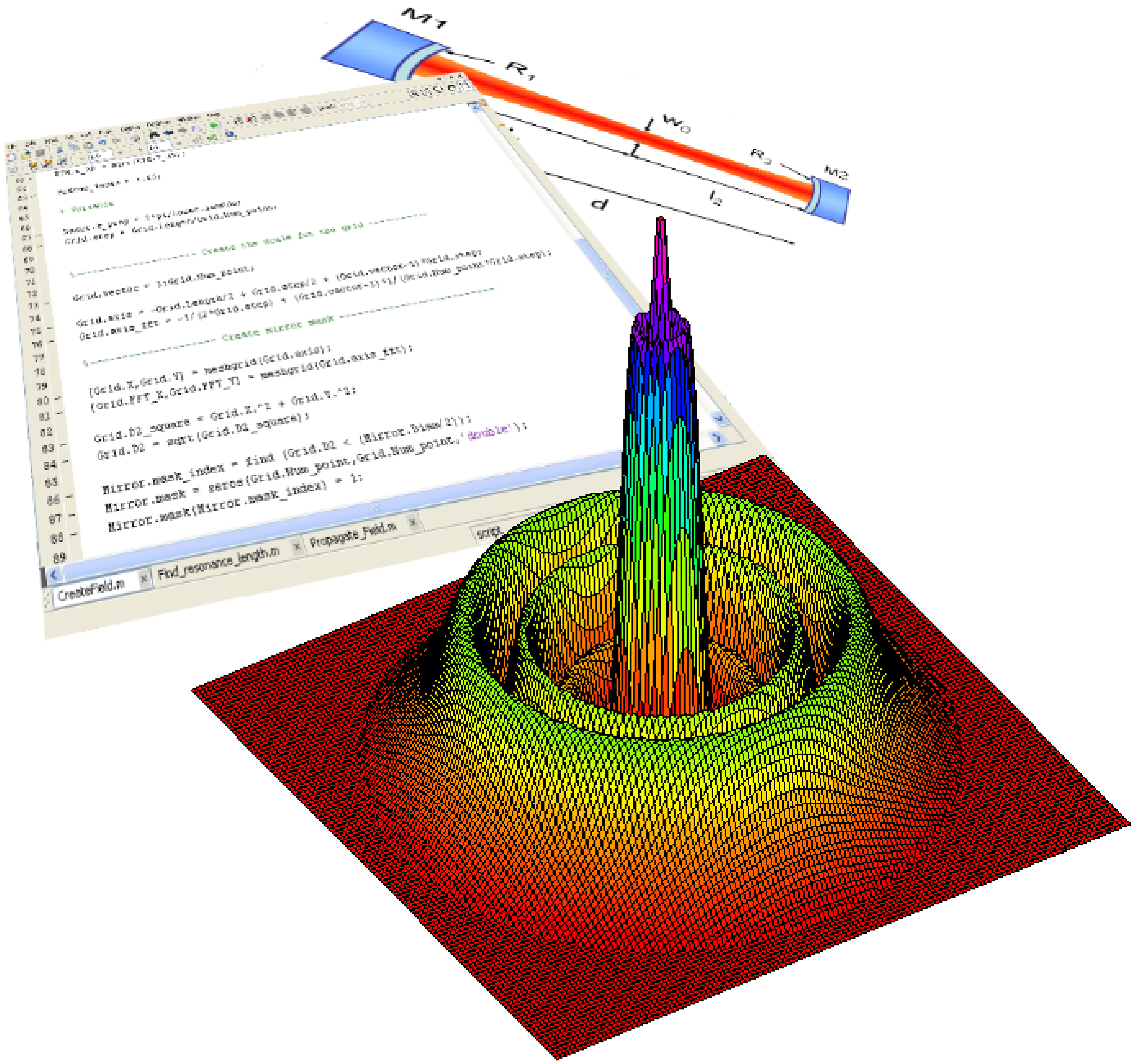
OSCAR

A matlab based optical FFT code



Jerome Degallaix

OSCAR

The Non-Definitive Guide

First edition	July 2008
Second edition	October 2008
Third edition	April 2009

OSCAR, the Matlab code, this documentation as well as the examples were written by:
Jerome Degallaix
Albert Einstein Institute
Callinstrasse 38
D-30167 Hannover
Germany

OSCAR is an optical FFT code used to calculate the steady state optical field circulating in Fabry Perot cavities. The code can integrate non-sperical mirrors and any arbitrary input fields. Recent applications for OSCAR have been: calculation of thermal lensing effect and calculation of diffraction loss and cavity eigen modes for mesa beams. One great advantage of OSCAR is the simplicity and flexibility of the code, everyone with only minimal knowledge of Matlab can easily modified OSCAR code to suit specfic purposes.

If you find a bug when using OSCAR, please feel free to contact the author. Do not hesitate also to report any unclear part from this manual or from the code itself. Of course I am also open to suggestions and possible improvements.

Contents

Chapter 1

Principle of the FFT code

In this chapter, we detail the two main steps used to propagate an electric field in a Fabry-Perot cavity. Firstly we will understand how the propagation of any arbitrary electric fields can be simulated using a Fourier transform. And secondly, we will see how any wavefront distortions encountered by the electric fields can be included in the numerical simulations. Whenever possible some simple Matlab codes are presented to show the numerical implementation of the algorithm. For clarity, the snippets of code are **not** optimised.

1.1 Why OSCAR ?

OSCAR is a FFT code which is able to simulate Fabry Perot cavities with arbitrary mirror profiles. One of the key features of OSCAR is the possibility to easily modify the code to suit the user purpose. For this reason, OSCAR is written with the Matlab language, one can import/export files (mirror maps or cavity eigen modes profile for example), create a ring cavity, create batch file or plot 2D optical field with little programming skill[1]. The fact that I encourage everyone to understand the code of OSCAR in details and to also modify it, is the main reason for this lengthy manual.

1.1.1 Origin of OSCAR

I started writing and using OSCAR during my PhD where I had to simulate the effects of thermal lensing in high optical power cavities. Wavefront distortions induced by the optical absorption have seldom spherical profiles,

[1]The code can also be useable with the free Matlab-alternatives such as Sci-Lab or Octave with only minor modifications.

which means it is delicate to quantify thermal lensing effects with modal expansion code such as Finesse. To simulate the effects of the optical absorption on the mirror, I was using the fem package Ansys which can calculate the temperature distribution inside the mirror substrate or the surface deformations due to the absorption of a Gaussian beam. From the Ansys results, it is straight forward to calculate the wavefront distortion induced by thermal lensing. However, at this stage, I was facing a problem, how can I estimate the effects of the thermal lens if it has to be inserted in a Fabry-Perot cavity ? how much the circulating power will decrease ? Will the cavity optical modes keep their Gaussian profiles ? I needed an optical simulation tool which can be used with any arbitrary mirror profile. OSCAR was born. For the inquisitive reader, the name OSCAR is the acronym of Optical Simulation Containing Ansys Results.

The first version of OSCAR was written with the software IGOR. This code was then translated to Matlab and used to calculate diffraction losses by my UWA colleague Pablo Barriga. Finally, I re-wrote and optimised the Matlab code to substantially decrease the OSCAR computational time.

1.1.2 Which results could I get from OSCAR ?

OSCAR is a versatile tool to simulate Fabry Perot cavities. The following is a (non-exhaustive) list of the results which can be obtained with OSCAR:

- calculate the Gouy phase shift between higher order optical modes. It may be useful for flat beams for example, where no analytical calculations of the Gouy phase shift has been derived yet (as far as I know)

- calculate the coupling loss between the input beam and the cavity eigen modes in case mode mismatching

- calculate the circulating beam (intensity and profile) inside the cavity, which may be different from the eigen modes if the cavity has a low finesse

- calculate diffraction loss and eigen modes of cavity for arbitrary mirror profiles

- calculate the effects of imperfect optics, due to the micro-roughness of the coating surface or thermal lens inside the substrate of the mirrors.

1.1.3 What OSCAR is not for

OSCAR is designed to simulate anything which can be derived from the steady state, classical, optical field circulating inside a Fabry Perot cavity. It means OSCAR does not take into account radiation pressure or quantum effect. OSCAR (in the present version) is not suitable to compute error signals or a whole Michelson interferometer.

1.2 Propagation of an optical field

In this section we will see how it is possible to propagate a coherent electric field, i.e. a laser beam, with the help of the Fourier transform. The principle of the FFT optical propagation code is similar to the Fourier transformation method used to calculate the response of a linear system to an arbitrary function $f_i(t)$. The function $f_i(t)$ is expressed as a continuous superposition of harmonic functions of different frequencies ν:

$$f_i(t) = \int_{-\infty}^{\infty} \widetilde{f}_i(\nu) \exp(2\pi j \nu t) d\nu \tag{1.1}$$

Where $\widetilde{f}_i$ is the Fourier transform of the function f_i. The function $t \rightarrow \exp(2\pi j \nu t)$ is the harmonic function of frequency ν. These are the basic functions used to expand f_i. If the response of the system is known for each elementary harmonic function $\exp(2\pi j \nu t)$, the response of the system to the input f_i can be derived using three steps. First step, the function f_i is expanded in the sum of basic harmonic functions by doing a Fourier transformation. Second step the response of the system to each harmonic function is calculated. This is done simply by a multiplication in the frequency domain. Finally, third step, the output of the system to the input f_i is derived by doing the inverse Fourier transformation of the output harmonic functions (step similar to the equation 1.1).

The FFT optical propagation code follows exactly the principle described in the previous paragraph. In this case the elementary functions are plane waves of different spatial frequencies.

1.2.1 Propagation of a plane wave

We can first understand how to propagate of an elementary plane wave u through free space. At $z = 0$, u can be simply written:

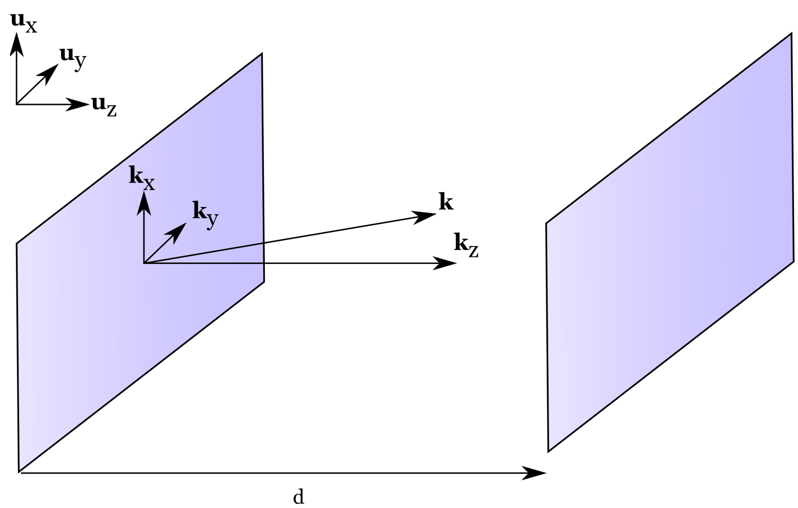


Figure 1.1: Propagation of a plane wave along the z axis.

$$u(x,y,0) = \exp(-jk_x x - jk_y y) \quad (1.2)$$

with $k_{x/y}$ the propagation constant along the x/y axis. Equation (1.2) can also be written:

$$u(x,y,0) = \exp(-j2\pi(\nu_x x + \nu_y y)) \text{ with } \nu_x = k_x/(2\pi) \text{ and } \nu_y = k_y/(2\pi) \quad (1.3)$$

If u is written as shown in equation (1.3), it is easy to recognise u as a harmonic function of two variables (x,y) with respective spatial frequencies [2] (ν_x, ν_y). If we propagate u along the z axis over a distance d:

$$u(x,y,d) = \exp(-jk_x x - jk_y y - jk_z d) \quad (1.4)$$

with $k = (k_x^2 + k_y^2 + k_z^2)^{\frac{1}{2}} = 2\pi/\lambda$. If we suppose the wave to be traveling in a direction close to the z axis (paraxial approximation) as shown in figure 1.1, $k_z \gg k_x$ and $k_z \gg k_y$, then k_z can be written as:

$$k_z = (k^2 - k_x^2 - k_y^2)^{\frac{1}{2}} \simeq k - \frac{(k_x^2 + k_y^2)}{2k} \quad (1.5)$$

$$\simeq k - \lambda\pi(\nu_x^2 + \nu_y^2) \quad (1.6)$$

[2]The term spatial frequency ν must be understood as the number of wavelength per unit of length $\nu = 1/\lambda = k/2\pi$

So we can rewrite equation (1.4) as:

$$u(x, y, d) = u(x, y, 0) \exp(-j(k - \lambda\pi(\nu_x^2 + \nu_y^2))d) \tag{1.7}$$

The equation (1.7) is the foundation of the FFT propagation code. It tells us that the propagation of a plane wave along the z axis over a distance d can simply be represented by a phase shift. The function $d \rightarrow \exp(-j(k - \lambda\pi(\nu_x^2 + \nu_y^2))d)$ could also be seen as the propagation operator for an input plane wave traveling a distance d under the paraxial approximation. The term $\exp(-j(kd))$ represents the phase shift of a plane wave propagating along the z axis and the term $\exp(j\lambda\pi(\nu_x^2 + \nu_y^2)d)$ adds a phase correction to take into account the fact that the wave propagates with a small angle with respect to the z axis[3].

We noticed in equation (1.3), that the elementary plan wave $u(x, y, 0)$ can also be seen an harmonic function with spatial frequency ν_x and ν_y. To keep the analogy developed in the introduction of this chapter with the classical time domain Fourier transform, the plane wave is equivalent to the elementary harmonic function $t \rightarrow \exp(2\pi j\nu t)$ of frequency ν. Since we know how to propagate a plane wave, we understand now how it may be possible to propagate any arbitrary field if we can manage to expand it as a superposition of elementary plane waves.

1.2.2 Propagation of an arbitrary field

Equation (1.7) tells us how to propagate a plane wave. So to propagate any arbitrary electric fields E, we need to know how to expand the electric field onto the set of plane waves $\exp(-j2\pi(\nu_x x + \nu_y y))$. We would like to find $\widetilde{E}(\nu_x, \nu_y)$ such as:

$$E(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \widetilde{E}(\nu_x, \nu_y) \exp(-j2\pi(\nu_x x + \nu_y y)) d\nu_x d\nu_y \tag{1.8}$$

With $\widetilde{E}(\nu_x, \nu_y)$ the complex amplitude of the component of the field E with spatial frequency (ν_x, ν_y). We recognise equation (1.8) as an inverse 2D Fourier transform[4]. Using the properties of the Fourier transform[2] we deduce the expression for $\widetilde{E}(\nu_x, \nu_y)$:

[3] The angle θ_x between the direction of propagation and the x axis is $\theta_x = sin^{-1}(k_x/k)$. In the case of small angles, it is simply $\theta_x = \lambda\nu_x$

[4] In fact, if we respect the convention found in signal processing technique, equation (1.8) is not an inverse Fourier transform but a Fourier transform[1]. It is not tragic, since we will stay consistent with the convention presented here.

$$\widetilde{E}(\nu_x, \nu_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} E(x, y) \exp(j2\pi(\nu_x x + \nu_y y)) dx dy \tag{1.9}$$

Combining equations (1.7), (1.8) and (1.9) , we know how to propagate in free space a transverse electric field E from $z = 0$ to $z = d$. To calculate the resulting field after the propagation, three steps are required:

1. Decomposition of the field $E(x, y, 0)$ into a sum of elementary plane waves. Mathematically, this step represents a 2D Fourier transformation.

$$\widetilde{E}(\nu_x, \nu_y, 0) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} E(x, y, 0) \exp(j2\pi(\nu_x x + \nu_y y)) dx dy \tag{1.10}$$

2. Propagation of each plane waves, which is equivalent of adding a phase shift in the frequency domain.

$$\widetilde{E}(\nu_x, \nu_y, d) = \widetilde{E}(\nu_x, \nu_y, 0) \exp(-j(k - \lambda\pi(\nu_x^2 + \nu_y^2))d) \tag{1.11}$$

3. Recomposition of the electric field from the propagated plane waves. This step is in fact an inverse Fourier transformation.

$$E(x, y, d) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \widetilde{E}(\nu_x, \nu_y, d) \exp(-j2\pi(\nu_x x + \nu_y y)) d\nu_x d\nu_y \tag{1.12}$$

1.2.3 My first Matlab FFT code

There is an essential point to realize before implementing the three analytical steps described in the previous section. The computer does not deal with continuous electrical fields which means all the data fields have to be discretized. For example the amplitude of a Gaussian beam will be represented by a square matrix (called also grid later), each points of the matrix (called also pixel for convenience) will represent the amplitude of the Gaussian beam at a defined location. A 2D plot of such a matrix is shown in the top left corner of the figure 1.2.

Practically, the discretization process is governed by the choice of 2 parameters: the physical size represented by the matrix and the number of points in the matrix. For example, to discretize a laser beam with a beam radius of 1 cm we can use a matrix of size 128×128 representing an area of 10 cm by

Listing 1.1: Discretization of a Gaussian beam

```
Grid.Num_point = 128;                          % Number of point in one side the grid
Grid.Length = 0.10;                            % Physical dimension of the grid in meter
Grid.step = Grid.Length/Grid.Num_point;        % Physical size of one pixel of the grid

Grid.vector = 1:Grid.Num_point;                % Grid.vector = 1 2 3 ... Grid.Num_point
% Calculate the spatial scale used for each pixel:
Grid.axis = -Grid.Length/2 + Grid.step/2 + (Grid.vector-1)*Grid.step;

Field.Gaussian = zeros(Grid.Num_point,Grid.Num_point,'double');
Laser.amplitude = 1;                           % Arbitrary amplitude
Laser.waist = 0.01;                            % waist of the laser beam in meter

% Fill the matrix representing the (real) Gaussian beam:
for m = 1:Grid.Num_point
    for n = 1:Grid.Num_point
        Field.Gaussian(m,n) = Laser.amplitude * exp(-(Grid.axis(m)^2+Grid.axis(n)^2)/Laser.waist^2);
    end
end
```

10 cm. This example can be implemented in Matlab in a straight forward way as shown in the listing 1.1.

If the scale of the matrix (called *Grid.axis* in the listing 1.1) representing the amplitude distribution is easy to understand, a more delicate point is the scaling of the Fourier transform of the input beam. This scaling of the spatial frequency is required since we need to know the spatial frequency represented by each pixel of the discrete Fourier transform of the Gaussian beam. Concretely we need to know the discrete values of ν_x, ν_y from equation 1.10.

First thing to understand is that the discrete Fourier transform of a 2D complex matrix is also a 2D complex matrix with the same dimensions [3]. The low spatial frequencies are located in the middle of the matrix and the high spatial frequencies on the edge. Typically if the original matrix has for dimensions $N \times N$, the spatial frequency 0 (the average component) is located at the index $(N/2+1, N/2+1)$[5]. Meanwhile, the frequency separation $\Delta\nu$ between 2 adjacent pixels of the Fourier matrix is:

$$\Delta\nu = \frac{1}{Grid.Length} = \frac{1}{N \times Grid.step} \tag{1.13}$$

[5] In fact the FFT algorithm returns the Fourier transform of the input matrix with the low spatial frequency spread at the four corners of the matrix. However for a better readability, the low frequency are then shifted back to the center of the matrix.

With *Grid.Length* and *Grid.step* the variables defined in the listing 1.1. So the minimal (negative) spatial frequency calculated is $-N/2 * \Delta\nu$ often called the Nyquist frequency and the maximal spatial frequency is $(N/2-1) * \Delta\nu$. An example for the frequency scale of the Fourier transform of the matrix is presented in the top right plot of the figure 1.2.

Special attention must be taken to understand the vertical and horizontal scales in the figure 1.2. The numbers written for the scales are in fact the value of the scale between 2 pixels as it can be seen by zooming on the ticks. So for example to know what is the spatial frequency of the first top row (horizontal line) of the Fourier matrix on the top right plot which represents the maximal spatial frequency, we have to take the average of the top two ticks. So the maximal spatial frequency is $1/2 * (96.875 + 90.625) = 93.75\mathrm{m}^{-1}$, which is as expected equal to $(N/2-1) * \Delta\nu = 15/0.16 = 93.75$

One natural question we could ask is what is the good size of the grid ? The size of the grid must be large enough to represent faithfully the Gaussian beam, no substantial energy from the beam must lay outside the grid. So the dimension of the grid must be *at least* 3-4 times bigger than the biggest beam diameter encountered. A large grid size is also important to sample properly the low special frequency as shown at the bottom of figure 1.2. Remarkably even if the laser beam radius increased as the beam propagates from its waist, the amplitude of the Fourier transform is constant along the propagation. Indeed the propagation as shown in equation 1.11 is simply represented by a phase shift in the Fourier domain.

After the physical size of the grid has been chosen, the second important parameter to be decided is the number of points in the grid. Because of the way the discrete Fourier transform is calculated in the FFT algorithm, it is strongly recommended to choose the number of points for the side of the matrix to be a power of 2. Usually a good compromise between speed and accuracy is given for N =64,128 or 256. The influence of the number of points on the accuracy of the results can (and should) always be checked by doing the same simulations with different mesh of the grid.

The size of the grid divided by the number of pixels is the dimension represented by one pixel, in other word the resolution of the grid (variable *Grid.step*). Of course, the resolution of the grid must match the size of the physical feature we would like to simulate. For example it is useless to try to simulate the effect of features having a size 1 mm in a mirror map with a grid resolution of 1 cm.

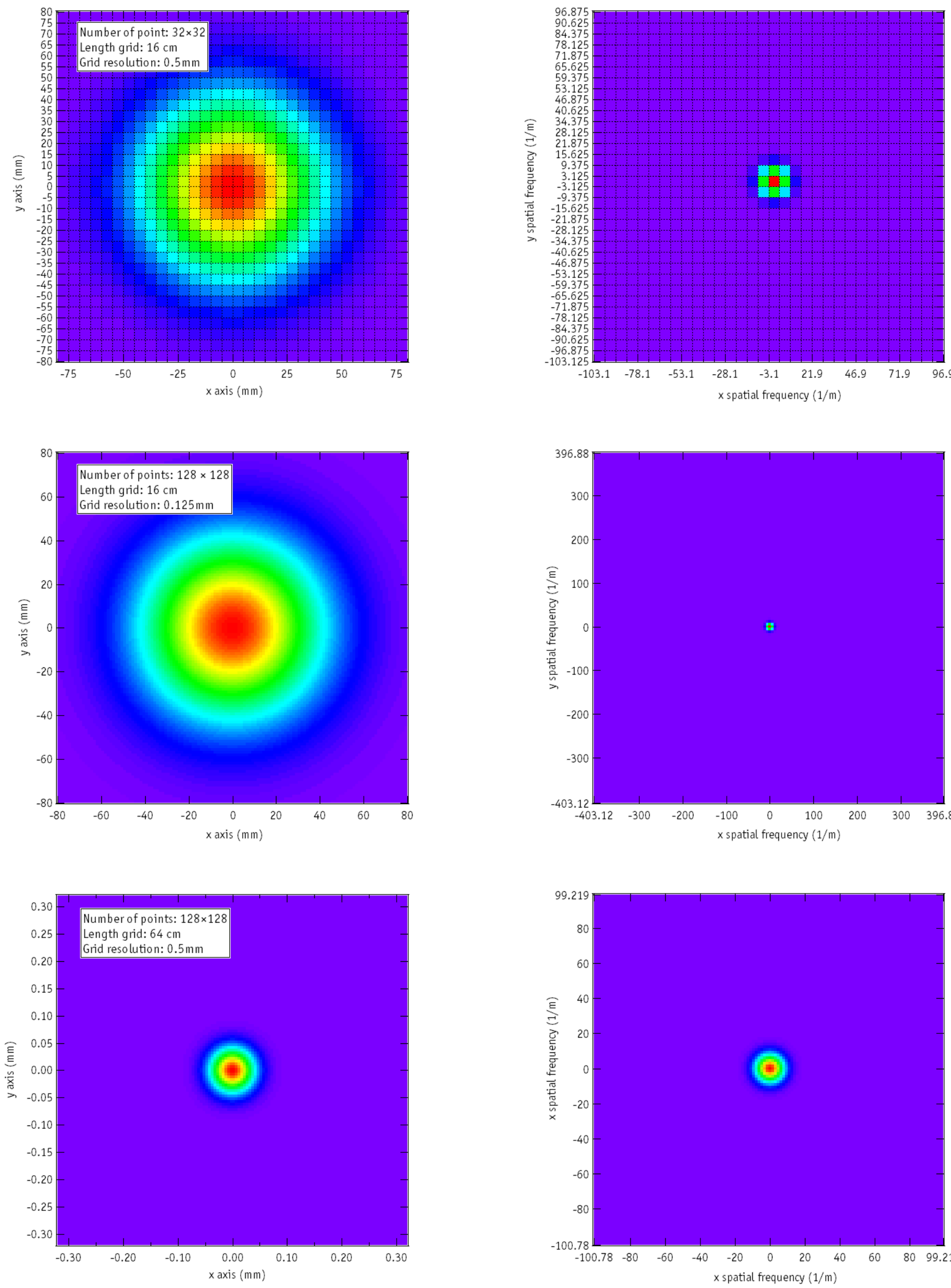


Figure 1.2: Example of the influence of the physical size of the grid and the number of points used for the representation of a Gaussian beam (left column) and its Fourier transform (right column). As expected intuitively, to have a proper representation of both the Gaussian beam and its Fourier transform it is essential to have simultaneously a large space window (parameter: *Grid.Length*) and a large number of points (parameter: *Grid.Num_point*). For this example the waist of the laser beam is 4 cm.

Listing 1.2: The code used to propagate the matrix Field.Start

```
% Distance of propagation in meter
Distance_prop = 100;

% Spatial frequency of the pixels in the Fourier space
Grid.axis_fft = -1/(2*Grid.step) + (Grid.vector-1)*1/(Grid.Num_point*Grid.step);

% Define the propagation matrix

Mat_propagation = zeros(Grid.Num_point,Grid.Num_point,'double');

for m = 1:Grid.Num_point
    for n = 1:Grid.Num_point

        Mat_propagation(m,n) = exp(i*(-Laser.k_prop*Distance_prop + ...
            pi*Laser.lambda*(Grid.axis_fft(m)^2 + Grid.axis_fft(n).^2)*Distance_prop));

    end
end

%------------------- Propagate the field -----------------------------

Field.Fourier = fftshift(fft2(Field.Start));          % Do the Fourier transform of the input field
Field.Fourier = Field.Fourier .* Mat_propagation;  % Do the propagation in the frequency domain
Field.End = ifft2( ifftshift (Field.Fourier));          % Do the inverse Fourier transform

% As a result Field.End represents Field.Start propagated over 100 meters
```

We just saw how a square matrix can be used to represent a discrete electric field, so now we can try to simulate its propagation it using the 3 consecutive steps described by equations (1.10), (1.11) and (1.12). During the second step, the Fourier transform of the electric field is multiply by a complex number depending of the distance of propagation and also the spatial frequency. Practically this multiplication is achieved by multiplying pixel by pixel 2 matrices: the Fourier transform matrix with a propagation matrix. The propagation matrix is usually defined before hand and only once since it will be used repeatedly as we will see later in section 1.4. The matlab code at the core of the FFT code is presented in the listing 1.2, it is the direct sequel from the previous listing where we defined the matrix used to represent the electric field.

The propagation of an electric field using a FFT code is an extremely pow-

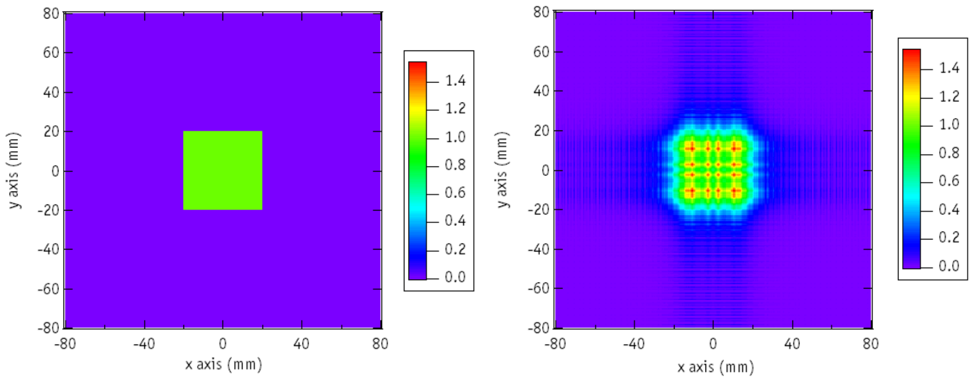


Figure 1.3: Propagation of an uniform square of light using our FFT propagation code. The initial square light field is presented on the left plot. The field resulting from the propagation of the initial field over 100 m is shown on the right plot. Structures similar to the Hermitte Gaussian modes begin to emerge when the square field propagates in free space.

erful tool. With the FFT code we can propagate any arbitrary profile of the laser beam, not only the beams from the usual Hermite Gaussian set. Such an example is presented in figure 1.3. The code used to produce these two plots is given with the OSCAR distribution, the name of the Matlab script is My_First_FFT_code.m. The initial field is a theoretical square of uniform amplitude (left plot). The resulting field after the propagation in free space over 100 m is shown in the right plot. A similar result could have been obtained based on the propagation of Hermite Gaussian modes. However to have an accurate representation of the initial field, a very large number of the higher order modes must have been taken into account which requires large amount of computer processing power. The field is discretise on a 1024×1024 matrix. The physical size of the grid is 16 cm $\times$ 16 cm.

We arrive now at the end of this section which is dedicated to understand how we can numerically simulate the propagation of an arbitrary laser beam. However having a code only capable to propagate a beam is seldom useful. We are usually more interested to simulate real optical system with mirrors, aperture and imperfect optics. How to include optics in the code is the subject of the next section.

1.3 Adding realistic optics

It is time now to introduce in our simulation two essential optical components: mirrors and lenses. Theses components alter the beam wavefront radius of curvature and so are used for beam shaping (in clear: they make the laser beam smaller or bigger). This can be easily implemented in OSCAR as we will discover in the following paragraphs.

1.3.1 Arbitrary wavefront distortion

Any wavefront distortion can be characterised by its induced optical path length difference $\Delta OPL(x, y)$. In a general manner, when the laser beam crosses a medium of non-uniform refractive index $n(x, y, z)$ the optical path length $OPN(x, y)$ along the optical axis parallel to the 'z' direction can be defined as:

$$OPN(x,y) = \int_0^L n(x,y,z)dz \tag{1.14}$$

With L the length of the medium. Since we are not interested in any constant offset due to the optical path length, it is often more relevant to introduce the optical path length difference ΔOPN as:

$$\Delta OPN(x,y) = \int_0^L n(x,y,z)dz - \int_0^L n(0,0,z)dz \tag{1.15}$$

The laser field E_i passing trough an element inducing a wavefront distortion characterised by $\Delta OPN(x, y)$ get an additional space dependent phase shift according to:

$$E_o(x,y) = E_i(x,y)\exp\left(-jk\Delta OPL(x,y)\right) \tag{1.16}$$

As we can see the effect of the wavefront distortion can be implemented in the physical space and it is not related to any Fourier transform. In fact in any optical FFT code, the Fourier transform is only used to propagate the electric field over a certain distance. Any other calculations are made in the usual physical space.

1.3.2 Mirrors

One of the most useful wavefront distortion is the one induced by mirrors. A mirror is a reflective spherical surface which is used to steer (flat mirror) or focus the beam (convergent or divergent mirrors). From simple geometrical

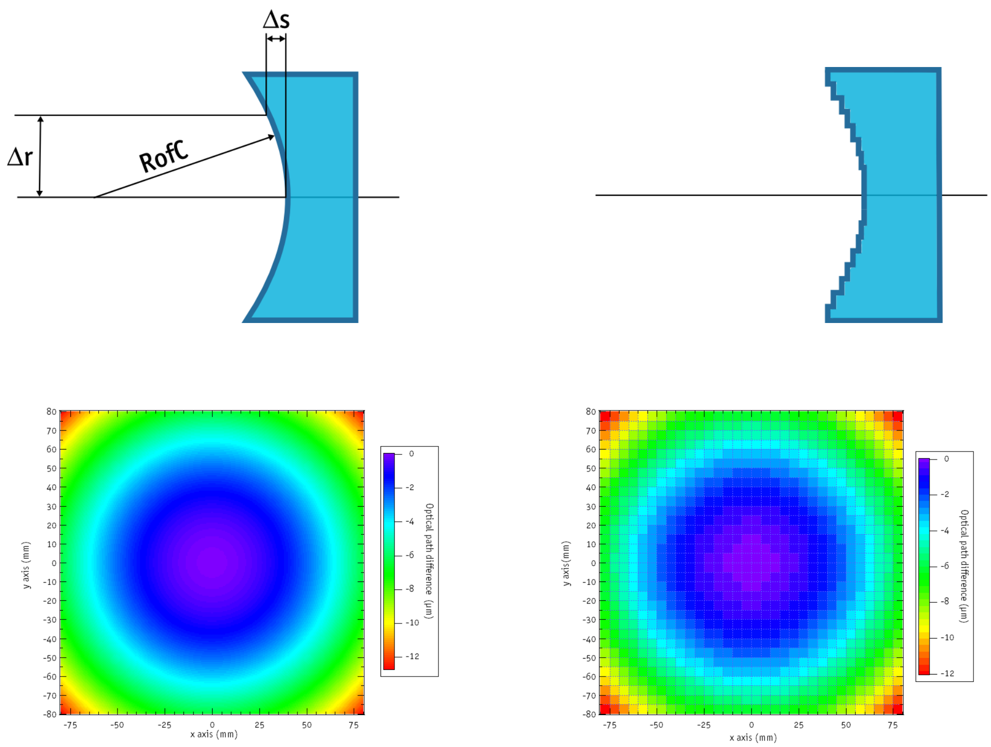


Figure 1.4: Example of the optical path difference induced by 1 km radius of curvature mirror. The optical path difference is simply two times the sagitta of the mirror. The right plot is the left plot discretised on a grid of 32 $\times$ 32 pixels.

consideration (see the top left plot in figure 1.4) we can calculate the change in sagitta Δs as a function of Δr the distance from the mirror center:

$$\Delta s = RofC - \sqrt{RofC^2 - \Delta r} \tag{1.17}$$

With $RofC$ the radius of curvature of the mirror. The optical path difference is simply two times the sagitta:

$$\Delta OPN(x, y) = 2\left(RofC - \sqrt{RofC^2 - \sqrt{x^2 + y^2}}\right) \tag{1.18}$$

Of course since we are using a numerical simulation, the optical path difference representing the mirror has also to be discretized using the same grid as the one used for the laser beam. A discretized optical path difference for a mirror is represented on the right part in the figure 1.4.

There is no difficulty to create with Matlab the matrix used to represent the optical path difference induced by a mirror. The value of the optical path as a function of the coordinate (x, y) has previously been shown in equation 1.18. The direct Matlab implementation is presented in the listing 1.3. By definition in OSCAR a concave mirror has a positive radius of curvature, which means the optical path difference is negative.

Listing 1.3: The code used to create the mirror matrix

```
% Definition of the mirror radius of curvature in meter
Mirror.RofC = 1000;

% Create mirror grid
Mirror.OPN = zeros(Grid.Num_point,Grid.Num_point,'double');


for m = 1:Grid.Num_point
    for n = 1:Grid.Num_point

        Radius_sqr = (Grid.axis(m)^2+Grid.axis(n)^2);
        Mirror.OPN(m,n) = -2*(Mirror.RofC - sqrt(Mirror.RofC^2 - Radius_sqr));

    end
end
```

If the mirror is not perfectly spherical because of thermal lensing effect or because the mirrors are part of a flat beam cavity, we simply have to modify the equation 1.18 accordingly by adding the known deviation.

1.3.3 Lenses

In OSCAR, we use exactly same procedure as that for mirrors to simulate lenses. The optical path difference induced by the lens is the same as that induced by a mirror whose radius of curvature is two times the focal length of the lens we wish to simulate. The fact that the lens is used in transmission and a mirror in reflection is not relevant in OSCAR. Indeed the evolution of the laser beam confined between two mirrors can always be simulated by the propagation of a laser beam passing through a periodic system of lenses[4].

1.3.4 Aperture

Apertures can be represented by two complementary physical areas: one area transmits integrally the light falling on it whereas the other area blocks integrally the light. Apertures are useful to simulate correctly finite size optics. In OSCAR, the optical path difference representing a mirror is defined over the whole calculation grid independently of the real size of the mirror. To simulate a finite size mirror, we multiply the reflected field by an aperture which has the same diameter as that of the mirror. The aperture simulates the fact that any light falling outside the mirror is lost.

Practically, an aperture $A(x, y)$ is represented by matrix of 0 and 1. A 0 at the position (x, y) indicates that the light is blocked and a 1 indicates that the light is transmitted. An example of a circular aperture is presented in figure 1.5. So to simulate the reflection from a finite size mirror, we can include the aperture effect in the previous equation 1.16:

$$E_o(x, y) = E_i(x, y) \exp\left(-jk\Delta OPL(x, y)\right) A(x, y) \tag{1.19}$$

As for the optical path induced by the mirror, we can also define a matrix representing the aperture. In the OSCAR code this matrix is called $Mirror.mask$. This matrix is only filled with 0 and 1, 1 when the pixel is inside the mirror and 0 otherwise. The simple Matlab code to create a circular aperture in OSCAR is described in the listing 1.4.

Listing 1.4: The code used to create a circular aperture

```
% Define aperture diameter in meter
Aperture_diameter = 0.1;

Mirror.mask = zeros(Grid.Num_point,Grid.Num_point,'double');

for  m = 1:Grid.Num_point
    for  n = 1:Grid.Num_point

        Radius = sqrt(Grid.axis(m)^2+Grid.axis(n)^2);
        if  (Radius < Aperture_diameter/2)
            Mirror.mask(m,n) = 1;
        end
    end
end
```

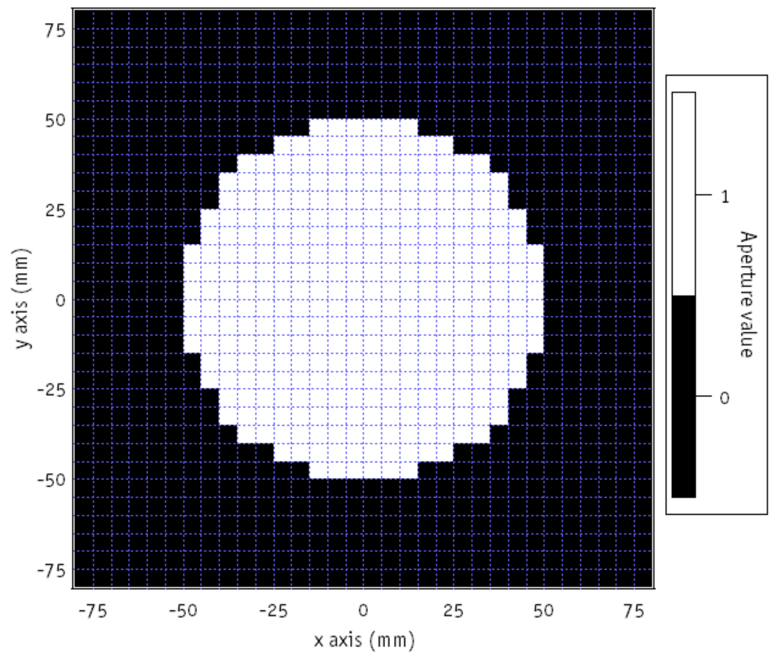


Figure 1.5: Plot of the 2D matrix representing a circular aperture of diameter 10 cm. The number of points for the grid is 32×32 points and the physical size of the grid is 16 cm $\times$ 16 cm.

1.3.5 Code implementation

To simulate the reflection of an electric field by a mirror (or a transmission through a lens), we define a function called *Propa_mirror*. The function takes for parameters the input electric field, the optical path difference induced by the mirror and the reflectivity of the mirror as shown in the listing 1.5 . The output of the function is the electric field after the reflection on the mirror. The function is a direct implementation of the equation 1.16.

The aperture of the mirror is also included in the reflection however it is not an argument of the function *Propa_mirror* since we suppose that the mirrors have all the same diameter, so the aperture matrix is constant[6]. in the function, the reflectivity of the mirror is scalar, which means the reflectivity is homogenous and constant over the mirror surface. The reflectivity of the

[6] The function can easily be modified if the mirrors have different sizes

mirror can also be defined as a matrix if the reflective coating is not perfectly uniform.

Listing 1.5: The function used to simulate the reflection of an electric field by a mirror

```
function Output = Propa_mirror(Wave_field, Wave_mirror, reflectivity)

global Mirror;
global Laser;

Output = Wave_field .* exp(i * Wave_mirror*Laser.k_prop) * reflectivity .* Mirror.mask;
```

1.4 Simulating a Fabry Perot cavity

Since we have seen in the two last sections how to propagate a laser beam in free space (section 1.2) and how to simulate the reflection by a mirror (section 1.3), we have all what we need to simulate a Fabry Perot cavity.

A Fabry Perot cavity is usually constituted by two mirrors facing each other. Between these two mirrors, a light field is circulating, bouncing back and forth between the two reflective coatings. One of the main interest of the Fabry Perot is that the optical power of the circulating field can be much higher than the power of the input field. With OSCAR, it is possible for a given input field to calculate the total circulating power, reflected power and transmitted power as well as the spatial profile of all the light fields.

The method used by OSCAR to calculate the circulating field in a Fabry Perot cavity is well known for most readers. Indeed, the same method is often used in undergraduate lectures to calculate analytically the circulating field in the cavity[5]. OSCAR calculates the circulating field by propagating back and forth the laser beam between the two mirrors and then summing all the fields at one particular plane as shown in figure 1.6.

In more details, we can write the OSCAR algorithm used to compute the circulating power. Using the notation from the figure 1.6 the different consecutive steps can be described as:

1. Define the cavity parameters as well as the mirror profiles and the input beam.

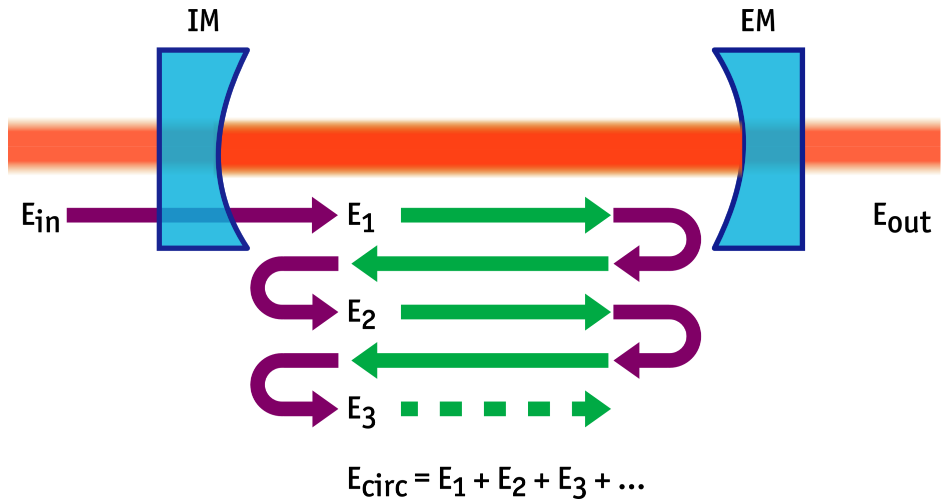


Figure 1.6: Description of the algorithm used in OSCAR to calculate the circulating field in a Fabry Perot cavity. The violet arrows represent a change in phase for the light field which is described by equation 1.16 and the green arrows represent the propagation of the light field using a FFT code. From E_i to E_{i+1}, the light field has undergone one round trip in the cavity. IM and EM stand respectively for Input Mirror and End Mirror.

2. Propagate the input beam E_{in} through the input mirror . For this purpose the input mirror can assimilated to a lens, so the listing 1.5 can be used. As a result, we obtain the field E_1.

3. After one round trip in the cavity, the field E_1 becomes E_2. One round trip in the cavity consists specifically of one propagation through the cavity length using the FFT code, one reflection on the end mirror, another propagation back to the input mirror and then finally a reflection on the input mirror.

4. Repeat the last operation to create the set of electric field E_i.

5. Then sum all the field E_i to have the cavity circulating power E_{circ}. The number of light field E_i to be considered to have an accurate result depends on the finesse of the cavity.

6. The transmitted field E_{out} is simply the circulating field transmitted through the ETM.

In this above pseudo-code, we did not mention any resonance condition to maximise the circulating power in the cavity. Practically, we should always define the round trip phase shift for the field in the cavity (or a microscopic position shift for the cavity length) before calculating the cavity circulating power. The round trip phase shift allows us to set the cavity to be resonant for the fundamental mode or any other optical modes if necessary. The procedure to find the suitable resonance length is detailed in the next chapter in section 2.3.

1.5 Further reading

In this chapter, only a general and simplistic view of the FFT code has been presented. To have a better understanding of the power and limit of the code some further reading is highly recommended. Here some suggestions:

- The thesis from Brett Bochner[6] which programmed one of the first FFT code used by LIGO is a mine of treasures. The chapter two of this thesis about the technical realization of an optical FFT code contains some essential issues to grasp for a successful code, for example how the radius of curvature of the mirror can set an upper limit on the resolution of the grid, the use of aliasing filters or methods to calculate the steady state fields.

- Andrew Trigdell wrote also a FFT code in the ANU group. His master thesis may be hard to find but a following article[7] gives some good insights of the procedures involved in the code.

- A wonderful article to understand how to create a simple and robust FFT code to simulate high finesse cavity has been written by Partha Saha. The explanation how to calculate the steady state cavity circulating field is crystal clear and very elegant (however the end of the article may appear a little bit more obscure at the first reading).

- Technics to build a FFT code (and some other optical numerical codes) can be found in the recent thesis of Juri Agresti[8]. This thesis is also an excellent example about how to deal with non spherical mirror.

- One of the first article describing how to use a FFT to calculate the eigen modes of a cavity by Gordon and Li[9]. Although the article is 40 years old, all the modern FFT program are still based on the fundamental method explained in this paper.

Chapter 2

The OSCAR code in details

This chapter is a step by step guide to run OSCAR. Using the simple example of a Fabry Perot cavity, we will describe the essential procedures and give some hints in order to obtain valid results. The different scripts for this example can be found in the folder entitled Calculate_Pcirc. To run the full simulation, execute the Matlab script called Run_OSCAR.m.

2.1 The cavity to simulate

We are interested to simulate a Fabry Perot cavity with a slightly mismatched input beam. The physical parameters of the optical system to simulate are summarized in table 2.1. After defining our cavity, we have to choose 2 essential parameters for our simulation: the physical size of the grid *Grid.Length* and the number of points of the grid *Grid.Num_point*. Our grid must include the mirror aperture, so the size of the grid must be at least equal to the mirror diameter. In our case the mirrors have a diameter of 25 cm, so the dimension of the grid could be 30 cm $\times$ 30 cm. Then, we can think about the number points required to sample the laser beam. Of course, a large number of points leads to accurate results but at the price of a lengthy computational time. From experience a grid with 128 $\times$ 128 is a safe choice to simulate a cavity with smooth mirrors.

To fasten the calculations, we suppose the substrate of the input and end mirror to be thin, so the substrates are equivalent to thin lenses for a beam passing through. We also suppose the laser beam to be defined at the input mirror reflective coating but still outside the cavity, so we do not have to propagate the input laser beam in space or in the substrate before the transmission through the input mirror.

Table 2.1: Parameters of the input beam and the Fabry Perot cavity we wish to simulate. The variable name is the name of the variable in the OSCAR program and so also in the Matlab workspace.

Parameters		*Variable name*	Value
Cavity length	(m)	*Length_cav*	1000
Substrate refractive index		*Refrac_index*	1.5
Mirror diameter	(mm)	*Mirror.Diam*	250
Input laser			
Wavelength	(nm)	*Laser.lambda*	1064
Beam radius	(mm)	*Laser.size*	20
Wavefront curvature	(m)	*Laser.radius*	-2000
Optical power	(W)	*Laser.power*	1
Input mirror			
Radius of curvature	(m)	*ITM.RofC*	2500
Transmission		*ITM.T*	0.005
Loss	(ppm)	*ITM.L*	50
Reflectivity		*ITM.R*	1 - (Transmission + Loss)
End mirror			
Radius of curvature	(m)	*ETM.RofC*	2500
Transmission	(ppm)	*ETM.T*	50
Loss	(ppm)	*ETM.L*	50
Reflectivity		*ITM.R*	1 - (Transmission + Loss)

2.2 Declaration and initialisation

The first Matlab script to run when starting a simulation is the script called CreateField.m. During this script all the variables required for the simulation will be defined. That includes as well the matrix representing all the mirror maps (in reflection and transmission), mirror aperture(s) and the matrix describing the input laser beam. For coherence, all the variables must be defined in the International System of Units (SI), which means all the variables representing a length are in meter.

The script CreateField.m is most of the time self explaining and does not require extensive thinking. First the variables *Grid.Length* and *Grid.Num_point* are defined and then all the variables listed in the table 2.1 are given. From the given parameters, the mirror aperture matrix is created (according to the listing 1.4), following by the propagation matrix (listing 1.2), the mirror

$\Delta OPL(x,y)$ maps (listing 1.3) and finally the matrix of the input beam.

For convenience, two matrices extensively used in intermediate calculations are also defined: *Grid.D2* is a matrix where the value of each pixel is the distance between the pixel and the origin, i.e $value = \sqrt{x^2+y^2}$ and *Grid.D2_square* is the previous matrix but with every pixel squared, i.e $Grid.D2\_square = Grid.D2.^2$

2.3 Finding the resonance length

In this section the role of one of the most important script in OSCAR is explained. This script is called Find_resonance_length.m and is used to find the microscopic shift of the cavity length which is required to be on resonance. This script is essential because before calculating the circulating field in a Fabry Perot cavity, the cavity has to be set on resonance where the circulating power of the fundamental mode TEM_{00} will be maximized. In the domain of gravitational wave detection, all the optical cavities of the detector are resonant for the fundamental mode or very near the resonance as in the case of DC-readout or detuned signal recycling.

2.3.1 Setting the resonance the length

The first thing to understand is how OSCAR implements a microscopic length shift of the cavity length. For example, for setting the cavity on resonance we have to shift the cavity length (called *Length_cav*) by a value δL with δL smaller than half a wavelength. The first (and the simplest) idea is to the set a new cavity length to *Length_cav*+δL. This solution is perfectly viable and gives correct result in Matlab for kilometer long cavities. However, from a numerical point of view it may not be the most robust solution since we have to add a length of the order of the kilometer with a length smaller than one micrometer[1].

Another solution to set the cavity on resonance, is to add after each light round trip in the cavity a constant phase shift. This is this solution that we use in OSCAR. So to simulate a cavity length shift of δL, a phase shift of $k2\delta l$ is added after each round trip of the field E_i. Practically, when calculating the circulating field, the matrix of the field E_i is multiplied by a scalar factor $\exp(jk2\delta l)$ just before the field is reaching back the input mirror. To be

[1]For reference, in Matlab the relative accuracy in the number representation can determined with the function eps('double'), on my computer it is of the order 10^{-16}.

consistent with the previous chapter it must have been $\exp(-jk2\delta l)$ however in the OSCAR code, it is implemented as $\exp(jk2\delta l)$, the sign convention can be arbitrary (as soon as it is kept constant for all the procedures).

In OSCAR, the variable which represents the shift in the cavity length necessary to be on the desired resonance is called *Length.reso_zoom*. To determine the right value for *Length.reso_zoom*, the script Find_resonance_length.m has to be run first. By convention, *Length.reso_zoom* is in fact $\delta l/2$ which means that the variable *Length.reso_zoom* represents the shift in the round trip length to make the cavity resonant. With this convention if *Length.reso_zoom* is shifted by one wavelength, the resonance frequency is shifted by one free spectral range.

2.3.2 Finding the resonance the length in details

The principle to find the resonance length of the cavity is quite simple: the cavity circulating power is monitored as the cavity round trip length is scanned over one wavelength. The resonance length *Length.reso_zoom* is the length which maximises the circulating power. A typical plot of the circulating power as a function of the microscopic cavity round trip length is shown in figure 2.1.

The first idea to draw a plot of the cavity circulating power as a function of the cavity tuning is straight forward. We simply run the FFT code to calculate the circulating power for all the different detuning we would like to test. For example, in figure 2.1, the horizontal axis which spans over one wavelength is divided into 2000 points. So we can imagine to run the FFT code, 2000 times for each particular round trip length. This procedure is absolutely correct, however extremely slow.

Practically, in OSCAR to draw the plot in figure 2.1, a technique first described by Gordon and Li[9] is employed. The principle is explained by the follow step:

1. First all the fields E_i are calculated and stored for an arbitrary position of the cavity tuning (so for an arbitrary resonance length). This step is only done once.

2. The circulating field E_{circ} for a particular microscopic round trip length δl is built up by summing all the field E_i with the proper phase shift:

$$E_{circ} = \sum_i E_i \exp(jk\delta l) \tag{2.1}$$

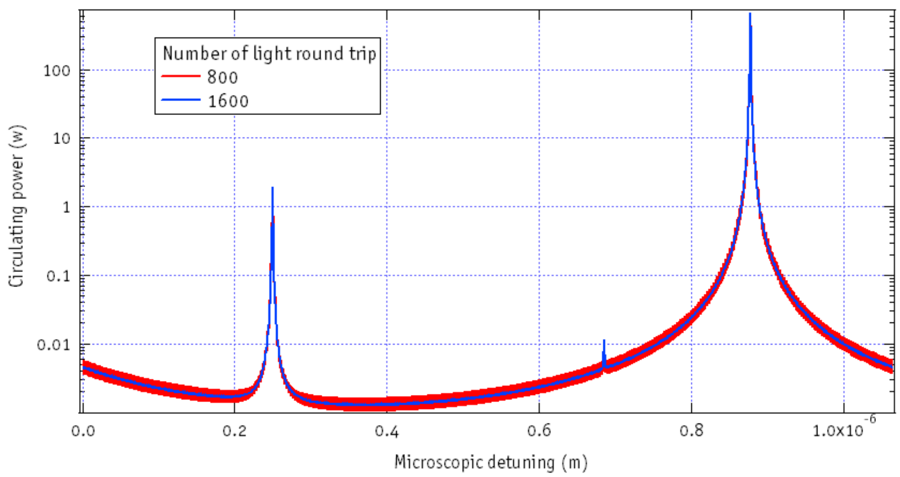


Figure 2.1: Circulating power in the cavity as the function of the macroscopic round trip length detuning scanned over one wavelength. The higher peak indicates the resonance position of the fundamental mode TEM_{00}. We can also notice two smaller peak revealing the presence of higher order optical modes, this is a strong proof that the input beam is not matched with the cavity eigen mode. The figure has been plotted for two values of *Length.nb_iter*, this variable determines the number of field E_i to cumulate when calculating the circulating power. On resonance, the build up circulating power is of course higher when we use 1600 fields instead of 800 (even if it is not obvious in the figure due to the large vertical scale).

3. Repeat the previous step over all the detuning length δl we wish to try.

The advantage of this technique is that the FFT code is just called once to calculate the field E_i, then the reconstruction of the power buildup for the various length detuning is just a matter of summing and multiplication. The only limitation of this technique is that a huge amount of memory is required since all the fields E_i are to be stored. For example if we would like to store 500 complex matrices of 256 $\times$ 256 points, 500 megabytes of free memory is required.

2.3.3 Code implementation

To calculate the resonance length in OSCAR, so to calculate the value of *Length.reso_zoom*, several script are involved. Here the list:

- Find_resonance_length.m is the main procedure. At the end of the procedure, the variable *Length.reso_zoom* which maximises the cavity circulating power, is returned. This script is divided into two similar parts: first part the cavity detuning is scanned over one wavelength, then in the second part we scan around the maximum position found after the first scan with a much greater detuning resolution (we zoom around the maxima found in the first part). At the beginning of the script, two important variables are defined: *Length.nb_iter* determines the number of step used to scan the cavity over one wavelength (usually 2000) and *Length.nb_propa_field* is the number of light round trip taken into account when calculating the circulating power (i.e. number of fields E_i we used).

- Propagate_Field.m is script called at the beginning of Find_resonance_length.m. This script calculates all the intermediate field E_i using the FFT code and stores the results in one variable called *Field.propa*. *Field.propa* is a 3D matrix having with a size of *Grid.Num_point* $\times$ *Grid.Num_point* $\times$ *Length.nb_propa_field*.

- Build_Field_Cavity.m is a function called intensively by Find_resonance_length.m. This function takes for argument a length detuning and returns the build-up circulating field following the equation 2.1.

- Calculate_power.m is a simple function which takes for input a 2D electric field and returns the optical power in Watt of the input field.

2.3.4 Some comments

The method used by OSCAR to find the resonance length is slow. In term of calculation time, it is the bottleneck of this FFT code. It is possible to find different approaches to calculate the resonance length and some are much faster, however I prefer to keep the method described above. Why ? because the plot of the circulating power as a function of the wavelength (figure 2.1) contains numerous essential information which help debugging the simulation or understanding the optical system. Here some examples:

- If during a simulation, the plot of the circulating power as a function of the detuning does not look like the one in figure 2.1, but instead looks flat or with very small bumps it means the cavity is unstable. In the same idea, the linewidth of the peak in the plot is inversely proportional to the finesse of the cavity so it is a good indication of the round trip loss in the cavity.

- The number of peaks in the plot is directly proportional to the mode mismatching or misalignment between the input beam and the cavity eigen modes. In the case of perfect mode matching only one peak is present, which means that all the input light coupled to only one optical mode (preferably the fundamental Gaussian beam). By looking at the shape of the higher order modes which are excited (see figure 2.1), we could have an idea of the type of mode mismatching and/or misalignment. For example, if all the higher order modes which are excited looks like TEM_{m0}, it means the input beam is misaligned with the cavity axis in the horizontal direction.

- Finally, since we also know the position detuning for the higher order modes, we can also set the cavity on the resonance of the higher order modes if necessary. The knowledge of the relative detuning position of the resonance for the higher order modes allows also the calculation of the Gouy phase shift between higher order modes (which maybe unknown if the beam is not Gaussian).

We do not need a lot of light round trip to have an accurate result for the resonance length position. In the previous example, we use 800 or 1600 round trips for the calculation but only 50 round trips can already give a correct answer. The only difference is that the plot of the circulating power as a function of the detuning may not look so sharp (so we can miss the presence of smaller resonance peaks).

For verification purpose, it may be important to check the shape of the circulating field for different detuning length. For example to display the circulating field for a detuning position of 2.5×10^{-7} which corresponds to the first left peak in the figure 2.1, we just have to write the following command

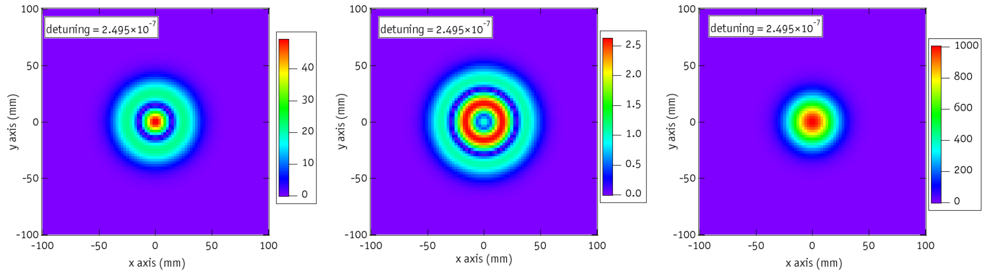


Figure 2.2: 2D amplitude of the circulating field responsible for the 3 peaks in the plot of the cavity circulating as a function of the detuning (figure 2.1). We can recognise from left to right the mode LG_{10}, LG_{20} and finally the fundamental mode LG_{00}. The presence of the there first LG_{m0} optical modes indicates that the input beam is properly aligned with the cavity axis but we have a slight mode mismatching.

in Matlab:

```
>> Plot_Field(Build_Field_Cavity(2.5E-7))
```

With such command, we can check which optical modes can build up in the cavity. The 2D amplitude of the three optical modes corresponding to the three peaks in figure 2.1 are presented in figure 2.2 in the order of increasing detuning.

2.4 Calculating the circulating field

The previous procedure Find_resonance_length.m is essential to determine the operating point for the cavity. In most cases the cavity is locked on the fundamental mode TEM_{00}, so the procedure Find_resonance_length.m returns by default the value of the detuning required to make the TEM_{00} resonant inside the cavity. It is always implicitly assumed that the maximum circulating power is obtained for a resonant TEM_{00} and not for a higher order optical mode.

After we have calculated the resonance length, the script Get_results.m can be called. This script calculates the static fields in the Fabry perot cavity and displayed the total circulating power as well as other results.

The procedure to calculate the circulating field was explained in the section 1.4 and is briefly reminded here. First the input laser field (variable *Field.Start* in OSCAR) crosses the input test mass substrate, creating the intermediate circulating field (variable *Field.Circ* and also called E_1 in figure 1.6). Then the intermediate circulating field is propagated back and forth between the cavity mirrors with the number of round trip determined by the variable *Iter.final*. The total circulating field (variable *Field.Total*) is derived by summing all the field intermediate field at a defined position in the cavity, in OSCAR it is done after the reflection from the input mirror. Everything described above can be simply achieved in Matlab as shown in the listing 2.1.

Listing 2.1: The core of the OSCAR programm to calculate the circulating field in a cavity

```
Length.reso_zoom = 8.7794200e-007;
Iter . final  = 3000;

Phase_shift  =  exp(i*Laser.k_prop* Length.reso_zoom);

Field.Circ = Propa_mirror(Field.Start, Mirror.ITM_trans,i*ITM.t);

for  q = 1:Iter . final

    Field . Total = Field.Total + Field.Circ;

    Field . Circ = Make_propagation(Field.Circ,Mat_propagation);
    Field . Circ = Propa_mirror(Field.Circ,Mirror.ITM_cav,ETM.r);
    Field . Circ = Make_propagation(Field.Circ,Mat_propagation);
    Field . Circ = Field.Circ * Phase_shift;
    Field . Circ = Propa_mirror(Field.Circ, Mirror.ITM_cav,ITM.r);

end
```

After the calculation of the total circulating field in the cavity *Field.Total*, the transmitted beam *Field.Transmit* and the reflected beam *Field.Reflect* can be easily derived. The transmitted beam is simply the total circulating field after a transmission through the end mirror. The reflected beam is the sum of the input field directly reflected by the input mirror and the field leaking from the cavity (which is the total circulating field transmitted by the input mirror).

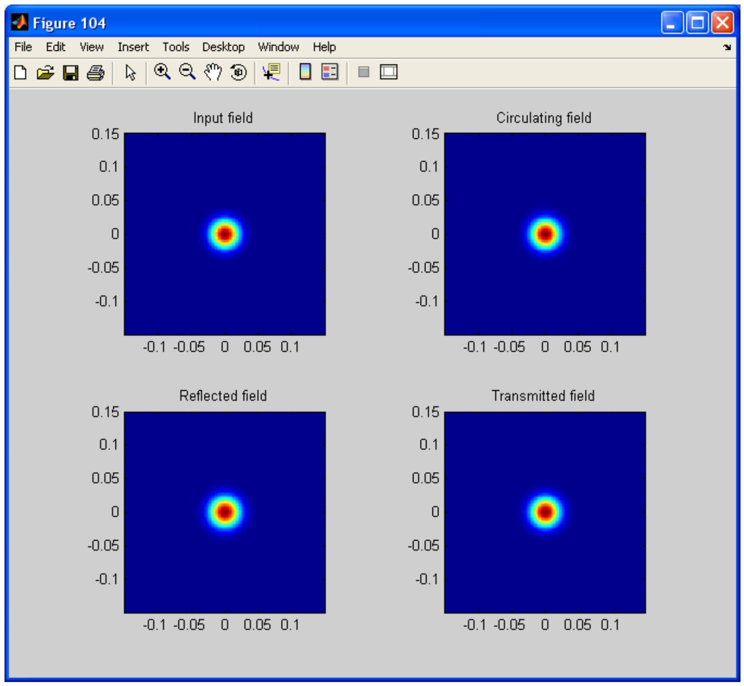


Figure 2.3: Amplitude profile of the different steady state optical fields in the Fabry Perot cavity.

2.5 Displaying the results

After the total circulating field, the transmitted field and the reflected field have been calculated, the results can be displayed. First some parameters (with obvious names) are written in the Matlab command window, then a 2D plot of the different optical fields present is displayed as shown in figure 2.3.

```
------ Display the results ------
Circulating power (W): 749.256602
Reflected power (W): 0.885891
Transmitted power (W): 0.037463
Beam radius on the ITM (m): 0.020574
Beam radius on the ETM (m): 0.020577
Cavity waist size (m): 0.016462
Location of the waist from ITM (m): -499.853070
```

Table 2.2: Influence of the total number of light round trip considered (variable *Iter.final*) on the results. The size of the grid is set to 128 × 128.

Number of iteration	Circulating power (W)	Cavity waist size (mm)	Cavity waist position from IM (m)
1000	640.712	18.403	-499.835
2500	747.546	18.403	-499.852
5000	749.903	18.403	-499.853
10000	749.906	18.403	-499.835
Finesse	749.906	18.403	-500

To feel confident in the results from OSCAR, it is always good to test the results with different parameters for the number of light round trip or the size of the grid. If the results are very sensitive to one parameters, it usually means that there is a problem somewhere, and further checks are required.

For example, the results as a function of number of light round trip are presented in the table 2.2. As expected on resonance, the circulating power, defined by the power of the sum of all the fields E_i increases when the number of field E_i computed increases. The right number of iteration to consider for a simulation depends on the finesse of the cavity. For a low finesse cavity, with a high round trip loss, the power contained in the field E_i will quickly decrease and rapidly becomes negligible after few round trips. In table 2.2, the FFT results are compared with the results from the software Finesse [10], which is based on mode expansion. The Finesse script used to simulate the Fabry Perot cavity is presented in appendix B.

Another important parameter to test is the resolution of the grid. The results for different sizes of the grid are presented in the table 2.3. As we can see the results are pretty robust even with a coarse grid. For a grid size of 32 × 32, the grid resolution is 1 cm, which means the beam radius is just represented by 2 pixels in the cavity. Even with such a low resolution, the results are still accurate.

2.6 Calculating the cavity eigen-mode and diffraction loss

We can first wonder what is called a cavity eigen-mode ? A cavity eigen mode is an electric field which comes back exactly with the same spatial profile after one cavity round trip. The spatial profile must be identical but not necessary

Table 2.3: Influence of the size of the grid (variable *Grid.Num_point*) on the results. The number of iteration is 5000. The computation time is normalised by the computation time required for a grid of size 128 × 128 (which is less than 2 minutes on a modern computer).

Size of the grid	Circulating power (W)	Cavity waist size (mm)	Cavity waist position from IM (m)	Normalised computation time
32 × 32	749.902	18.403	-499.854	0.09
64 × 64	749.903	18.403	-499.853	0.31
128 × 128	749.903	18.403	-499.853	1.00
256 × 256	749.902	18.403	-499.844	7.32
Finesse	749.906	18.403	-500	-

the amplitude since the cavity may be lossy. The usual cavity eigen modes are the set of Hermitte-Gauss and Laguere-Gauss for cavity with spherical mirrors, as most of the readers must already know.

For high finesse cavity[2], the circulating field is in fact a cavity eigen mode. This can be easily understood if the input cavity field is negligible in front of the cavity circulating field, a more rigorous demonstration can be found in [11] for example. Mode cleaner cavities are based on this principle, the transmitted beam is the TEM_{00} cavity eigen-mode, whereas the input beam may be composed of several optical modes.

Usually, we seldom need a FFT code to calculate the eigen modes of a Fabry Perot cavity with spherical mirrors since the exact analytical solutions are known [12]. However, we may want to know the cavity eigen modes if the mirrors are not perfectly spherical (because of thermal lensing) or if beam clipping due to finite size mirrors is important. To know the cavity eigen modes in this case, we defined an input beam close in shape to the supposed cavity eigen modes and then calculate the cavity circulating field. The cavity circulating field will be the cavity eigen mode. Some problems may arise if the cavity is nearly degenerated and the input beam is composed of several resonant or near resonant modes. In this case we can think of two solutions:

1. Increase the cavity finesse to have a better separation between optical modes and if necessary generate some specific losses to attenuate the undesirable optical modes.

2. Take the cavity circulating field as a new input field and start a new

[2]Quantitatively (and approximatively), we could say that a cavity has a high finesse when the the circulating power is much higher than the input power

Listing 2.2: Piece of code used to calculate the diffraction loss. We suppose that we have already calculate the circulating field *Field.Total*.

```
% Normalise the circulating field (=the eigen-nmode)
Field.loss = Field.Total;
Field.loss = Field.loss/sqrt(Calculate_power(Field.loss));

% Make a round trip with a reflectivity of 1 for the mirrors
Field.loss = Make_propagation(Field.loss,Mat_propagation);
Field.loss = Propa_mirror(Field.loss,Mirror.ETM_cav,1);
Field.loss = Make_propagation(Field.loss,Mat_propagation);
Field.loss = Propa_mirror(Field.loss, Mirror.ITM_cav,1);

% Calculate the diffraction loss
Dif_loss = (1 - Calculate_power(Field.loss));
fprintf('Diffraction loss per round trip: %d \n',Dif_loss);
```

calculation. This step can be done several times and could be understood as cascading optical cavities to increase the mode cleaning effect.

Since we have now an idea about how to calculate a cavity eigen mode, we can also try to calculate the diffraction loss for this mode. We called diffraction loss, the loss due to the finite size of the mirrors. This loss are also sometime referred as clipping loss. Since in theory, Gaussian beams have an infinite spatial extend, diffraction losses are always present. However the loss can become negligible for large diameter mirrors and small laser beam radius. For example, the diffraction loss of a laser beam of beam radius 6 cm after reflection on a 15 cm radius mirror is only 4 ppm [13]. The diffraction loss are more important for higher order optical modes and ultimately can even affect the profile of the mode [14].

In OSCAR the diffraction loss of an eigen mode is computed by calculating the round trip loss of the mode when the reflectivities of the cavity mirrors are set to 1. So the first step is to calculate the cavity eigen mode, this is usually done by calculating the circulating field in a high finesse cavity. The second step is to normalised the power of the eigen mode to 1 and then propagate the mode one round trip in the cavity whose mirrors reflectivities are set to 1. Finally, the diffraction loss is simply the power lost by the mode during one round trip. The above steps are shown in the listing 2.2.

An example about how to calculate diffraction loss with OSCAR can be found in the folder called Calculate_diffraction_loss. This example is detailed in the chapter 3.2 of this manual.

2.7 A typical OSCAR run

A typical run of OSCAR consists of running consecutively three different Matlab scripts:

1. CreateField.m, is the script used to initialised the variables and defined the optical cavity parameters, the mirror maps as well as the input beam.

2. Find_resonance_length.m is used to find automatically the resonance length of the cavity which maximise the circulating power. In most cases, the resonance length locks the cavity on the TEM_{00}. Manually it is possible to lock the cavity on any arbitrary position, for example to see the resonance of one higher order mode.

3. Get_results.m is the main procedure which get the circulating field in the cavity and if required also the reflected and transmitted fields and the diffraction loss of the circulating field.

The call for three previous script are usually reunited into one single scrip called Run_OSCAR.m.

2.8 Script and function list

Here the list of Matlab scripts that you may find in every OSCAR folders. Depending on the goal of the simulation, the scripts may not be exactly the same and some variations exist.

- Beam_parameter.m, a function which takes for parameter a complex 2D Gaussian field and returns the beam radius and the wavefront radius of curvature. The fit only works for fundamental Gaussian beam but it can easily be adapted to also fit higher order modes.

- Build_Field_Cavity.m is a function used to find the resonance length, see section 2.3.3.

- Calculate_power.m a simple function which takes a 2D field and returns the optical power of the field.

- CreateField.m, the first script called in OSCAR to initialise all the variables, see section 2.2.

- CreateMirror.m script to create the matrix representing the wavefront distortion induced by the mirrors, see section 1.3.2.

- Find_resonance_length.m script to find the microscopic detuning required to set the cavity on resonance, see section 2.3.2.

- Get_results.m main OSCAR procedure to calculate the circulating power in the cavity. The script is also used to do some post processing of the results, see section 2.4 and 2.5.

- Make_propagation.m propagates a 2D field over a certain distance. The function takes two arguments a field and a propagation matrix and return the field after propagation. The function is the implementation of the last three lines in listing 1.2.

- Plot_Field.m, a function which takes a 2D field and plots the amplitude of the field in 2D (or 3D if it is desired).

- Propa_mirror.m a function to simulate the effect of a wavefront distortion. The function takes a 2D input field, a mirror map or any distortion and a reflectivity (or transmission) and returns the field after reflection (or transmission). The function is described in section 1.3.5.

- Propagate_Field.m creates the initial 3D matrix used to calculate the circulating field for the different detuning, see section 2.3.3 for further explanations.

Chapter 3

Applications

In this chapter, we provide examples of some typical results which can be obtained with OSCAR. These examples could be taken as a starting point to build more complex simulations. The OSCAR scripts associated with each examples are provided in the OSCAR package.

3.1 Distortion of the optical field due to thermal lensing

Let's consider the same Fabry-Perot cavity as the one described in the previous chapter (section 2.1). Instead of using an input laser beam of 1 W, we upgrade the input power to 500 W (so the circulating power is now 375 kW) and we are interested to simulate some thermal lensing. Both input and end mirrors are supposed to be made of fused silica with a substrate absorption of 2 ppm/cm and a coating absorption of 0.5 ppm.

At least two distinctive effects can be induced due to the optical power absorbed in the mirrors:

- A temperature gradient inside the substrates of the mirrors appears. The temperature gradient generates a refractive index gradient (thermo-optic effect) which induces a wavefront distortion for the beam crossing the optics.

- Since the temperature is no longer uniform in the test mass, the curvature of the optic surface will change as a result of thermal expansion. In this case we can expect the eigen mode of the cavity to change as well.

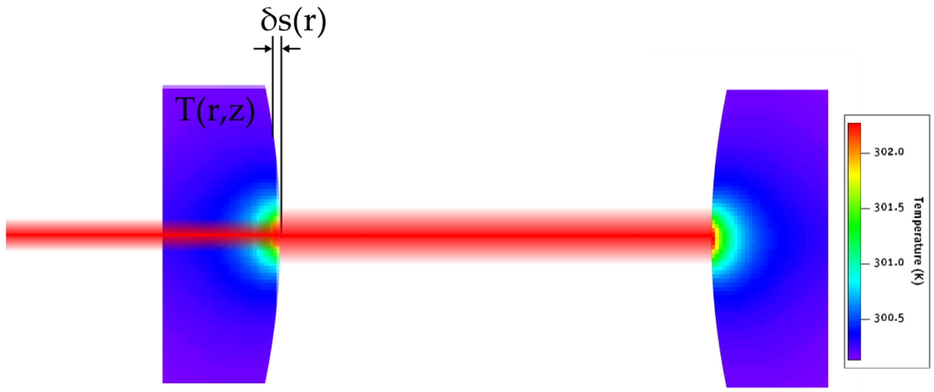


Figure 3.1: Schematics of the hot cavity. The mirror diameter is 250 mm and the thickness is 100 mm. We can notice that the temperature profile is dominated by the coating absorption. The optical parameters are detailed in the text. For clarity, we did not represent the curvature of the cold optics and we suppose the same distortion in the input and end mirrors. $T(r,z)$ represents the temperature distribution inside the test mass and $\delta s(r)$ the change in sagitta due to the optical absorption.

Before we continue, it is better to make some assumptions to lighten the calculations. If the reader understands the method described here it is straight forward to implement a full model. First we will suppose that we have uniform absorption in the test mass, so we can take advantage of the cylindrical symmetry. The power absorbed is dominated by the coating absorption, so we will only take into account the temperature gradient $T(r,z)$ in the optics and the thermal expansion of the mirror high reflective (HR) coating surface $\delta s(r)$. The deformations of the anti reflective coating are assumed to be negligible.

OSCAR can not simulate directly the effect of the optical absorption. If required an analytical formula can be implemented [15] however it is not as flexible as finite element simulations in particular if thermal lensing compensation scheme has to be investigated. I used the software ANSYS to simulate the temperature gradient $T(r,z)$ in the optics and the thermal expansion of the mirror surface $\delta s(r)$[1]. A schematic of the hot cavity is shown in figure 3.1.

From the temperature distribution inside the substrate and the sagitta

[1]We assume uniform absorption in the test mass, so we can take advantage of the cylindrical symmetry.

change, we can derive the optical path length difference $\Delta OPN_{sub}^{trans}(r)$ induced by the substrate in transmission according to:

$$\begin{aligned}\Delta OPN_{sub}^{trans}(r) &= \int_0^L n(r,z)dz - \int_0^L n(0,z)dz + (n-1)\delta s(r) \\ &= \int_0^L \beta T(r,z)dz - \int_0^L \beta T(0,z) + (n-1)\delta s(r)dz \quad (3.1)\end{aligned}$$

With β the thermo-optic coefficent and n the refractive index of the substrate. Similarly, we can calculate the wavefront distortion $\Delta OPN_{sub}^{ref}(r)$ for the input beam reflected directly reflected on the input mirror:

$$\Delta OPN_{sub}^{ref}(r) = 2\left(\int_0^L \beta T(r,z)dz - \int_0^L \beta T(0,z) + n\delta s(r)\right) \quad (3.2)$$

And finally, the wavefront distortion of the beam reflected on the mirrors inside the cavity is:

$$\Delta OPN_{cav}^{ref}(r) = 2\delta s(r) \quad (3.3)$$

Of course, the three wavefront distortions just defined, have to be added to any wavefront distortion already present when the cavity is cold, especially the ones induced by the curvature of the mirrors. For references, the main distortions due to thermal lensing which have to be included in OSCAR are plotted in figure 3.2.

Here an example how to proceed in reality. First run ANSYS to simulate the temperature distribution in the test mass as well as the thermal expansion of the optic. Then from these results, we can save a text file with two results, first the wavefront distortion induced by the thermo-optic effect for the optic in transmission and the change in sagitta of the optics. The text file is then loaded in OSCAR and the new wavefront distortions for the all optics of the cavity are calculated accordingly. An example of such a text file is presented below, the first column is the radius, the second the optical path length difference in transmission and the third the change in sagitta, all the columns are in meter.

```
-1.2500000e-001 -1.0824598e-006 -2.6695000e-008
-1.2142857e-001 -1.0731224e-006 -2.6433549e-008
-1.1785714e-001 -1.0655394e-006 -2.6172131e-008
```

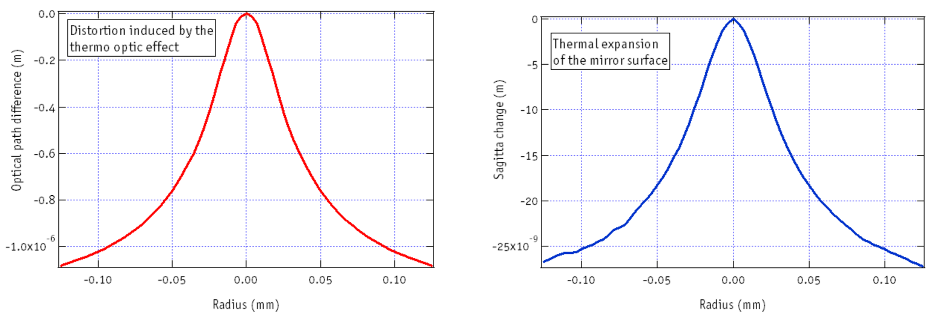


Figure 3.2: Optical path length difference induced by the mirror substrates in transmission when only the thermo optics effect is considered (left) and change in the sagitta of the reflective side of the mirrors (right). It interesting to note the difference between the two vertical scales, the left plot represents an optical path in micrometers, whereas the right plot is in tens of nanometers. These plots are derived from the results of ANSYS simulations and will then be integrated into OSCAR.

```
-1.1428571e-001 -1.0577238e-006 -2.5911061e-008
-1.1071429e-001 -1.0496365e-006 -2.5695000e-008
-1.0714286e-001 -1.0398661e-006 -2.5695000e-008
-1.0357143e-001 -1.0304229e-006 -2.5615955e-008
-1.0000000e-001 -1.0216618e-006 -2.5349271e-008
```

The file can found in the OSCAR distribution under the name From_ANSYS.txt in the folder Calculate_TL_effect. A graphical representation of the data file from ANSYS is shown in figure 3.2.

Most of the time, the resolution used with the results file from ANSYS is different from the grid resolution used in OSCAR so we have to resample the results using an interpolation method. How the results file is integrated into OSCAR is presented into the listing 3.1

After implementing the distorted mirrors, we can calculate the resonance length for the fundamental mode and then calculate the total circulating field. This is done by running successively the scripts Find_resonance_length.m and Get_results.m. The comparison between the cold cavity is presented in table 3.1.

Some very interesting points can be deduced by understanding the two lines of the table 3.1:

Listing 3.1: Commands used to read thermal lensing results from ANSYS.

```
% Load the results file
load('From_ANSYS.txt')
loaded.radius = From_ANSYS(:,1);
loaded.TL = interp1(From_ANSYS(:,1),From_ANSYS(:,2),Grid.D2,'spline')*0.2;
loaded.sag = interp1(From_ANSYS(:,1),From_ANSYS(:,3),Grid.D2,'spline')*0;

% Add the thermal lensing distortion to the previous wavefront
% Special attention to the sign!

Mirror.ITM_cav = Mirror.ITM_cav - 2*loaded.sag;
Mirror.ETM_cav = Mirror.ETM_cav - 2*loaded.sag;

Mirror.ETM_trans = Mirror.ETM_trans + loaded.TL - (Refrac_index-1)*loaded.sag;
Mirror.ITM_trans = Mirror.ITM_trans + loaded.TL - (Refrac_index-1)*loaded.sag;

Mirror.ITM_ref = Mirror.ITM_ref + 2*loaded.TL - 2*Refrac_index*loaded.sag;
```

Table 3.1: Comparison of the cavity gain and size of the beam on the input mirror for a cavity with and without thermal lensing.

	Cold cavity	Hot cavity
Cavity gain	749.3	444.8
Beam radius on IM (mm)	20.6	20.8

- Due to thermal lensing, the beam radius only increases by 1 %. That indicates that the mirror profiles are only slightly affected by thermal lensing since the cavity eigen modes are very similar for both cases: cold and hot cavities. This is not a surprise since the change in sagitta of the reflective sides of the mirrors is relatively small as we have previously seen.

- The decrease in the optical gain is quite important, since the optical gain is almost divided by a factor 2 between the cold and hot cases. It means in the hot cavity case, we have a strong mode mismatching between the input beam and the cavity fundamental mode. Since the latter has almost not changed, we can deduce that the mode mismatching is induced by the thermal lens in the substrate of the input mirror.

- We have assumed a certain amount of optical power absorbed in the mirror, it was based on the optical power circulating in the cold cavity. But since the circulating has decreased due to the mode mismatching, our thermal lens calculating wrong. A simple iterative process can be written to solve this problem and find the steady state parameters.

- This simple example shows that in our fused silica mirrors, the main thermal lensing effect is due to the thermal lens in the substrate of the input mirror generated by the optical absorption in the high reflective coating. The conclusion may be different for different substrates such as sapphire or calcium fluoride.

3.2 Calculating diffraction losses

One of the most promising application of FFT optical codes is to calculate the diffraction loss of the circulating cavity field. The method used to calculate the diffraction loss has been explained in section 2.6. An example how to calculate the diffraction loss of the mode HG_{10} is presented in the folder Calculate_diffraction_loss.

The diffraction loss calculation can be decomposed into three steps:

1. Find the resonance length of the cavity for the mode HG_{10}. For that we can inject a mode HG_{10} and maximise the circulating power. This approach is too easy, instead we will inject a uniform pattern of light and excite all the optical modes HG_{m0}. Then we will select manually the resonance length of the mode HG_{10} in the spectrum of the cavity.

2. Find the HG_{10} eigen mode of the cavity. The shape of this mode can be different from the theory if the mode undergoes some serious clipping.

3. Calculate the diffraction loss as the power lost during one light round trip with perfect reflective mirror.

We decide to not inject directly a mode HG_{10} (right plot on figure 3.3) but instead inject a simpler light pattern which will excite all the optical modes along the horizontal axis. This pattern is one vertical strip with positive amplitude and one vertical strip with negative amplitude as shown in the left plot figure 3.3.

By using the procedure Find_resonance_length.m, we can calculate all the cavity modes excited due to our particular input beam. The circulating power as the cavity is scanned over one FSR is shown in figure 3.4. Each peak in this plot represents the resonance of one of the cavity mode. We can display

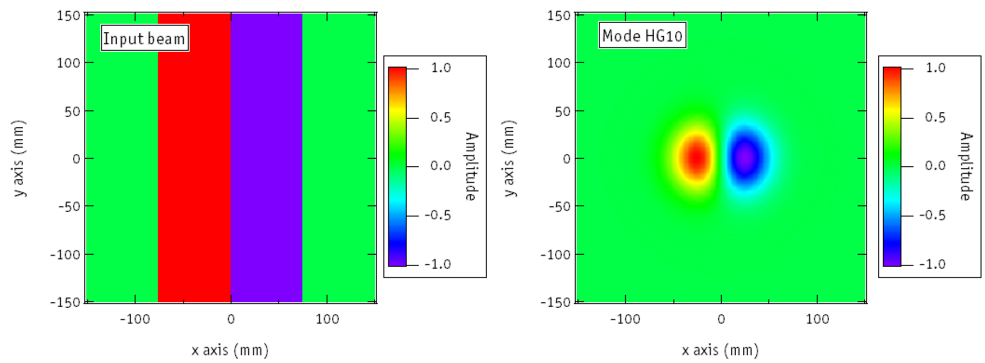


Figure 3.3: In the left plot, is the profile of the input electric field that we inject in the cavity to excite the HG_{10} shown on the right. Both optical pattern are normalised in amplitude.

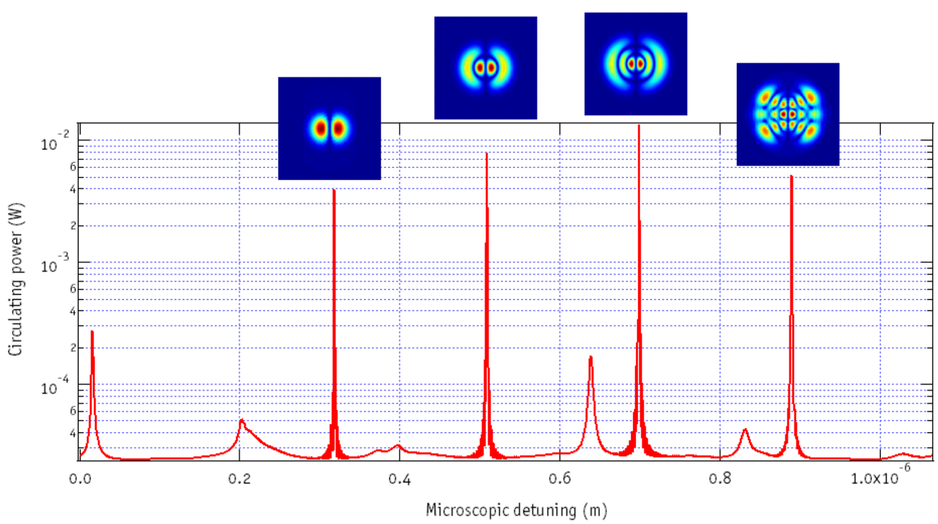


Figure 3.4: Cavity spectrum showing the different resonance lengths of the optical modes. The power profile of the main optical modes is presented above their respective resonance peaks.

Table 3.2: Influence of the size of the grid on the diffraction loss).

Size of the grid	Diffraction loss (ppm)
128 × 128	618
256 × 256	620
512 × 512	621

different cavity eigen modes for the first four highest peaks and we found that the mode HG_{10} resonates for a detuning of 3.181×10^{-7} m.

Since we have found the resonance length for the mode HG_{10}, we can now use the procedure Get_results.m to calculate the diffraction loss using the method described in section 2.6. As the result we found the diffraction loss equal to 618 ppm per round trip. Important beam clipping can also change the shape of the eigen mode of the cavity, this can be easily demonstrated using a FFT code[14].

We can now check how the diffraction loss depends of the size of the grid. The results are shown in 3.2. As we can see the results do not change as we increase the size of the grid (which is a good sign). Below a grid size of 128 × 128, no realistic eigen modes can be found in the cavity.

3.3 Using flat beams

In this section, we will show how OSCAR can be used to calculate the frequency separation between higher order modes. This is in fact an indirect calculation of the Gouy phase shift between optical modes. To make the example more interesting, we will not use the usual Gaussian beams but flat beams. As a consequence of using flat beams, the mirrors of our cavity will no longer have spherical profiles [16].

To create the mirror profiles, we will not use directly the analytical formula which is relatively complicated, but instead we will use the wavefront curvature of the flat beam at the mirror position. Since the flat beam is an eigen mode of the cavity, the curvature of the mirrors must match the wavefront of the incoming beam. So the three steps to calculate will be:

1. Use the simple analytical formula to define the flat beam at the waist of the cavity. Since both mirror will have identical profile, we know that the waist position is in the middle of the cavity.

Listing 3.2: Script to create the mirror profiles used to support a given light field

```
%----------------------- Create nearly concentric flat beam --------------------
% Create the profile at the cavity waist

waist_0 = sqrt((Laser.lambda * Length_cav) / (2*pi));
p = 3*waist_0; % Defined the width of the Mesa beam

x_temp = Grid.D2/waist_0;
Field.mesa_tmp = (1./x_temp).*exp(-x_temp.^2).*besselj(1,2*x_temp*p/waist_0);

clear('x_temp')

% Propagate the beam from the waist to the mirror
Field.On_mirror = Make_propagation(Field.mesa_tmp,Mat_propagation_h);

% Take the phase of the beam at the mirror and calculate the equivalent
% change in sagitta
Mirror.ETM_cav = (2/Laser.k_prop)*angle(Field.On_mirror);
Mirror.ITM_cav = Mirror.ETM_cav;
```

2. Propagate the beam along half the cavity length, so the flat beam is now at the mirror position.

3. Calculate the wavefront curvature of the beam at the mirror and then set the mirror profile to be identical to the wavefront.

The method described in the three points above can be directly implemented in OSCAR as shown in the listing 3.2.

Since we are just interested in the frequency difference between 2 eigen modes of the cavity (and not in the cavity circulating power), the input laser beam only needs to be slightly matched to the cavity eigen modes (and we must have a high finesse to achieve a good mode selection). The important point is that the input beam couples (or excites) at least to the two cavity eigen modes of interest.

As for the input light field, we decided to use a classic fundamental Gaussian beam. By this way we will couple to most the cavity eigen modes which are circularly symmetric, e.g. the equivalent of the LG_{m0} modes. For example the spectrum of the cavity as we scan over one wavelength is shown in figure 3.5. As expected we managed to excite the fundamental mode and also some other higher order modes.

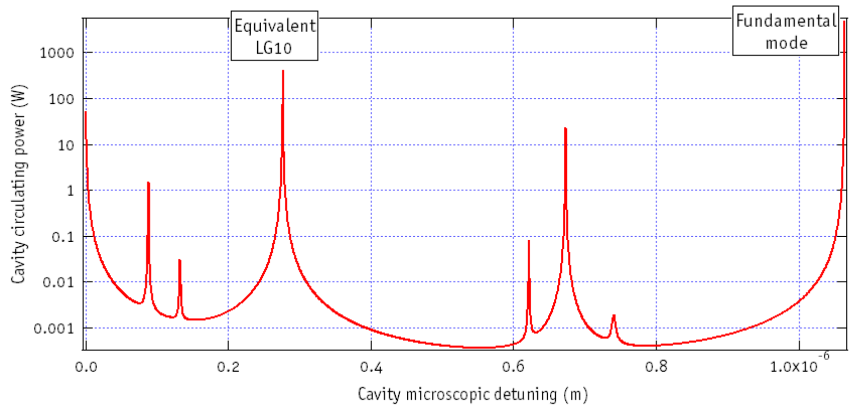


Figure 3.5: Cavity spectrum of the cavity supporting flat beams. It is important to note the fundamental for a detuning of 0. This is not a coincidence, but a direct consequence of the way we defined the mirror profiles (by propagating a beam from the cavity waist).

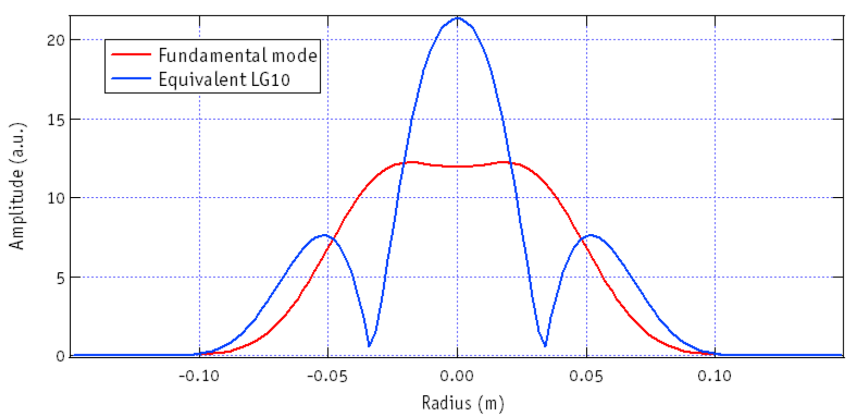


Figure 3.6: Comparison of the profile of the fundamental flat beam with the equivalent LG_{10}. Both beam contains the same optical power.

Just to have a look, we can display a cross section of the fundamental flat beams and the equivalent LG_{10} to compare the energy distribution. Such a comparison is shown in figure 3.6.

So we can go back to our initial question what is the frequency separation between the fundamental mesa beam and the mode LG_{10} ? From the spectrum plot, we know the difference of detuning in length between the fundamental mode and the equivalent LG_{10} is $\Delta l = 2.7664 \times 10^{-7}$ m. Since the displacement of one wavelength λ is equivalent to one free spectral range $(c/2L)$, the frequency difference Δf between the 2 modes is simply:

$$\begin{aligned}\Delta f &= (c/2L) * (\Delta l/\lambda) \\ &= (3 \times 10^8/4000) * (2.7664 \times 10^{-7}/1.064 \times 10^{-6}) \\ &= 19.5\text{kHz}\end{aligned} \tag{3.4}$$

Using a similar calculation, we could also deduce the Gouy phase shift between the two cavity eigen modes.

3.4 Deriving a Pound Drever Hall locking signals

In this example, we will show how to implement sidebands in OSCAR. For testing purpose, we can try to derive the Pound Drever Hall (PDH) error signal which is used to lock a Fabry Perot cavity on resonance [17]. The error signal is derived in reflection by combining a resonant (or near resonant) carrier field with a non resonant pair of sidebands. In this case, it can be shown that the error signal is in fact proportional to the imaginary part of the reflected field [18]. The script for this example can be found in the folder Calculate_PDH_signals

So the first thing to do, is to create sidebands using a phase modulator. Consider an electric field of amplitude E_0 and angular frequency ω_0 incoming on a phase modulator. E_m the electric field after the modulator can be written as:

$$\begin{aligned}E_m &= E_0 \exp(i\omega_0 t + m \sin(\omega_m t)) \\ &= E_0 \exp(i\omega_0 t) \sum_{k=-\infty}^{+\infty} (-1)^k J_k(m) \exp(ik\omega_m t)\end{aligned} \tag{3.5}$$

With m the modulating index and ω_m the modulation frequency. $J_k(m)$ is the bessel function of the first kind of order k. For example, the values for the first Bessel functions are shown in table 3.3. As a good approximation we can just consider the first order sidebands $k = \pm 1$. So equation 3.5 can be simplified to:

Table 3.3: Values of the first Bessel functions for different modulating indexes m. Conveniently, the Bessel functions follow the property $J_{-k}(m) = (-1)^k J_k(m)$

m	0.1	0.2	0.4	Taylor series
J_0(m)	0.9975	0.9900	0.9604	$1 - \frac{m^2}{4} + \mathcal{O}(x^4)$
J_1(m)	0.0499	0.0995	0.1960	$\frac{m}{2} + \mathcal{O}(x^3)$
J_2(m)	0.0012	0.0050	0.0197	$\frac{m^2}{8} + \mathcal{O}(x^4)$
J_3(m)	0.0000	0.0002	0.0013	$\frac{m^3}{48} + \mathcal{O}(x^5)$

$$E_m = E_0 \exp(i\omega_0 t)(1 + \frac{m}{2}\exp(i\omega_m t) - \frac{m}{2}\exp(-i\omega_m t)) \tag{3.6}$$

So the electric field after the phase modulator can be expanded as the superposition of three electric fields of angular frequency ω_0 (the carrier) and $\omega_0 + \omega_m$ and $\omega_0 - \omega_m$ (respectively the upper and lower sidebands). Since each field has different frequencies, they will also have different resonant conditions in the cavity.

Consider the sideband with the frequency $\omega_0 + \omega_m$. Propagating over a distance L the sideband undergoes the phase shift $\exp(i\frac{\omega_0+\omega_m}{c}L)$. That means, the sideband phase shift is the phase shift of the carrier plus an additional phase shift. Since before entering the cavity, the carrier and the sidebands have the same spatial profile, we can simulate the sidebands in the cavity by taking the simulated carrier and by adding an extra phase shift.

Practically, to simulate the carrier and the sidebands, we can simulate independently three optical fields, the carrier and the lower and upper sidebands. The three fields can be treated the same way in the simulation, except that they have different round trip phase shift. Below is the three phase shifts for one round trip in the cavity:

$$\begin{aligned}
&\exp(ikLength.reso_zoom) && \text{for the carrier} \\
&\exp(ikLength.reso_zoom) \times \exp(i\frac{\omega_m}{c}2Length_cav) && \text{for the upper sidebands} \\
&\exp(ikLength.reso_zoom) \times \exp(-i\frac{\omega_m}{c}2Length_cav) && \text{for the lower sidebands}
\end{aligned} \tag{3.7}$$

As a reminder *Length_cav* is the length of the cavity and *Length.reso_zoom* is the microscopic tuning used to set the resonance condition for the carrier.

Since we know how to calculate the carrier and the sidebands fields circulating in the cavity, we can also calculate the reflected fields from the cavity. The reflected light is also the superposition of three fields, the reflected carrier, the reflected lower sidebands and the reflected upper sidebands with respective amplitude E_0^{ref}, E_-^{ref} and E_+^{ref}. The power P_{det} detected by photodiode placed in reflection will then be:

$$
\begin{aligned}
P_{det} &= |E_0^{ref} + E_-^{ref} e^{-i\omega_m t} + E_+^{ref} e^{i\omega_m t}|^2 \\
&= |E_0^{ref}|^2 + |E_-^{ref}|^2 + |E_+^{ref}|^2 \\
&\quad + \cos(\omega_m t)\Re(E_0^{ref} E_-^{ref*} + E_0^{ref*} E_+^{ref}) \\
&\quad + \sin(\omega_m t)\Im(E_0^{ref} E_-^{ref*} + E_0^{ref*} E_+^{ref}) \\
&\quad + 2\Re(E_+^{ref*} E_-^{ref} e^{-i2\omega_m t})
\end{aligned}
\tag{3.8}
$$

With $\Re$ and $\Im$ the real and imaginary part and $*$ designates the complex conjugate. The PDH error signal e_{PDH} is derived by demodulating in phase P_{det} and the resulting signal is then low-pass filtered. As a result e_{PDH} can simply be written:

$$
e_{PDH} = \Im(E_0^{ref} E_-^{ref*} + E_0^{ref*} E_+^{ref}) \tag{3.9}
$$

The Matlab implementation of the method described above is straight forward. We first defined three electric fields: the carrier and then the lower and upper sidebands. Then we propagate each field in the cavity, calculate the reflected field and then plot the error signal using the formula 3.9. To make the result more pertinent, we fact scan the cavity over one free spectral range, so we can see the evolution of the error signal across the resonance.

To define the sidebands frequency and amplitude we have to define new variables (with self-explaining names) as shown in the script 3.3.

Then we can calculate the three circulating fields in the cavity called *Field.Total Field.Total_sidebands_plus* and *Field.Total_sidebands_minus* using the usual method introduced in section 2.4. The only difference in the algorithm between the propagation of the carrier and the sidebands is the addition of an extra phase shift per round trip for the sidebands as shown in equation 3.7.

Listing 3.3: New variables to define the sidebands frequency and amplitude

```
% Frequency of the sidebands in Hertz and modulation index
Sidebands.frequency = 10E6;
Sidebands.modulation = 0.4;

% New carrier amplitude
Laser.amplitude = Laser.amplitude * besselj(0,Sidebands.modulation);
% Amplitude for one of the 2 sidebands
Sidebands.amplitude = besselj(1,Sidebands.modulation);
Ration_amp = Sidebands.amplitude/Laser.amplitude;

% To add the additional phase shift undergone by the sidebands during the propagation
Sidebands.k_prop = 2*pi*Sidebands.frequency/3E8;
```

Listing 3.4: Calculating the PDH error signals from the fields reflected by the cavity

```
Field.PDH = imag(Field.Reflect_carrier.*conj(Field.Reflect_sidebands_plus)+...
            conj(Field.Reflect_carrier).*Field.Reflect_sidebands_minus);
% Normalised PDH signal
Power.PDH(r) = sum(sum(Field.PDH))/(Laser.amplitude*Sidebands.amplitude);
```

After we calculate the three reflected fields from the cavity named *Field.Reflect_carrier* *Field.Reflect_sidebands_plus* and *Field.Reflect_sidebands_minus*, we can derive the PDH error signal using the formula shown by equation 3.9, the equivalent Matlab listing is shown in 3.4.

The plot of the power circulating in the cavity as a function of the microscopic detuning is presented in figure 3.7. The central peak is due to the resonance of the carrier and the two smaller peaks indicates the resonance position for the sidebands. Finally the usual PDH error signal as calculated by OSCAR is shown in figure 3.8. Around the resonance of the carrier, we have an error signal to lock the cavity. In theory, we have also an error signal to lock the cavity on the sidebands resonance (less useful).

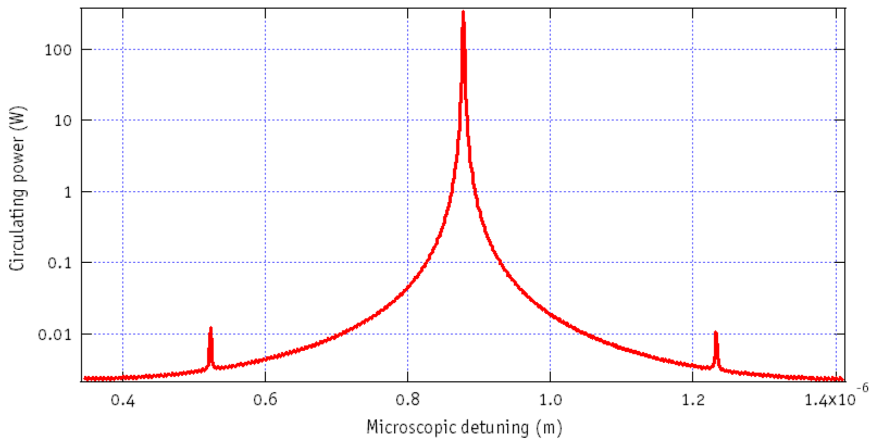


Figure 3.7: Circulating power inside the cavity as the cavity is scanned over one FSR. The two small peaks around the main resonance indicate the resonance of the sidebands.

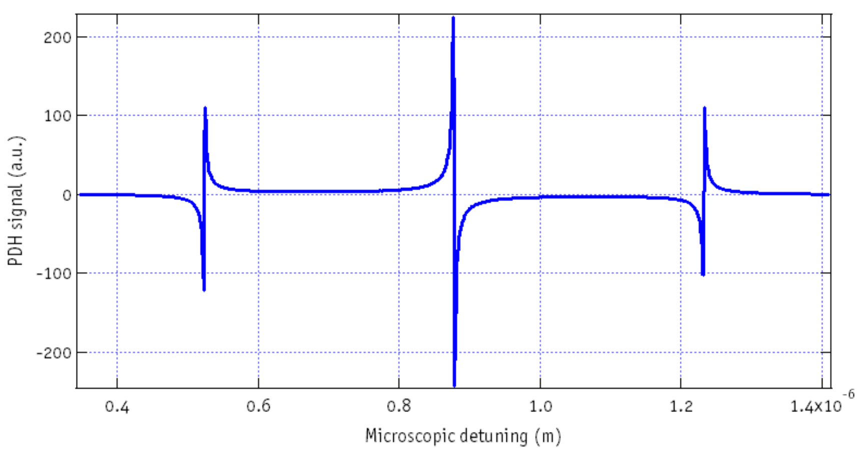


Figure 3.8: Pound Drever Hall error signal as a function of the detuning.

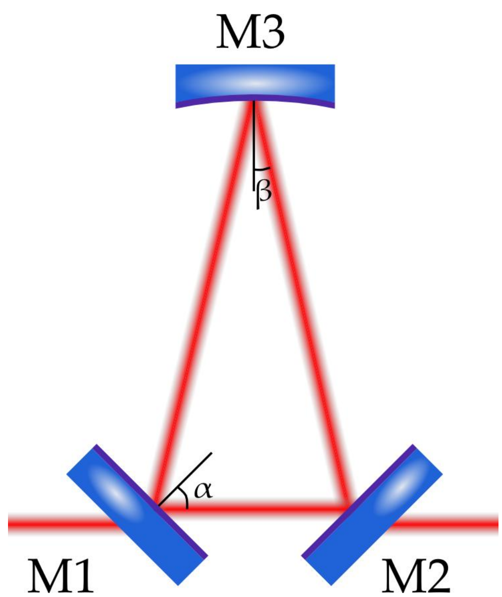


Figure 3.9: Optical setup of a three mirrors ring cavity. Unlike a Fabry-Perot cavity, where the mirrors are usually perpendicular to the incident beam, for the mode cleaner the incoming beam arrives with a large angle on the two bottom mirrors. The mirror notation used in the OSCAR example are also indicated.

3.5 Simulation of a three mirrors ring cavity

We can try now to add an additional mirror to our simulations and simulate a three mirrors ring cavity, often used as a mode cleaner. In the previous example we have been dealing with an optical beam perpendicular to the optical surface, this is no longer the case as seen in figure 3.9. The fact that the incident beam arrives with an angle to the surface is the main difficultly for this kind of simulation. However as we will see this problem can easily be overcome.

In the OSCAR mode cleaner example (folder Simulate_mode_cleaner), we suppose that the incident plane is horizontal. We keep this convention in the manual and it is relatively easy to adapt the code for any arbitrary angle of incidence. Let's examine what is happening on reflection: as soon as the angle of incidence of an electric field to a reflective surface is different from zero, two effects can arise:

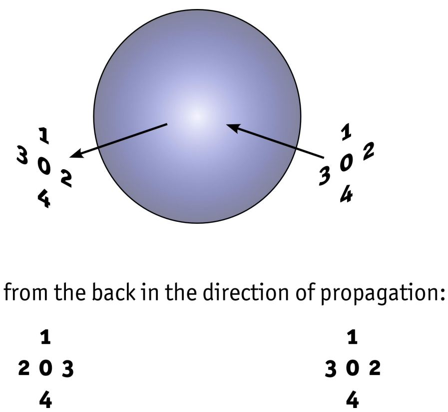


Figure 3.10: Example of an image reflected by a mirror. It is interesting to see that after reflection the image is flipped in the horizontal direction.

- The reflected beam will present astigmatism directly related to the angle of incidence. That can be understood by imagining that the incident beam will be stretched in the horizontal direction as it is projected to the mirror surface upon reflection.

- The beam on reflection will get flipped in the horizontal direction, this is purely a geometrical effect and the reader can easily be convinced by looking at the figure 3.10.

The two items described above are implemented in OSCAR in the new function *Propa_mirror* showed in the listing 3.5. Compared to the function *Propa_mirror* seen in the first chapter (section 1.3.5), we have added one new input parameter: *Grid_angle_X*. *Grid_angle_X* is a new distorted 2D grid representing the projection of the grid used for propagation (called *Grid.D2*) to the mirror surface.

The function used to simulate the reflection on a mirror can be decomposed in the following steps:

1. Project the incident beam on the mirror surface. So the incident beam will look astigmatic.

2. Reflect the projected incident beam in the same way as a incoming beam normal to the mirror surface

3. Project the reflected beam to the on the plane of propagation

Listing 3.5: Reflection on a mirror for arbitrary angle of incidence

```
% Stretch the laser beam as seen by the mirror
Output = Propa_mirror(Wave_field, Wave_mirror, reflec,Grid_angle_X)

Output = interp2(Grid_angle_X,Grid.Y,Wave_field,Grid.X,Grid.Y,'cubic');

Output = Output .* exp(-i * Wave_mirror*Laser.k_prop) * reflec .* Mirror.mask;


% Go back to the normal grid
Output = interp2(Grid.X,Grid.Y,Output,Grid_angle_X,Grid.Y,'cubic',0);

% Flip the matrix along the x axis
Output = Output(:,Grid.Num_point:-1:1);
```

4. Flip the beam in the horizontal plane

The steps described above are directly implemented in Matlab as shown in the script 3.5.

In the examples provided with OSCAR, the transmission of a 3 mirrors ring cavity is presented. To show the mode cleaning effect, the input beam is the normalised sum of the TEM_{10} and TEM_{01}. As expected both modes have different resonance condition inside the mode cleaner as shown in figure 3.11. The microscopic resonance length of the two modes is separated by half a wavelength, which is the expected result since the mode TEM_{10} encountered an extra 3π shift during its round trip propagation.

For this example, we set the working point of the cavity for the mode TEM_{10} to be resonant inside the cavity. The transmitted field from the mode cleaner is shown at the bottom right corner of figure 3.12. As we would hope the transmitted field is a pure TEM_{10}. For consistency, we also check that the reflected field is very similar to a pure TEM_{01}, but not exactly since the input beam was not perfectly matched to the cavity.

Below is the output from OSCAR:

```
------ Display the results ------
Circulating power (W): 49.524735
Reflected power (W): 0.503506
Transmitted power (W): 0.495243
```

So the mode cleaner was able to separate the mode TEM_{10} and TEM_{01}. We have slightly more power reflected than transmitted by the cavity, this is

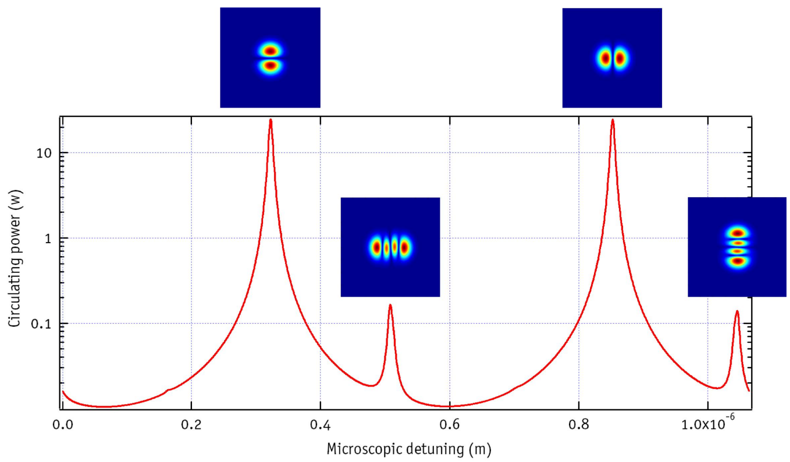


Figure 3.11: Scan of the mode cleaner over one free spectral range. The two main peaks are due to the resonance of the TEM_{01} and TEM_{10}. Additional peaks in the spectrum indicates a slight modemismatching.

a direct consequence of the fact that the input beam is not perfectly matched to the cavity.

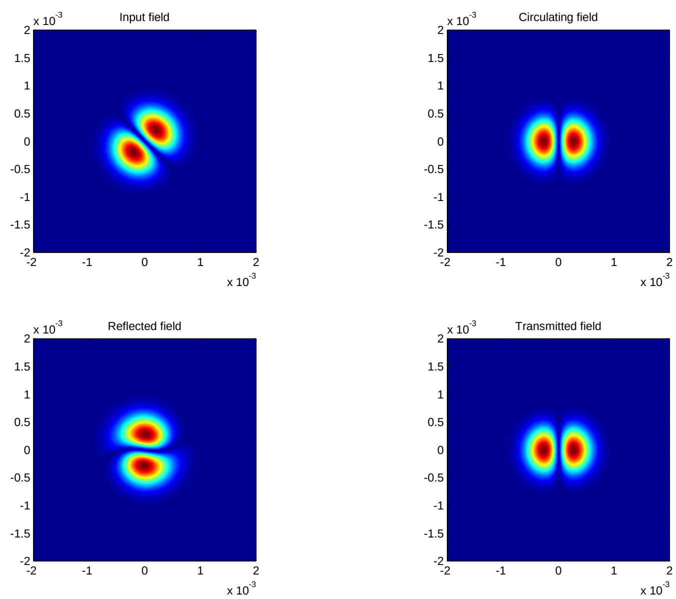


Figure 3.12: Output screen from OSCAR. The input field, a sum of TEM_{10} and TEM_{01} is shown on the top left corner. The reflected and the transmitted field are shown respectively in the bottom left and right picture.

Bibliography

[1] V. K. Madisetti and D. Williams, *The Digital Signal Processing Handbook*. CRC Press, 1998.

[2] http://en.wikipedia.org/wiki/Fourier_transform. A compact reminder for the main properties of the Fourier transform.

[3] M. Seul, L. O'Gorman, and M. J. Sammon, *Practical Algorithms for Image Analysis: Description, Examples, and Code*. Cambridge University Press, 2000. Some extensive illustrations of the results of the 2D Fourier transform on some simple patterns.

[4] H. Kogelnik and T. Li, "Laser beams and resonators," *Appl. Opt.*, vol. 5, pp. 1550–1567, 1966. A classic article at the foundation of many others, absolutely worth reading.

[5] W. T. Silfvast, *Laser Fundamentals*. Cambridge University Press, 2004. The section 11.2 of this book is a good reminder how to calculate the circulating field in an iterative way.

[6] B. Bochner, *Modelling the Performance of Interferometric Gravitational-Wave Detectors with Realistically Imperfect Optics*. PhD thesis, MIT, 1991.

[7] A. Tridgell, D. E. McClelland, and C. M. Savage, "Numerically modelling a dual recycling interferometric gravitational wave detector," in *Gravitational Astronomy: Instrument Design and Astrophysical Prospects* (W. S. Publishing, ed.), (Singapore), 1991.

[8] J. Agresti, *Researches on Non-standard Optics for Advanced Gravitational Waves Interferometers*. PhD thesis, University of Pisa, 2007.

[9] A. G. Fox and T. Li, "Computation of optical resonator modes by the method of resonance excitation," *IEEE j. quantum electron*, vol. 4, pp. 460–465, 1968.

[10] http://www.rzg.mpg.de/~adf/. Webpage of the package Finesse.

[11] P. Saha, "Fast estimation of transverse fields in high-finesse optical cavities," *J. Opt. Soc.Am. A*, vol. 14, no. 9, pp. 2195–2202, 1997.

[12] W. Koechner, *Solid-State Laser Engineering.* Springer Series in Optical Sciences, Springer, 2006.

[13] A. E. Siegman, *Lasers.* University Science Books, 1986. THE laser Bible, does it still need an introduction ?

[14] P. Barriga, B. Bhawal, L. Ju, and D. G. Blair, "Numerical calculations of diffraction losses in advanced interferometric gravitational wave detectors," *J. Opt. Soc.Am. A*, vol. 24, pp. 1731–1741, 2007.

[15] P. Hello and J. Y. Vinet, "Analytical models of thermal aberrations in massive mirrors heated by high power laser beams," *J. Phys. (France)*, vol. 51, pp. 1267–1282, 1990.

[16] M. Bondarescu1 and K. S. Thorne, "New family of light beams and mirror shapes for future ligo interferometers," *Phys. Rev. D*, vol. 74, p. 082003, 2006.

[17] R. W. P. Drever, J. L. Hall, F. V. Kowalski, J. Hough, G. M. Ford, A. J. Munley, and H. Ward, "Laser phase and frequency stabilization using an optical resonator," *Applied Physics B: Lasers and Optics*, vol. 31, pp. 97–105, 1983.

[18] E. D. Black, "An introduction to Pound–Drever–Hall laser frequency stabilization," *American Journal of Physics*, vol. 69, pp. 79–87, 2001.

Appendix A

Analytical formulation of the Gaussian beam propagation using Fourier transform

In this appendix, we will see how it is possible to rediscover all the formulas related to the laser beam propagation just by using the Fourier technique explained in chapter 1. Basically, we will do by hand the three steps which are numerically done in Matlab to propagate a Gaussian beam: first, 2D Fourier transform of the laser beam, then add the phase shift in the frequency domain and finally, inverse Fourier transform.

To facilitate the mathematical calculation, we will extensively use the following formula:

$$\int_{\infty}^{+\infty} \exp\left(-ax^2 + ibx\right) dx = \sqrt{\frac{\pi}{a}} \exp\left(\frac{-b^2}{4a}\right) \tag{A.1}$$

Let's consider a Gaussian beam $E(x, y, z)$ of amplitude A. For simplification we suppose the waist w_0 of the beam to be located at $z = 0$. The beam can be written:

$$E(x, y, 0) = A \exp\left(-\frac{x^2 + y^2}{w_0^2}\right) \tag{A.2}$$

We will now propagate the Gaussian beam along the z axis over a distance d. To not complicate the calculation, we will just consider one transverse dimension at first, for example the dimension along x since all the following equations can be decoupled in the x/y dimension.

So the first step is to calculate the Fourier transform $\widetilde{E}(\nu_x, 0)$ of the field $E(x, 0)$:

$$\widetilde{E}(\nu_x, 0) = \int_{\infty}^{+\infty} A \exp\left(-\frac{x^2}{w_0^2}\right) \exp(j2\pi\nu_x) dx \tag{A.3}$$

Using the formula A.1, the above equation becomes:

$$\widetilde{E}(\nu_x, 0) = A\sqrt{\pi w_0^2} \exp\left(-(\pi\nu_x w_0)^2\right) \tag{A.4}$$

To simulate the propagation the electric field over a distance d, the proper phase shift is added to Fourrier transform:

$$\widetilde{E}(\nu_x, d) = A\sqrt{\pi w_0^2} \exp\left(-(\pi\nu_x w_0)^2\right) \exp(-jkd + j\lambda\pi\nu_x^2 d) \tag{A.5}$$

Finally, we take the inverse Fourier transform of the equation A.5 to go back to the familiar x space coordinate system:

$$\begin{aligned}
E(x, d) &= \int_{\infty}^{+\infty} A\sqrt{\pi w_0^2} \exp\left(-(\pi\nu_x w_0)^2\right) \exp(-jkd + j\lambda\pi\nu_x^2 d) \exp(-j2\pi\nu_x x) d\nu_x \\
&= A\sqrt{\pi w_0^2} \exp(-jkd) \int_{\infty}^{+\infty} \exp\left(-(\pi^2 w_0^2 - j\lambda\pi d)\nu_x^2 - j2\pi x\nu_x\right) d\nu_x \\
&= A\sqrt{\pi w_0^2} \sqrt{\frac{\pi}{\pi^2 w_0^2 - j\pi\lambda d}} \exp(-jkd) \exp\left(\frac{-4\pi^2 x^2}{4(\pi^2 w_0^2 - j\lambda\pi d)}\right) \\
&= A\sqrt{\frac{1}{1 - j\frac{\lambda d}{\pi w_0^2}}} \exp(-jkd) \exp\left(-\frac{x^2}{w_0^2\left(1 - j\frac{\lambda d}{\pi w_0^2}\right)}\right)
\end{aligned} \tag{A.6}$$

We can now define the usual Raleigh range z_r as:

$$z_r = \frac{\pi w_0^2}{\lambda} \tag{A.7}$$

By inserting the Raleigh range z_r into A.6 and by adding the similar result from the other y transverse dimension, we obtain:

$$E(x, y, d) = A\frac{1}{1 - j\frac{d}{z_r}} \exp(-jkd) \exp\left(-\frac{x^2 + y^2}{w_0^2\left(\frac{1}{1-j\frac{d}{z_r}}\right)}\right) \tag{A.8}$$

Let's analyze now the two main constituent of equation A.8, the exponential and the complex factor in front of it. The complex number in the exponential can be separated between its real and imaginary parts:

$$
\begin{aligned}
\exp\left(-\frac{x^2+y^2}{w_0^2\left(\frac{1}{1-j\frac{d}{z_r}}\right)}\right) &= \exp\left(-\frac{x^2+y^2}{w_0^2\left(1+\frac{d^2}{z_r^2}\right)} - j\frac{(x^2+y^2)\frac{d}{z_r}}{w_0^2\left(1+\frac{d^2}{z_r^2}\right)}\right) \\
&= \exp\left(-\frac{x^2+y^2}{w_0^2\left(1+\frac{d^2}{z_r^2}\right)} - j\frac{(x^2+y^2)}{w_0^2\left(\frac{z_r}{d}+\frac{d}{z_r}\right)}\right) \\
&= \exp\left(-\frac{x^2+y^2}{w_0^2\left(1+\frac{d^2}{z_r^2}\right)} - j\frac{(x^2+y^2)}{\frac{2z_r}{k}\left(\frac{z_r}{d}+\frac{d}{z_r}\right)}\right) \\
&= \exp\left(-\frac{x^2+y^2}{w_0^2\left(1+\frac{d^2}{z_r^2}\right)} - jk\frac{(x^2+y^2)}{2d\left(1+\frac{z_r^2}{d^2}\right)}\right)
\end{aligned}
\tag{A.9}
$$

We can now consider the factor in front equation A.8:

$$
A\frac{1}{1-j\frac{d}{z_r}} = A\left(\frac{1}{1+\frac{d^2}{z_r^2}}\right)^{\frac{1}{2}} \exp\left(j\arctan\left(\frac{d}{z_r}\right)\right) \tag{A.10}
$$

Combining equations A.9 and A.10, we can deduce the familiar equations governing the propagation of Gaussian beams:

$$
E(x,y,d) = A\frac{w_0}{w(d)}\exp\left(-\frac{x^2+y^2}{w(d)^2}\right)\exp\left(-jk\left(d+\frac{x^2+y^2}{2R(d)}\right)+j\arctan\left(\frac{d}{z_r}\right)\right) \tag{A.11}
$$

With:

$$
\begin{aligned}
w(d) &= w_0\left(1+\frac{d^2}{z_r^2}\right) \\
R(d) &= d\left(1+\frac{z_r^2}{d^2}\right)
\end{aligned}
\tag{A.12}
$$

I found remarkable that by using the simple FFT steps to propagate the beam we can rediscover all the usual equations: the normalisation factor, the evolution of the beam radius and the wavefront radius of curvature as well as the Gouy phase shift. The courageous readers can also do the calculations presented here for the higher order modes in the Hermite Gauss base.

Appendix B

Finesse script

The Finesse script [10] used to simulate a Fabry Perot cavity equivalent to the one described in the table 2.1 is presented below. It is often extremely useful to create some simple Finesse model to check the exactitude of OSCAR results whenever possible. The script is also included in the OSCAR distribution under the name P_circ.kat in the folder Calculate_Pcirc.

```
l i1 1 0 n0                           # laser P=1W
gauss G1 i1 n0 0.017221 -517.112239   # Parameters for a beam radius 2cm
                                      # and a wavefront RofC of -2000m

s s0 1n 1 n0 n1
m1 SITM1 1 0 0 n1 n2
s sITM2 1n 1.45 n2 nITM
m1 ITM2 0.005 50E-6 0 nITM ncav1      # IM transmission and loss
attr ITM2 Rc -2000

s s3 1000 ncav1 ncav2

m1 ETM 50E-6 50E-6 0 ncav2 n4         # EM transmission and loss
attr ETM Rc 2000

cav cav_mode ITM2 ncav1 ETM ncav2

maxtem 8
trace 10
phase 0
startnode n0
```

```
pd0 p_circ ncav1
pd0 p_trans n4
pd0 p_ref n1

xaxis ETM phi lin -233 -234 3600
yaxis abs

gnuterm NO
retrace off
```