)/2)]))/([(y(2)-y(1))]*(length(y)-1));
      yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((abs(fft2(M)).^2)*(xx(2)-xx(1))*(yy(2)-yy(1)))));
       k=(2*pi)/(W);

       [XX,YY]=meshgrid(xx,yy);

      h2=((exp(i*k*z))/(i*(W)*z))*(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));


      figure(1)      % Image of diffracting aperture
      imagesc(x,y,M)
      title(['Single circular aperture of radius ',num2str(Rad),'mm'])
      xlabel('x position (m)')
      ylabel('y position (m)')


      figure(2)          % Image of fraunhoffer diffraction pattern (unscaled)
      imagesc(x,y,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
         axis image
         colormap(hot)
      title(['Fraunhoffer diffraction pattern produced by ',num2str(l),'nm light through a single circular aperture of radius
',num2str(Rad),'mm']);


      figure(3)          % Image of fraunhoffer diffraction pattern (rescaled)
      imagesc(xx,yy,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
         axis image
         colormap(hot)
      title(['Fraunhoffer diffraction pattern produced by ',num2str(l),'nm light through a single circular aperture of radius
',num2str(Rad),'mm']);xlabel('Angular distribution in x direction (radians)');ylabel('Angular distribution in y direction
(radians)');
```

```
    figure(4)          % Image of hidden fraunhoffer pattern features
    imagesc(xx,yy,abs(log2(h2.*fftshift(fft2(M))*sqrt(yscal))))
    axis image
    colormap(hot)
    title('{\bf Enhanced Diffraction Pattern}')
    xlabel('Angular distribution in x direction (radians)')
    ylabel('Angular distribution in y direction (radians)')


    figure(5)                % Intensity spectrum
    mesh(xx,yy,((abs(h2.*fftshift(fft2(M))*sqrt(yscal))^2)));shading interp;drawnow;
    title('Intensity spectrum'); xlabel('Spatial x frequency (rad)');ylabel('SPatial y frequency (rad)')

    t=toc

  end

  if N~=1 || D~=1                                          % Multiple circular apertures
    Rad=input('What is the radius of each aperture?          ');
    R=Rad*wdth;
    zz=(R^2)/(W*0.1);
    disp(['The fraunhoffer approximation will be valid only for an observation distance of more than ',num2str(zz),'m for
the parameters given'])
    spac=input('What is the spacing of the apertures?         ');
    s=spac*wdth;
    res=input('Select real space resolution (1-5 (1=1mm resolution, 5=1/5 mm resolution)) WARNING: HIGH
RESOLUTION IN REAL SPACE = LOW RESOLUTION IN FOURIER SPACE       ')
    n=2^11;
    M=zeros(n);
     I=1:n;
    x=(I-(n/2))*wdth/(10*res);
    y=((n/2)-I)*wdth/(10*res);
    xx=([-((length(x)-1)/2):((length(x)-1)/2)]*(W))/([(x(2)-x(1))]*(length(x)-1));
    yy=([-((length(y)-1)/2):((length(y)-1)/2)]*W)/([(y(2)-y(1))]*(length(y)-1));
    [X,Y]=meshgrid(x,y);



    if mod(N,2)==0
        g=2;           % Even number of slits wide
    else
           g=1;        % Odd number of slits wide
    end
    if mod(D,2)==0
      u=2;             % Even number of slits tall
    else
      u=1;             % Odd number of slits tall
    end


%%
    if g==1 && u==1                % Odd number of circular apertures wide and tall
      J=N-1;
      j=D-1;
      for G= -(J/2):(J/2);
        for H= -(j/2):(j/2);
        A=((((X-(G*((2*R)+s))).^2)+((Y-(H*((2*R)+s))).^2))<=(R^2));
        M(A)=1;
        end
      end

      yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((abs(fft2(M)).^2)*(xx(2)-xx(1))*(yy(2)-yy(1)))));
    k=(2*pi)/(W);
    [XX,YY]=meshgrid(xx,yy);

    h2=((exp(i*k*z))/(i*(W)*z))*(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));

      figure(1)            % Image of diffracting screen
      imagesc(x,y,M)
```

```
        title(['Diffracting screen with ',num2str(N),' by ',num2str(D),' circular apertures of radius ',num2str(Rad),'mm and
spacing ',num2str(s)])
        xlabel('x position (m)')
        ylabel('y position (m)')

        figure(3)           % Image of fraunhoffer diffraction pattern (rescaled)
        imagesc(xx,yy,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        xlabel('Angular distribution in x direction (radians)');
        ylabel('Angular distribution in y direction (radians)');
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given  by ',num2str(l),'nm light through ',num2str(N),' by ',num2str(D), '
circular apertures of radius ',num2str(R),' and spacing ',num2str(s)])


        figure(2)       % Image of fraunhoffer diffraction pattern (unscaled)
        imagesc(x,y,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern produced by ',num2str(l),'nm light through a single circular aperture of radius
',num2str(R),'mm']);xlabel('Angular distribution in x direction');ylabel('Angular distribution in y direction');xlabel('Angular
distribution in x direction');ylabel('Angular distribution in y direction');


        figure(4)           % Image of hidden fraunhoffer pattern features
        imagesc(xx,yy,abs(log2(h2.*fftshift(fft2(M)))*sqrt(yscal)))
        axis image
        xlabel('Spatial frequency in x direction (radians)');
        ylabel('Angular distribution in y direction (radians)');
        colormap(hot)
        title('{\bf Enhanced Diffraction Pattern}')


        figure(5)               % Intensity spectrum
        mesh(xx,yy,((abs(h2.*fftshift(fft2(M))*sqrt(yscal))^2)));shading interp;drawnow;
        title('Intensity spectrum'); xlabel('Spatial x frequency (rad)');ylabel('SPatial y frequency (rad)')

        t=toc
    end

%%
    if g==2   && u==2           % Even number of circular apertures tall and wide
        for G= -(N/2):((N/2)-1);
            for H= -(D/2):((D/2)-1);
            A=((((X-[(s/2)+R+[(G-1)*(s+(2*R))]+(s+2*R)]).^2)+((Y-[(s/2)+R+[(H-1)*(s+(2*R))]+(s+2*R)]).^2))<=(R^2));
            M(A)=1;
            end
        end

        yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((abs(fft2(M)).^2)*(xx(2)-xx(1))*(yy(2)-yy(1)))));
    k=(2*pi)/(W);
    [XX,YY]=meshgrid(xx,yy);

    h2=((exp(i*k*z))/(i*(W)*z))*(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));

        figure(1)           % Image of diffracting screen
        imagesc(x,y,M)
        title(['Diffracting screen with ',num2str(N),' by ',num2str(D),' circular apertures of radius ',num2str(N),'m and
spacing ',num2str(s),'m'])
        xlabel('x position (m)')
        ylabel('y position (m)')

        figure(3)           % Image of fraunhoffer diffraction pattern (rescaled)
        imagesc(xx,yy,abs(fftshift(fft2(M)))*sqrt(yscal))
        xlabel('Angular distribution in x direction (radians)');
        ylabel('Angular distribution in y direction (radians)');
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given by ',num2str(l),'nm light through ',num2str(N),' by ',num2str(D), ' circular
apertures of radius ',num2str(R),' and spacing ',num2str(s)])
```

```
        figure(2)          % Image of fraunhoffer diffraction pattern (unscaled)
        imagesc(x,y,abs(fftshift(fft2(M)))*sqrt(yscal))
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given  by ',num2str(l),'nm light through ',num2str(N),' by ',num2str(D), '
circular apertures of radius ',num2str(R),' and spacing ',num2str(s)]);xlabel('Angular distribution in x
direction');ylabel('Angular distribution in y direction');


      figure(4)            % Image of hidden fraunhoffer pattern features
      imagesc(xx,yy,abs(log2(fftshift(fft2(M))*sqrt(yscal))))
      xlabel('Angular distribution in x direction (radians)');
      ylabel('Angular distribution in y direction (radians)');
      axis image
      colormap(hot)
      title('{\bf Enhanced Diffraction Pattern}')


      figure(5)                  % Intensity spectrum
      mesh(xx,yy,(abs((fftshift(fft2(M))*sqrt(yscal))^2)));shading interp;drawnow;
      title('Intensity spectrum'); xlabel('Spatial x frequency (rad)');ylabel('SPatial y frequency (rad)')

      t=toc
      end

%%


 if g==2   && u==1                % Even number of circular apertures wide, odd number tall
    j=D-1;
        for G= -(N/2):((N/2)-1);
           for H= -(j/2):(j/2);
           A=((((X-[(s/2)+R+[(G-1)*(s+(2*R))]+(s+2*R)]).^2)+((Y-(H*((2*R)+s))).^2))<=(R^2));
           M(A)=1;
           end
        end

        yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((abs(fft2(M)).^2)*(xx(2)-xx(1))*(yy(2)-yy(1)))));
      k=(2*pi)/(W);
      [XX,YY]=meshgrid(xx,yy);

      h2=((exp(i*k*z))/(i*(W)*z))*(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));

        figure(1)            % Image of diffracting screen
        imagesc(x,y,M)
        title(['Diffracting screen with ',num2str(N),' by ',num2str(D),' circular apertures of radius ',num2str(R),'m and
spacing ',num2str(s),'m'])
        xlabel('x position (m)')
        ylabel('y position (m)')

        figure(2)          % Image of fraunhoffer diffraction pattern (unscaled)
        imagesc(x,y,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given  by ',num2str(l),'nm light through ',num2str(N),' by ',num2str(D), '
circular apertures of radius ',num2str(R),'m and spacing ',num2str(s),'m']);xlabel('Angular distribution in x
direction');ylabel('Angular distribution in y direction');


        figure(3)            % Image of fraunhoffer diffraction pattern (rescaled)
        imagesc(xx,yy,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        xlabel('Angular distribution in x direction (radians)')
        ylabel('Angular distribution in y direction (radians)')
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given by ',num2str(l),'nm light through ',num2str(N),' by ',num2str(D), ' circular
apertures of radius ',num2str(R),'m and spacing ',num2str(s),'m'])


        figure(4)          % Image of hidden fraunhoffer pattern features
```

```
        imagesc(xx,yy,abs(h2.*log2(fftshift(fft2(M))*sqrt(yscal))))
        xlabel('Angular distribution in x direction (radians)')
        ylabel('Angular distribution in y direction (radians)')
        axis image
        colormap(hot)
        title('{\bf Enhanced Diffraction Pattern}')


        figure(5)                   % Intensity spectrum
        mesh(xx,yy,(abs((h2.*fftshift(fft2(M))*sqrt(yscal))^2)));shading interp;drawnow;
        title('Intensity spectrum'); xlabel('Spatial x frequency (rad)');ylabel('SPatial y frequency (rad)')

        t=toc
 end


%%
 if g==1   && u==2                % Odd number of circular apertures wide, even number tall
    J=N-1;
        for G= -(J/2):(J/2);
           for H= -(D/2):((D/2)-1);
           A=((((X-(G*((2*R)+s))).^2)+((Y-[(s/2)+R+[(H-1)*(s+(2*R))]+(s+2*R)]).^2))<=(R^2));
           M(A)=1;
           end
        end

yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((abs(fft2(M)).^2)*(xx(2)-xx(1))*(yy(2)-yy(1)))));
      k=(2*pi)/(W);
      [XX,YY]=meshgrid(xx,yy);

      h2=((exp(i*k*z))/(i*(W)*z))*(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));

        figure(1)             % Image of diffracting screen
        imagesc(x,y,M)
        title(['Diffracting screen with ',num2str(N),' by ',num2str(D),' circular apertures of radius ',num2str(N),'m and
spacing ',num2str(s),'m'])
        xlabel('x position (m)')
        ylabel('y position (m)')

        figure(2)          % Image of fraunhoffer diffraction pattern (unscaled)
        imagesc(x,y,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given  by ',num2str(l),'nm light through ',num2str(N),' by ',num2str(D), '
circular apertures of radius ',num2str(R),'m and spacing ',num2str(s),'m']);xlabel('Angular distribution in x
direction');ylabel('Angular distribution in y direction');


        figure(3)             % Image of fraunhoffer diffraction pattern
        imagesc(xx,yy,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        xlabel('Angular distribution in x direction (radians)')
        ylabel('Angular distribution in y direction (radians)')
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given  by ',num2str(l),'nm light through ',num2str(N),' by ',num2str(D), '
circular apertures of radius ',num2str(R),'m and spacing ',num2str(s),'m'])


        figure(4)             % Image of hidden fraunhoffer pattern features
      imagesc(xx,yy,abs(log2(h2.*fftshift(fft2(M))*sqrt(yscal))))
      xlabel('Angular distribution in x direction (radians)')
      ylabel('Angular distribution in y direction (radians)')
      axis image
      colormap(hot)
      title('{\bf Enhanced Diffraction Pattern}')


      figure(5)               % Intensity spectrum
      mesh(xx,yy,(abs(h2.*fftshift(fft2(M))*sqrt(yscal))^2));shading interp;drawnow;
      title('Intensity spectrum'); xlabel('Spatial x frequency (rad)');ylabel('SPatial y frequency (rad)')
```

```
        t=toc
        end
%%


    end

%%

end

%%

if p== 2        % Rectangular slits
    N=input('How many apertures wide is the array?     ');
    D=input('How many apertures tall is the array?        ')


    if N==1 && D==1     % Single slit
        width=input('What is the aperture width (mm)?     ');
        w=width*wdth;
        height=input('What is the aperture height (mm)?    ');
        h=height*wdth;
        c=[w h];
        p=sort(c);
        a=p(2);
        zz=(a^2)/(W*0.1);
        disp(['The fraunhoffer approximation will be valid only for an observation distance of more than ',num2str(zz),'m for
the parameters given'])
        res=input('Select real space resolution (1-5 (1=1mm resolution, 5=1/5 mm resolution)) WARNING: HIGH
RESOLUTION IN REAL SPACE = LOW RESOLUTION IN FOURIER SPACE       ')
        n=2^11;
        I=1:n;
        x=(I-(n/2))*wdth/(10*res);
        y=((n/2)-I)*wdth/(10*res);
        M=zeros(n);
        [X,Y]=meshgrid(x,y);
        A=abs(X)<=w/2;
        B=abs(Y)<=h/2;
        M(A&B)=1;
        xx=([-((length(x)-1)/2):((length(x)-1)/2)]*(W))/([(x(2)-x(1))]*(length(x)-1));
        yy=([-((length(y)-1)/2):((length(y)-1)/2)]*W)/([(y(2)-y(1))]*(length(y)-1));
        yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((abs(fft2(M)).^2)*(xx(2)-xx(1))*(yy(2)-yy(1)))));
        k=(2*pi)/(W);

         [XX,YY]=meshgrid(xx,yy);

        h2=((exp(i*k*z))/(i*(W)*z))*(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));


        figure(1)       % Image of diffracting aperture
        imagesc(x,y,M)
        title(['Single slit of width ',num2str(w),'m and height ',num2str(h),'m'])
        xlabel('x position (m)')
        ylabel('y position (m)')

        figure(2)          % Image of fraunhoffer diffraction pattern (unscaled)
        imagesc(x,y,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given by ',num2str(I),'nm light through a single slit of width ',num2str(w),'m and
height ',num2str(h),'m']);xlabel('Angular distribution in x direction');ylabel('Angular distribution in y direction');


        figure(3)          % Image of fraunhoffer diffraction pattern
        FT2=fft2(M);
        imagesc(xx,yy,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        xlabel('Angular distribution in x direction (radians)')
        ylabel('Angular distribution in y direction (radians)')
        axis image
        colormap(hot)
```

```
      title(['Fraunhoffer diffraction pattern given by ',num2str(l),'nm light through a single slit of width ',num2str(w),'m and
height ',num2str(h),'m'])


      figure(4)          % Image of hidden fraunhoffer pattern features
      xlabel('Angular distribution in x direction (radians)')
      ylabel('Angular distribution in y direction (radians)')
      imagesc(xx,yy,abs(log2(h2.*fftshift(fft2(M))*sqrt(yscal))))
      axis image
      colormap(hot)
      title('{\bf Enhanced Diffraction Pattern}')


      figure(5)          % Intensity spectrum
      mesh(xx,yy,(abs(h2.*fftshift(fft2(M))*sqrt(yscal))^2));shading interp;drawnow;
      title('Intensity spectrum'); xlabel('Spatial x frequency (rad)');ylabel('SPatial y frequency (rad)')

      t=toc


   end
%%

   if N~=1  || D~=1            % Multiple slits
      width=input('What is the width of each aperture in mm?     ');
      w=width*wdth;
      height=input('What is the height of each aperture in mm?     ');
      h=height*wdth;
      spac=input('What is the spacing of apertures in mm?     ');
      s=spac*wdth;
c=[w h];
      p=sort(c);
      a=p(2);
      zz=(a^2)/((W)*0.1);
      disp(['The fraunhoffer approximation will be valid only for an observation distance of more than ',num2str(zz),'m for
the parameters given'])
      Res=input('Select real space resolution (1-20 (1=1mm resolution, 20=1/20 mm resolution)) WARNING: HIGH
RESOLUTION IN REAL SPACE = LOW RESOLUTION IN FOURIER SPACE       ');

      if mod(N,2)==0
          g=2;          % Even number of slits
      else
              g=1;      % Odd number of slits
      end
      if mod(D,2)==0
         u=2;            % Even number of slits tall
      else
         u=1;            % Odd number of slits tall
      end

%%
 if g==1  && u==1             % For odd number of slits wide and tall
      J=N-1;
      O=D-1;
      n=2^11;
      I=1:n;
     x=(I-(n/2))*wdth/(10*res);
      y=((n/2)-I)*wdth/(10*res);
      M=zeros(n);
      [X,Y]=meshgrid(x,y);
      xx=([-((length(x)-1)/2):((length(x)-1)/2)]*(W))/([(x(2)-x(1))]*(length(x)-1));
      yy=([-((length(y)-1)/2):((length(y)-1)/2)]*W)/([(y(2)-y(1))]*(length(y)-1));
      hight=height*res;
      widt=width*res;
      space=spac*res;


   for G= -(J/2):(J/2);
   for H= -(O/2):(O/2);
   M(((n/2)+((-hight/2)+H*hight)+(H*space)):((n/2)+((hight/2)+H*hight)+(H*space)),((n/2)+((-
widt/2)+G*widt)+(G*space)):((n/2)+((widt/2)+G*widt)+(G*space)))=1;
```

```
    end
    end

        yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((abs(fft2(M)).^2)*(xx(2)-xx(1))*(yy(2)-yy(1)))));
        k=(2*pi)/(W);

         [XX,YY]=meshgrid(xx,yy);

        h2=((exp(i*k*z))/(i*(W)*z))*(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));

        figure(1)               % Image of diffracting screen
        imagesc(x,y,M)
        title(['Diffracting screen with ',num2str(N),' by ',num2str(D),' rectangular slits of width ',num2str(w),'m, height
',num2str(h),'m and spacing ',num2str(s),'m'])
        xlabel('x position (m)')
        ylabel('y position (m)')

        figure(2)           % Image of fraunhoffer diffraction pattern (unscaled)
        imagesc(x,y,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given by ',num2str(l),'nm light through screen with ',num2str(N),' by
',num2str(D),' rectangular slits of width ',num2str(w),'m, height ',num2str(h),'m and spacing
',num2str(s),'m']);xlabel('Angular distribution in x direction');ylabel('Angular distribution in y direction');


        figure(3)               % Image of fraunhoffer diffraction pattern
        FT2=fft2(M);
        imagesc(xx,yy,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        xlabel('Angular distribution in x direction (radians)')
        ylabel('Angular distribution in y direction (radians)')
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given by ',num2str(l),'nm light through screen with ',num2str(N),' by
',num2str(D),' rectangular slits of width ',num2str(w),'mm, height ',num2str(h),'mm and spacing ',num2str(s),'mm'])


        figure(4)               % Image of hidden fraunhoffer pattern features
        imagesc(xx,yy,abs(log2(h2.*fftshift(fft2(M))*sqrt(yscal))))
        xlabel('Angular distribution in x direction (radians)')
        ylabel('Angular distribution in y direction (radians)')
        axis image
        colormap(hot)
        title('{\bf Enhanced Diffraction Pattern}')


        figure(5)               % Intensity spectrum
        mesh(xx,yy,(abs(h2.*fftshift(fft2(M))*sqrt(yscal))^2));shading interp;drawnow;
        title('Intensity spectrum'); xlabel('Spatial x frequency (rad)');ylabel('SPatial y frequency (rad)')

        t=toc
 end

%%


if g==2 && u==1             % For even number of slits wide, odd tall
        O=D-1;
        n=2^11;
        I=1:n;
       x=(I-(n/2))*wdth/(10*res);
        y=((n/2)-I)*wdth/(10*res);
        M=zeros(n);
        [X,Y]=meshgrid(x,y);
        xx=([-((length(x)-1)/2):((length(x)-1)/2)]*(W))/([(x(2)-x(1))]*(length(x)-1));
        yy=([-((length(y)-1)/2):((length(y)-1)/2)]*W)/([(y(2)-y(1))]*(length(y)-1));
        hight=height*res*10;
        widt=width*res;
        space=spac*res;
```

```
   for G= -(N/2):((N/2)-1);
      for H= -(O/2):(O/2);
         M(((n/2)+((-hight/2)+H*hight)+(H*space)):((n/2)+((hight/2)+H*hight)+(H*space)),((n/2)+((G-0.5)*space)+((G-
1)*widt)+(widt+space)):(((n/2)+((G-0.5)*space)+(G*widt)+(widt+space))))=1;
      end
   end
      yscal1=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))));
      yscal2=(sum(sum((abs(fft2(M)).^2)*(xx(2)-xx(1))*(yy(2)-yy(1)))));
      yscal=abs(yscal1)/abs(yscal2);
      k=(2*pi)/(W);

       [XX,YY]=meshgrid(xx,yy);

      h2=((exp(i*k*z))/(i*(W)*z))*(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));

      figure(1)              % Image of diffracting screen
      imagesc(x,y,M)
      title(['Diffracting screen with ',num2str(N),' by ',num2str(D),' rectangular slits of width ',num2str(w),'m, height
',num2str(h),'m and spacing ',num2str(s),'m'])
      xlabel('x position (m)')
      ylabel('y position (m)')

      figure(2)          % Image of fraunhoffer diffraction pattern (unscaled)
      imagesc(x,y,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
      axis image
      colormap(hot)
      title(['Fraunhoffer diffraction pattern given by ',num2str(l),'nm light through screen with ',num2str(N),' by
',num2str(D),' rectangular slits of width ',num2str(w),'m, height ',num2str(h),'m and spacing
',num2str(s),'m']);xlabel('Angular distribution in x direction');ylabel('Angular distribution in y direction');


      figure(3)              % Image of fraunhoffer diffraction pattern
      FT2=fft2(M);
      imagesc(xx,yy,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
      xlabel('Angular distribution in x direction (radians)')
      ylabel('Angular distribution in y direction (radians)')
      axis image
      colormap(hot)
      title(['Fraunhoffer diffraction pattern given by ',num2str(l),'nm light through screen with ',num2str(N),' by
',num2str(D),' rectangular slits of width ',num2str(w),'m, height ',num2str(h),'m and spacing ',num2str(s),'m'])


      figure(4)              % Image of hidden fraunhoffer pattern features
      xlabel('Angular distribution in x direction (radians)')
      ylabel('Angular distribution in y direction (radians)')
      imagesc(xx,yy,abs(log2(h2.*fftshift(fft2(M))*sqrt(yscal))))
      axis image
      colormap(hot)
      title('{\bf Enhanced Diffraction Pattern}')


      figure(5)
      mesh(xx,yy,(abs(h2.*fftshift(fft2(M))*sqrt(yscal))^2));shading interp;drawnow;
      title('Intensity spectrum'); xlabel('Spatial x frequency (rad)');ylabel('SPatial y frequency (rad)')

      t=toc
 end


%%
if g==1 && u==2            % Odd slits wide, even tall
      J=N-1;
      n=2^11;
      I=1:n;
      x=(I-(n/2))*wdth/(10*res);
      y=((n/2)-I)*wdth/(10*res);
      M=zeros(n);
      [X,Y]=meshgrid(x,y);
      xx=([-((length(x)-1)/2):((length(x)-1)/2)]*(W))/([(x(2)-x(1))]*(length(x)-1));
      yy=([-((length(y)-1)/2):((length(y)-1)/2)]*W)/([(y(2)-y(1))]*(length(y)-1));
      hight=height*res;
```

```
        widt=width*res;
        space=spac*res;



    for G= -(J/2):(J/2);
        for H= -(D/2):((D/2)-1);
            M((((n/2)+((H-0.5)*space)+((H-1)*hight)+(hight+space))):((n/2)+((H-
0.5)*space)+(H*hight)+(hight+space)),((n/2)+((-widt/2)+G*widt)+(G*space)):((n/2)+((widt/2)+G*widt)+(G*space)))=1;
        end
    end

        yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((abs(fft2(M)).^2)*(xx(2)-xx(1))*(yy(2)-yy(1)))));
        k=(2*pi)/(W);

        [XX,YY]=meshgrid(xx,yy);

        h2=((exp(i*k*z))/(i*(W)*z))*(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));

        figure(1)               % Image of diffracting screen
        imagesc(x,y,M)
        title(['Diffracting screen with ',num2str(N),' by ',num2str(D),' rectangular slits of width ',num2str(w),'m, height
',num2str(h),'m and spacing ',num2str(s),'m'])
        xlabel('x position (m)')
        ylabel('y position (m)')

        figure(2)           % Image of fraunhoffer diffraction pattern (unscaled)
        imagesc(x,y,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given by ',num2str(l),'nm light through screen with ',num2str(N),' by
',num2str(D),' rectangular slits of width ',num2str(w),'m, height ',num2str(h),'m and spacing
',num2str(s),'m']);xlabel('Angular distribution in x direction');ylabel('Angular distribution in y direction');


        figure(3)               % Image of fraunhoffer diffraction pattern
        xlabel('Angular distribution in x direction (radians)')
        ylabel('Angular distribution in y direction (radians)')
        FT2=fft2(M);
        imagesc(xx,yy,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given by ',num2str(l),'nm light through screen with ',num2str(N),' by
',num2str(D),' rectangular slits of width ',num2str(w),'m, height ',num2str(h),'m and spacing ',num2str(s),'m'])


        figure(4)               % Image of hidden fraunhoffer pattern features
        xlabel('Angular distribution in x direction (radians)')
        ylabel('Angular distribution in y direction (radians)')
        imagesc(xx,yy,abs(log2(h2.*fftshift(fft2(M))*sqrt(yscal))))
        axis image
        colormap(hot)
        title('{\bf Enhanced Diffraction Pattern}')


        figure(5)               % Intensity Spectrum
        mesh(xx,yy,(abs(h2.*fftshift(fft2(M))*sqrt(yscal))^2));shading interp;drawnow;
        title('Intensity spectrum'); xlabel('Spatial x frequency (rad)');ylabel('SPatial y frequency (rad)')

        t=toc
 end



%%


 if g==2   && u==2          % For even number of slits wide and tall
        n=2^11;
        l=1:n;
        x=(l-(n/2))*wdth/(10*res);
```

```
        y=((n/2)-I)*wdth/(10*res);
        M=zeros(n);
        [X,Y]=meshgrid(x,y);
        xx=([-((length(x)-1)/2):((length(x)-1)/2)]*(W))/([(x(2)-x(1))]*(length(x)-1));
        yy=([-((length(y)-1)/2):((length(y)-1)/2)]*W)/([(y(2)-y(1))]*(length(y)-1));
        hight=height*res*10;
        widt=width*res;
        space=spac*res;


     for G= -(N/2):((N/2)-1);
         for H= -(D/2):((D/2)-1);
             M((((n/2)+((H-0.5)*space)+((H-1)*hight)+(hight+space))):((n/2)+((H-
0.5)*space)+(H*hight)+(hight+space)),((n/2)+((G-0.5)*space)+((G-1)*widt)+(widt+space)):(((n/2)+((G-
0.5)*space)+(G*widt)+(widt+space))))=1;
          end
     end

        yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((abs(fft2(M)).^2)*(xx(2)-xx(1))*(yy(2)-yy(1)))));
        k=(2*pi)/(W);

         [XX,YY]=meshgrid(xx,yy);

        h2=((exp(i*k*z))/(i*(W)*z))*(exp(((i*k)/(2*z))*((XX.^2)+(YY.^2))));

        figure(1)               % Image of diffracting screen
        imagesc(x,y,M)
        title(['Diffracting screen with ',num2str(N),' by ',num2str(D),' rectangular slits of width ',num2str(w),'m, height
',num2str(h),'m and spacing ',num2str(s),'m'])
        xlabel('x position (m)')
        ylabel('y position (m)')

        figure(2)           % Image of fraunhoffer diffraction pattern (unscaled)
        imagesc(x,y,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given by ',num2str(I),'nm light through screen with ',num2str(N),' by
',num2str(D),' rectangular slits of width ',num2str(w),'m, height ',num2str(h),'m and spacing
',num2str(s),'m']);xlabel('Angular distribution in x direction');ylabel('Angular distribution in y direction');


        figure(3)               % Image of fraunhoffer diffraction pattern
        xlabel('Angular distribution in x direction (radians)')
        ylabel('Angular distribution in y direction (radians)')
        FT2=fft2(M);
        imagesc(xx,yy,abs(h2.*fftshift(fft2(M)))*sqrt(yscal))
        axis image
        colormap(hot)
        title(['Fraunhoffer diffraction pattern given by ',num2str(I),'nm light through screen with ',num2str(N),' by
',num2str(D),' rectangular slits of width ',num2str(w),'m, height ',num2str(h),'m and spacing ',num2str(s),'m'])


        figure(4)               % Image of hidden fraunhoffer pattern features
        xlabel('Angular distribution in x direction (radians)')
        ylabel('Angular distribution in y direction (radians)')
        imagesc(xx,yy,abs(log2(h2.*fftshift(fft2(M)))*sqrt(yscal)))
        axis image
        colormap(hot)
        title('{\bf Enhanced Diffraction Pattern}')


        figure(5)               % Intensity spectrum
        mesh(xx,yy,(abs(h2.*fftshift(fft2(M))*sqrt(yscal))^2));shading interp;drawnow;
        title('Intensity spectrum'); xlabel('Spatial x frequency (rad)');ylabel('SPatial y frequency (rad)')

        t=toc
 end


end
end
```

%%
1D step-by-step Fresnel propagation

```
% Program for step-by-step fresnel propagation from slit in 1d

clear all;close all

width=input('What is the width of the slit? (mm)'        );
wl=input('What is the wavelength of light (nm)?'          )*1e-9;
wdth=1e-3;

origx=[-20:0.01:20]*wdth;
x=origx;
origy=abs(x)<=(width*wdth)/2;
y=origy;
k=2*pi/wl;

n=input('Select number of propagation steps         ');
dz=input('Select distance of propagation steps (m)      ');

for S=1:n;
    z1=(S-1)*dz;
    z2=z1+dz;
    [x,y,z1]=fresnel1b(x,y,wl,z1,dz);
    y2=y/max(abs(y));


    scaling1=(wl)*z2/((origx(2)-origx(1))*(length(origx)-1));
    endx=[-((length(origx)-1)/2):((length(origx)-1)/2)]*scaling1;
    h1=((exp(((i*k)/(2*z2))*origx.^2)));
    yscal=sum(((origy.*abs(h1)).^2)*(origx(2)-origx(1)))/sum((abs(fft(origy.*h1)).^2)*(endx(2)-endx(1)));
    h2=(1/(sqrt(i*(wl)*z2)))*(exp(((i*k)/(2*z2))*(endx.^2)));
    fres2=fftshift(fft(origy.*h1))*sqrt(yscal).*h2;
    fres2n=abs(fres2)/max(abs(fres2));
    fresang=angle(fres2);

    figure(S)
    subplot(1,3,1)
    plot(x,y2)
    title(['Field distribution after ',num2str(S),' propagation steps of ',num2str(dz),'m']);
    xlabel('x (m)');
    ylabel('Normalised Amplitude');
    subplot(1,3,2)
    plot(x,fres2n)
    title(['Field distribution computed with single step propagation to ',num2str(z2),'m']);
    xlabel('x (m)');
    ylabel('Normalised amplitude');
    subplot(1,3,3)
    error=(fres2n.^2)*(endx(2)-endx(1))-(y2.^2)*(x(2)-x(1));
    plot(endx,error)
    title(['Error between single and ',num2str(S),' step propagation to ',num2str(z2),'m']);
    xlabel('x (m)');
    ylabel('Error');
    sumerror=sum(error);
    percdiff=(abs(((fres2n)./y2)))/max(abs(((fres2n)./y2)));
    percdiffmean(S)=mean(percdiff);
    CHISQ(S)=sum(((y2-fres2n).^2)./fres2n);

    sumerrorArr(S)=sumerror;
    MEANY(S)=mean(y2);
    MEANF(S)=mean(fres2n);




    hold on


end
figure(n+1)
```

```
plot(sumerrorArr);
   title('Graph showing accumulation of error with increasing number of propagation steps');
   xlabel('Propagation step');
   ylabel('Difference in square integral');


   figure(n+2)
   plot(CHISQ)



hold off

for I=1:25
d=2*I;
EVEN(I)=CHISQ(d);
hold on
end
for I=1:25
d=2*I-1;
ODD(I)=CHISQ(d);
hold on
end

hold off

figure(n+3)
plot(EVEN)
title('X^2 vs even propagation step');
xlabel('Propagation step')
ylabel('X^2')

figure(n+4)
plot(ODD)
title('X^2 vs odd propagation step');
xlabel('Propagation step')
ylabel('X^2')
```

PSD calculator for mirror profile

```
disp(['The standard deviation of the height error is ',num2str(standevheight),'(rms)']);

xscale=1/((length(latx)-1)*(latx(2)-latx(1)));
endx=latx*xscale;

figure (5)

plot(latx,heighterror);                    % Test with raw height
title('Graph of height error against lateral position');
xlabel('x (m)');
ylabel('height error (m)');

figure(6)

plot((endx+min(endx)),abs(fftshift(fft(height))))

figure(7)


yscale=(sum((heighterror.^2)*(latx(2)-latx(1))))/(sum((abs(fft(heighterror)).^2).*(endx(2)-endx(1))));
transheighterror=sqrt(yscale).*abs(fft(heighterror));
PSD=((abs(transheighterror).^2))/((length(latx)-1)*(latx(2)-latx(1)));
forlogendx=endx-min(endx);

loglog(forlogendx,transheighterror);


figure(8)

plot(endx,fftshift(PSD))
```

```
figure(9)



loglog(forlogendx,(PSD))

title('PSD vs spatial frequency')
xlabel('Spatial frequency m^-1')
ylabel('Power Spectral density m^-3')

figure(10)

loglog(forlogendx(5:300),transheighterror(5:300))

title('PSD vs spatial frequency')
xlabel('Spatial frequency m^-1')
ylabel('Height error (m)')

figure(11)

loglog(forlogendx(301:600),transheighterror(301:600))

title('PSD vs spatial frequency')
xlabel('Spatial frequency mm^-1')
ylabel('Height error (m)')
```

1D mirror transformation calculator

```
clear all; close all
ang=input('What is the angle of incidence (radians)?              ');
wl=input('What is the wavelength of the incident radiation (nm)?           ')*10e-9;
f=0.001:0.001:1;
xxx=0.001:0.001:1;
[XXX,F]=meshgrid(xxx,f);
y5=-1.764*log10(F)-11.01;
yyy=10^y5;
yo=4*pi*yyy*cos(ang)/wl;
T=exp((i*yo/2)*sin(2*pi*F.*XXX));
M=prod(T);
figure(1);plot(f,M)
figure(2);imagesc(angle(T))
figure(3);plot(xxx,angle(M))
title('Graph showing relative phase delay imparted by mirror vs. lateral position on mirror')
xlabel('Lateral position on mirror (m)')
ylabel('Phase (rad)')
```

Function to define 2D circular aperture

```
% Function to define circular aperture- enter radius r in mm, resolution
% should be entered as 1/10res mm. Field matrix denoted as CIRC

function [CIRC,x,y]=circap(r,res);

wdth=1e-3;
R=r*wdth;
n=2^11;
l=1:n;
x=(l-(n/2))*wdth/(res*10);
y=((n/2)-l)*wdth/(res*10);
CIRC=zeros(n);
[X,Y]=meshgrid(x,y);
B=(((X.^2)+(Y.^2))<=((R)^2));
CIRC(B)=1;
```

Function to define 2D sinusoidal amplitude grating

```
% Function to define a sinusoidal amplitude grating with lines per m l,
% peak-to-peak amplitude transmittance m, with resolution res. Field matrix
% denoted as DIFF
```

```
function [DIFF,x,y]=diffgrat(m,l,res);

wdth=1e-3;
n=2^11;
l=1:n;
x=(l-(n/2))*wdth/(res*10);
y=((n/2)-l)*wdth/(res*10);
M=ones(n);
[X,Y]=meshgrid(x,y);


T=(0.5+(m/2)*cos((2*pi*l)*X));

DIFF=T.*M;
```

Function to define 2D sinusoidal phase grating

```
% Function to define a sinusoidal phase grating with lines per m l,
% peak-to-peak amplitude transmittance m, with resolution res. Field matrix
% is denoted as PHAS

function [PHAS,x,y]=diffgratp(l,m,res);

wdth=1e-3;
n=2^11;
l=1:n;
x=(l-(n/2))*wdth/(res*10);
y=((n/2)-l)*wdth/(res*10);
M=ones(n);
[X,Y]=meshgrid(x,y);

T=exp((i*m/2)*sin(2*pi*l*X));

PHAS=T.*M;
```

Function to define 2D multiplication by quadratic phase exponential

```
% Function for multiplication of 2D field by quadratic phase exponential
% given spatial field distribution and parameter c (e.g. focal length of
% lens). Field matrix G

function [G]=expmult(x,y,wl,c);

k=2*pi/wl;

[X,Y]=meshgrid(x,y);

G=(exp((i*k/2*c)*((X.^2)+(Y.^2))));
```

Function for 1D Fresnel propagation

```
% Function for 1D Fresnel propagation a distance z given spatial field
% distribution, wavelength, propagation distance. For use as operator in
% wave-optics analysis

function [endx,abfres,z2]=fresnel1b(x,y,wl,z1,dz);

if z1~=0
   z2=z1+dz;

else
   z2=dz;
end

k=(2*pi)/(wl);
scaling1=(wl*z2)/((x(2)-x(1))*(length(x)-1));
endx=[-((length(x)-1)/2):((length(x)-1)/2)]*scaling1;

h1=((exp(((i*k)/(2*z2))*(x.^2))));
h2=(1/(sqrt(i*(wl)*dz)))*(exp(((i*k)/(2*dz))*(endx.^2)));
```

```
yscal=sum(((y.*abs(h1)).^2)*(x(2)-x(1)))/sum((abs(fft(y.*h1)).^2)*(endx(2)-endx(1)));


fres2=fftshift(fft(y.*h1)).*h2*sqrt(yscal);

abfres=abs(fres2);
```

Function for 2D Fresnel propagation

```
% Function for 2D Fresnel propagation a distance z given spatial field
% distribution, wavelength, propagation distance. For use as operator in
% wave-optics analysis

function [MM,endx,endy,z2]=fresnel2(M,x,y,wl,z1,dz);

if z1~=0
   z2=z1+dz;
else
   z2=dz;
end

scaling=((wl)*z2)/((x(2)-x(1))*(length(x)-1));
k=(2*pi)/wl;
[X,Y]=meshgrid(x,y);
h2=(exp(((i*k)/(2*z2))*((X.^2)+(Y.^2))));
endx=[-((length(x)-1)/2):((length(x)-1)/2)]*((wl)*z2/((length(x)-1)*scaling));
endy=[-((length(y)-1)/2):((length(y)-1)/2)]*((wl)*z2/((length(x)-1)*scaling));
[XXX,YYY]=meshgrid(endx,endy);
mult=((exp(i*k*dz))/(i*(wl)*dz))*(exp(((i*k)/(2*dz))*(XXX.^2 + YYY.^2)));
fres2=fftshift(fft2(M.*h2));
yscal=(sum(sum((M.^2)*(x(2)-x(1))*(y(2)-y(1)))))/(sum(sum((fres2.^2)*(endx(2)-endx(1))*(endy(2)-endy(1)))));
fres2r=sqrt(yscal)*fftshift(fft2(M.*h2));
MM=abs(mult.*fres2r);

figure(2)

imagesc(endx,endy,MM);
axis image
colormap(hot)
title(['Amplitude distribution in real space at distance ',num2str(z2),'m']);  xlabel('x position (m)');ylabel('y position (m)')
```

Function to define 2D Gaussian beam profile

```
% Function to definine incident Gaussian beam profile in 2D. Field matrix
% denoted as A

function [A,x,y]=gaussbeam(w0,wl,z0,res)

wdth=1e-3;
k=2*pi/wl;
zR=pi*(w0^2)/wl;
wz=w0*sqrt(1+(z0/zR)^2);
Rz=z0*(1+(zR/z0)^2);
Guoy=atan(z0/zR);
n=2^11;
l=1:n;
x=(l-(n/2))*wdth/(res*10);
y=((n/2)-l)*wdth/(res*10);
M=ones(n);
[X,Y]=meshgrid(x,y);
RR=sqrt((X.^2)+(Y.^2));

E=(w0/wz)*exp(-(RR.^2)/(wz^2))*exp(-i*k*z0).*exp(-i*k*(RR.^2)/2*Rz)*exp(i*Guoy);

A=M.*E;
```

Function to define 2D rectangular aperture

```
% Function to define rectangular aperture- enter width and height in mm, resolution
% should be entered as 1/10res mm
```

```
function [RECT,x,y]=rectap(width,height,res);

wdth=1e-3;
n=2^11;
I=1:n;
x=(I-(n/2))*wdth/(res*10);
y=((n/2)-I)*wdth/(res*10);
RECT=zeros(n);
[X,Y]=meshgrid(x,y);
[X,Y]=meshgrid(x,y);
A=abs(X)<=(width*wdth)/2;
B=abs(Y)<=(height*wdth)/2;
RECT(A&B)=1;
```