TP PROJET
04/05/2024

Meta - lenses

Students :
Ambre GOHIER & **Talla DIAGNE**
Teacher :
Denis RIDEAU

Summary

1 Introduction

A meta-surface is composed of nanostructures smaller than the wavelength of light, enabling the creation of ultra-thin functional devices. Research using electron beam lithography has shown that a wide range of optical devices, including lenses, spectral filters, polarization band-pass filters, and beam shapes, can be effectively replaced by 'flat' meta-surface optics. These meta-surfaces have reached a level of advancement suitable for large-scale production. Semiconductor companies are gearing up for a significant increase in the meta-surface technology market in the coming years. To support this transition, accurate simulations of optical meta-surfaces and innovative design approaches are essential.

Traditional simulation techniques, like Finite-Difference Time-Domain (FDTD), can be challenging to implement due to the relatively large dimensions (mm) of these optical systems and the sub-wavelength diffraction patterns involved. Therefore, alternative approaches based on simplified models have proven effective, facilitating the simulation of large meta-surfaces.

The primary objective of these projects is to explore the potential of these new meta-surfaces and gradually develop the expertise to design meta-lenses.

Figure 1 depicts a simplified diagram of the diffraction structure. Incident light propagates from the left side and passes through the layers of the meta-surface. After reaching the near-field (NF) plane, the wave continues to propagate through free space towards the far-field (FF).

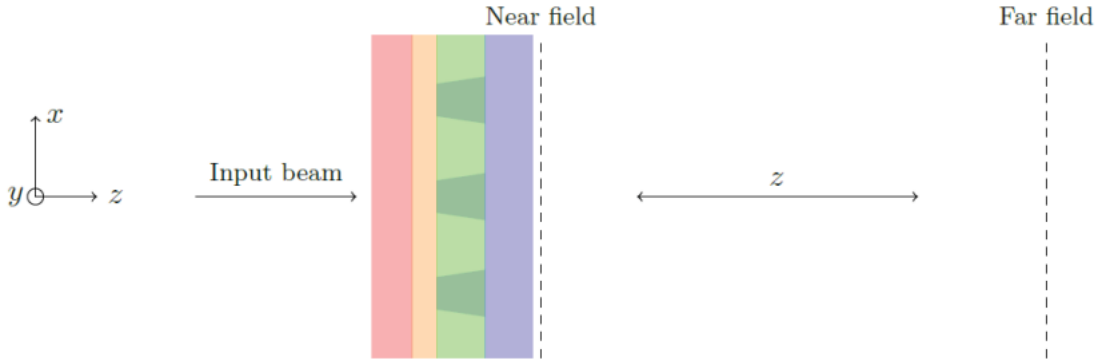


FIGURE 1 – When a laser illuminates a meta surface composed of pillars, it generates a distinct near-field phase pattern. After propagating a distance z, we measure the electromagnetic field's amplitude in the so-called far field.

Our initial focus will be on understanding the phase shift of the wave as it passes through a specific lens and its connection to the optical characteristics of the propagating waves. The phase of the signal contains significant information about the resulting image once it has propagated a distance z. We aim to acquire the knowledge necessary to project the Supoptique logo in the far field. [1]

1. Denis Rideau's document served as the model for this introduction. For the source, see [1].

2 Step 1 : Copying a moth's eye

In this part, we will draw inspiration from a moth's eye to craft our first meta-surface, which aims to enhance the transparency of an interface (to allow light to pass through).

2.1 The relationship between the dielectric constant and wavelength

As an initial step in optical modeling, engineers need to obtain the required data tables for all materials used in the simulation. To do that, we use the data from the website refractive index and extract the silicon SiO2 material that we would like to compare with.

We follow the following steps :

1. Shelf = "MAIN"
2. Book = "Si" (in Oxygen and oxides) or "SiO2"
3. Page = "Popova"
4. Go in the section called "Optical constants of SiO2 (Silicon dioxide, Silica, Quartz)" and download the DATA in "TXT - Tab separated".
5. Copied and baste the data in a Excel with two columns : one for the wavelength (wl) and one for the index (n)

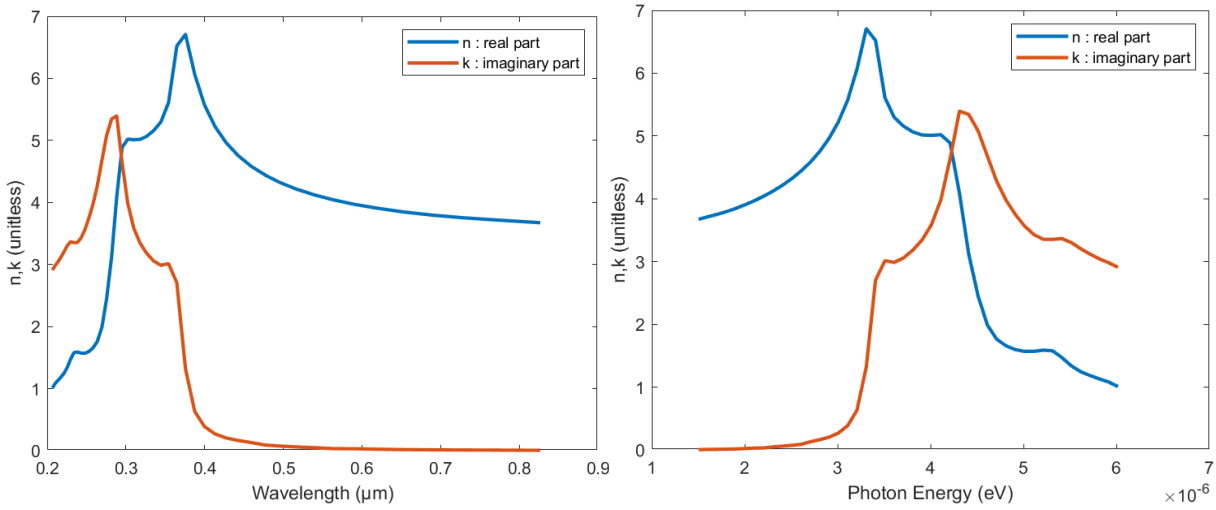
Now that we have the data, we would like to generate figures showing n and k according to the wavelength in two different units, eV and μm (see Figures 3 and 4). To do this and for practical modeling purposes, it's essential to understand the relationship between the pair (n, k) and the material's dielectric constants (ϵ_r, ϵ_i). Similarly, it's crucial to correlate the beam energy E with its wavelength, λ (let us recall that $E(eV) = \frac{hc}{\lambda e}$ with $e = 1,602.10^{-19} J$). We use the code that's presented below to make four figures.

$$E = \frac{nhcN}{\lambda} \quad with \quad N \quad the \quad number \quad of \quad photon \tag{1}$$

$$[n_r'(w) + ik(w)]^2 = \epsilon_r + i\epsilon_i \tag{2}$$

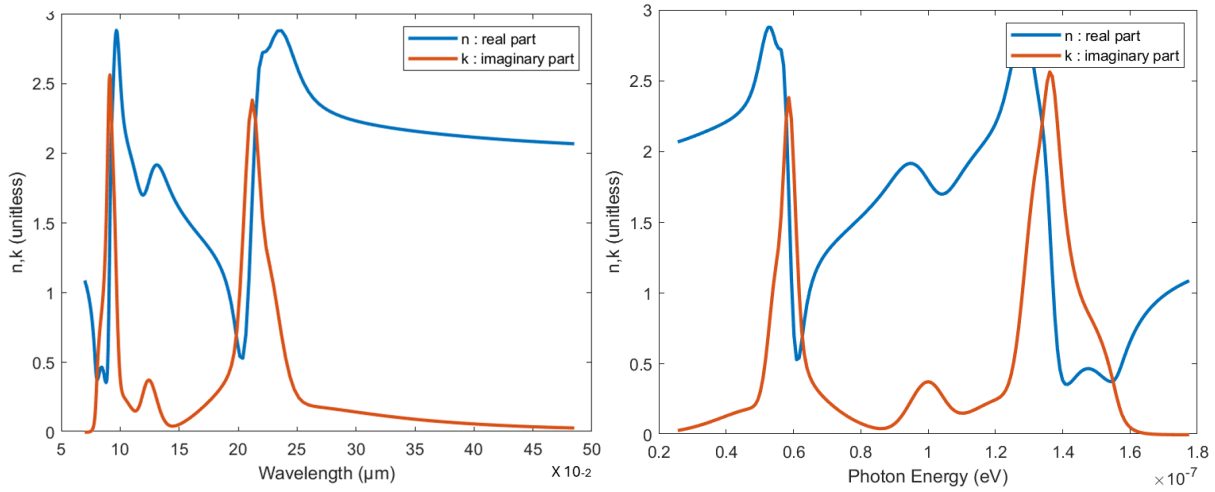
```
close all
% Constantes
h = 6.63e-34;
c = 3.00e8;
e = 1.6e-19;
%chargement des données
load(('Si_Palik.txt'),'ascii');%changer Si_Palik avec SiO2 quand
   necessaire
lambda=Si_Palik(:,1);%nm
E=h.*c./e./lambda;%eV
n=Si_Palik(:,2);
k=Si_Palik(:,3);
figure;
plot(lambda/1000,Si_Palik(:,2))%micromètre
hold on
plot(lambda/1000,Si_Palik(:,3))%micromètre
xlabel('Lambda (um)'),ylabel('n, k[1]');
legend('n','k');
figure;
plot(E,Si_Palik(:,2))%eV
hold on
plot(E,Si_Palik(:,3))%eV
xlabel('Photon Energy (eV)'),ylabel('n, k[1]');
legend('n','k');
```

FIGURE 2 – Code to generate Figures 3 and 4



(a) (b)

FIGURE 3 – Real and imaginary parts of Si index function of wavelength (a) and photon energy (b)



(a) (b)

FIGURE 4 – Real and imaginary parts of SiO2 index function of wavelength (a) and photon energy (b)

Another important quantity related to the imaginary part of ϵ is the optical absorption coefficient α. This value is a measure of how strongly the material absorbs light at a particular wavelength.

$$\alpha = \frac{4\pi k}{\lambda} \tag{3}$$

The Beer-Lambert absorption law is :

$$I = I0e^{-\alpha d} \tag{4}$$

With I_0 the initial intensity of the beam before entering the material, I is the transmitted intensity after passing through the material, α is the absorption coefficient, and d is the thickness of the material through which the beam is passing.

So, the Beer-Lambert Law states that the intensity I of light passing through an absorbing medium decreases exponentially with the thickness of the medium. This relationship helps in determining properties such as the absorption coefficient, which can provide insights into the characteristics of the material, such as its composition and structure. It is also useful in various applications, including the analysis of chemical concentrations, understanding of light-matter interactions, and designing devices like optical filters and sensors.

Since k represents the imaginary portion, Figures 3 and 4 enable us to understand that the material can be transparent or absorbent at a given wavelength (visible at 0.4 to 0.8 μm and infrared at 1 μm). By examining the curves, we can observe that Si is somewhat transparent in the visible spectrum and rather absorbant in the UV. In the SiO2, is nearly zero, making it absorbent between 250 and 500 nm.

2.2 Transmission, Absorption and Reflection coefficients

When a light beam enters a material, part of it is reflected back at the interface while the rest is transmitted (or refracted) into the material. Augustin-Jean Fresnel described this phenomenon, explaining that the amount of light reflected is quantified by the reflectance, R, and the amount transmitted is quantified by the transmittance, T. This occurs because light waves experience reflection at points where there is a difference in optical permittivity (also known as impedance) between adjacent materials. This difference is often referred to as an impedance mismatch. The degree of reflection of the light's energy at the boundary between two materials is directly proportional to the severity of the impedance mismatch ; thus, a greater mismatch leads to a larger portion of the light's energy being reflected.

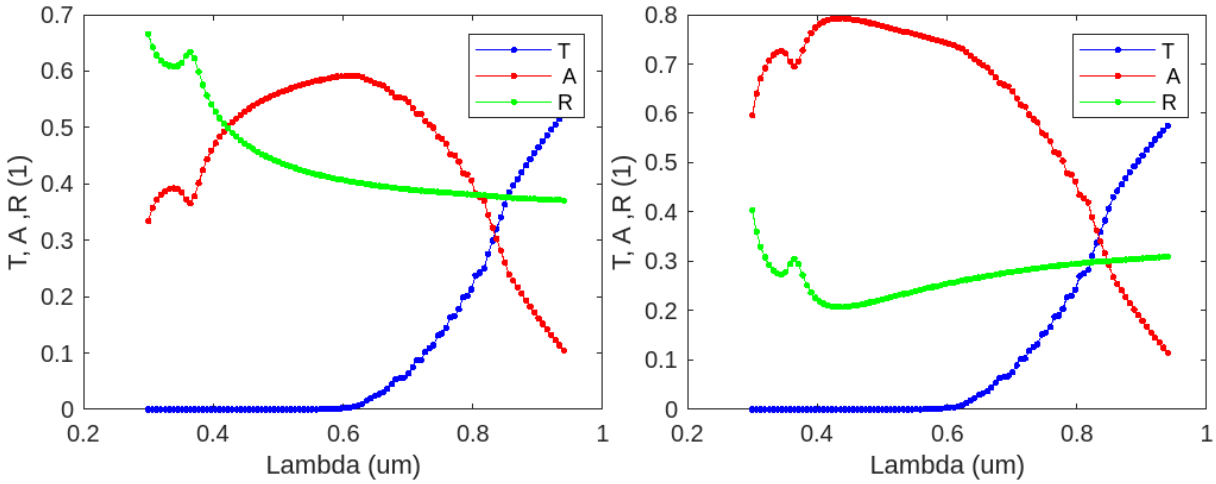
The community focused on optical devices has long been engaged in researching Anti-reflective coatings (ARCs) to reduce effective reflectance and glare. A well-known approach involves depositing multiple layers on the surface of the material to create a gradual adjustment in impedance.

The expression of the (amplitude) reflection r and transmission t coefficient at normal incidence are :

$$r = \frac{n_1 - n_2}{n_1 + n_2} \qquad t = \frac{2n_1}{n_1 + n_2} \tag{5}$$

With n_1 is the refractive index of the initial medium (from which the light is coming) and n_2 is the refractive index of the second medium (into which the light is entering). r measures the fraction of the incident wave's amplitude that is reflected back into the first medium, and t measures the fraction of the incident wave's amplitude that is transmitted into the second medium.

We implement these equations for Air/Si and Air/ML/Si systems and obtain the Figures 5 with $T = |t|^2$, $R = |r|^2$ and A + R + T =1 with the code from Annexe A. These figures are implement with a Rigorous Coupled-wave Analysis (RCWA). This code solves the Maxwell's equations, decomposing the incident light into planar waves of different directions, and taking into account both the amplitude and phase of the light wave. With that code one propagates the incident wave into a small portion of the coating (a square of 2.5 µm) and can calculate numerically, for each wavelength, what portion of the incident light is transmitted, reflected or absorbed.



(a) Transmission, Absorption and Reflection of a Air/Si

(b) Transmission, Absorption and Reflection of a Air/ML/Si

FIGURE 5 – T, A and R coefficients for different layers at normal incidence, calculated with RCWA

As we can see, the absorbent coefficient increases with the number of layers. The reflexion and transmission coefficiets are weakened as a result. However, this data was at a typical incidence.

When dealing with light that is incident at an angle other than normal (perpendicular) to the boundary between two media, the expressions for the reflection coefficient (R) and the transmission coefficient (T) become dependent on the angle of incidence as well as the indices of refraction of the two media. These equations differ for two polarizations : perpendicular (s-polarization) and parallel (p-polarization) to the plane of incidence. So we write the equations for at an interface between two media with indices of refraction n_1 and n_2, and with an angle of incidence θ [2].

$$R_s(\theta) = \frac{[\sqrt{a+z} - \sqrt{2}cos(\theta)]^2 + a - z}{[\sqrt{a+z} + \sqrt{2}cos(\theta)]^2 + a - z} \quad and \quad T_s(\theta) = 1 - R_s(\theta) \tag{6}$$

$$R_p(\theta) = \frac{[\sqrt{a+z} - \sqrt{2}sin(\theta)tan(\theta)]^2 + a - z}{[\sqrt{a+z} + \sqrt{2}sin(\theta)tan(\theta)]^2 + a - z} \quad and \quad T_p(\theta) = 1 - R_p(\theta) \tag{7}$$

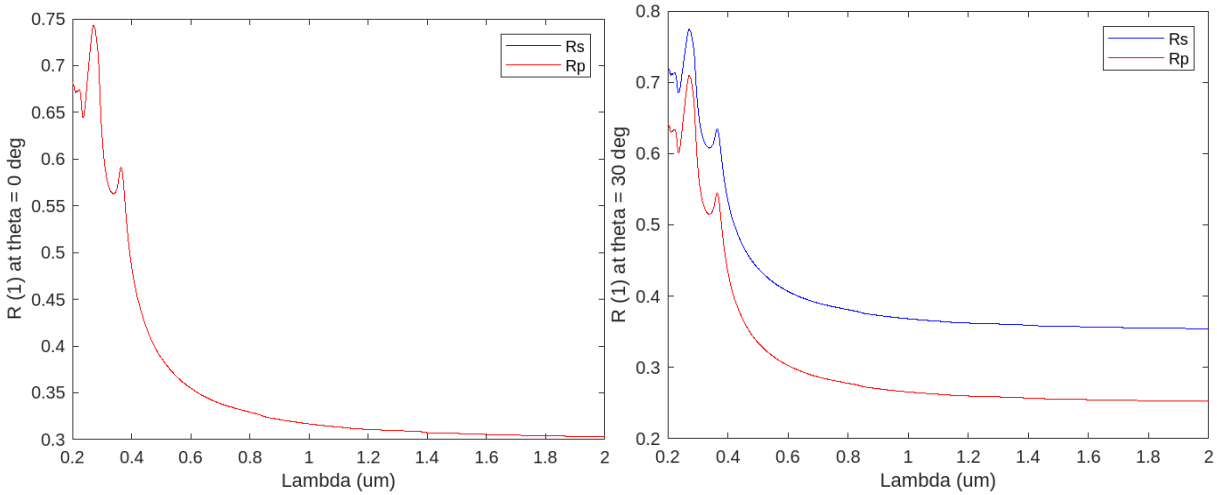
2. For the source, see [7].

With, $z = n^2 - K^2 - sin(\theta_1)^2$ and $a = \sqrt{z^2 + 4n^2K^2}$.

For unpolarized light, the average reflection and transmission coefficients can be approximated as :

$$R = \frac{R_s + R_p}{2}, \quad T = \frac{T_s + T_p}{2} \tag{8}$$

We obtain two figures with two different angles for the same system, Air/Si (see Figure 6). It is seen that the blue of Rs lies beneath the red of Rp in the figure (a).



(a) Reflection at 0 degrees as a function of λ for Rs and Rp polarizations

(b) Reflection at 30 degrees as a function of λ for Rs and Rp polarizations

FIGURE 6 – Varying degrees of reflection

We observe that Rs and Rp are not reflected in the same way when the angle is larger. Figure 7 illustrates the evolution.

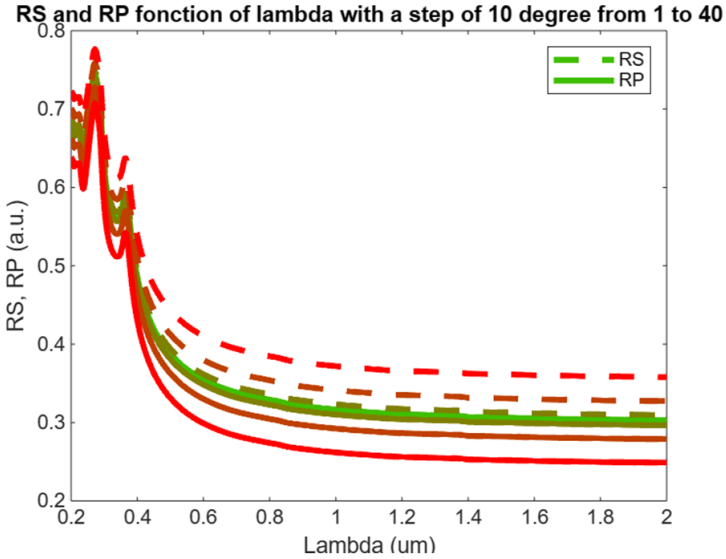


FIGURE 7 – Rs and Rp as a function of λ with a step of 10 degrees

To better describe, we can plot the R and T coefficients for angles between 0 and 90° and for TE and TM polarization. This is what is displayed in Figure 8. We can see the reflections are equal at normal incidence, and then become different with one always increasing and the other one slightly decreasing and reaching 0 before increasing.

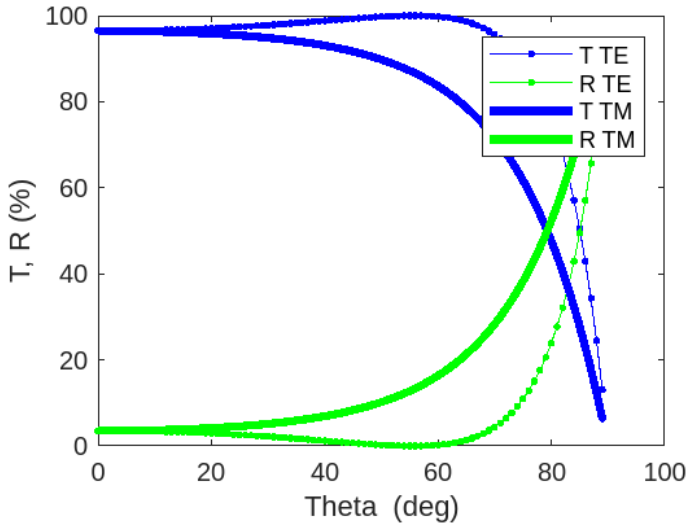


FIGURE 8 – Transmission and reflection of a Air/SiO2 system as a function of incident angle for two polarizations

We see that a given angle there is the **Critical Angle**. This angle (θ_c) is the smallest angle of incidence at which total internal reflection occurs. For angles of incidence greater than the critical angle, all the incident light is reflected, and none is transmitted through the boundary. The critical angle can be calculated using Snell's law :

$$n_1 sin(\theta_1) = n_2 sin(\theta_2)$$

where :
— n_1 is the index of refraction of the denser medium (higher index of refraction)
— n_2 is the index of refraction of the less dense medium (lower index of refraction)
— θ_1 is the angle of incidence in the denser medium
— θ_2 is the angle of refraction in the less dense medium

For total internal reflection to occur, the angle of refraction (θ_2) would reach 90 degrees, since beyond this angle, the refracted ray does not enter the second medium but skims along the interface. Setting $\theta_2 = 90$ in Snell's law gives :

$$\theta_c = arcsin(\frac{n_2}{n_1}) \tag{9}$$

For the critical angle to exist :
— Light must be traveling from a medium of higher refractive index to one of lower refractive index
— θ_c is only defined for $n_1 > n_2$ otherwise, the arcsinus function does not yield a real number since the sine of an angle cannot be greater than one.

2.3 ARC with a moth's eye

The objective is to develop an Anti-Reflective Coating (ARC) by structuring the silicon layer to resemble the pattern of a moth's eye, as depicted in Figure 9 and discussed in reference [3].

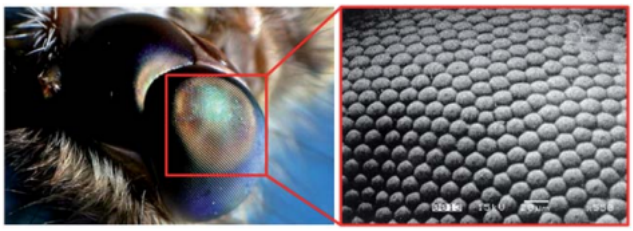


FIGURE 9 – The image on the left shows the eye of the moth, while the SEM image on the right displays the nano bumps or protrusions on the outer surface of the corneal lens.

Using RCWA, we model the absorption and reflection as a function of wavelength and simulate the effect of a surface structuration on light reflection on the Air/Si system. The surface structuration is represented by Figure 10, and the output of the RCWA code is shown in Figure 11. This time, exercise caution as the coefficients' units are percentages.

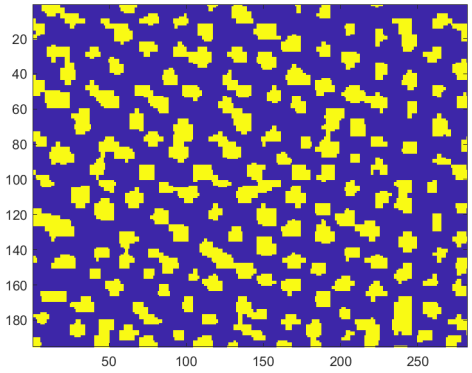


FIGURE 10 – Structure used for the anti-reflective metasurface simulation

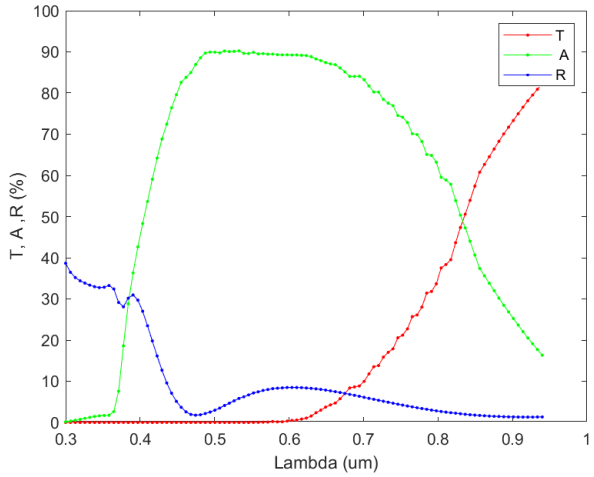


FIGURE 11 – T, R and A of the Air/Si system the surface of which is structured with a random pattern mimicking the moth's eye

Results : when compared to all other examples previously analyzed, we can observe a relatively rapid decline in the R coefficient. Therefore, we may infer that, in comparison to the single and multilayer coatings previously researched, the utilization of a nano-sized metastructure is an excellent solution for our anti-reflective coating.

2.4 Wave front phase shift

We characterize the maximum phase shift as the discrepancy in phase between a wave traversing through a reference material of uniform composition, having a thickness of h_0 and a refractive index of n_1, and a wave traversing through a non-uniform layer also with a thickness of h_0, consisting of two materials with refractive indices n_1 and n_2 respectively.

The source [1], page 5, 6 and 7, helps us to find the thickness of this non-homogeneous layer to varies the phase shift up to to 2 in the second material with respect to the reference one. We can write that :

$$h_0 = \frac{\lambda_0}{n_1 - n_2} \tag{10}$$

3 Step 2 : Propagation of light

We shall discuss the close field and the far field in this section. We also describe the Canonical transformation, which has a proportionality to the angle α_z and a variable change.

3.1 Definition of the Fourier transform

The Fourier transform and the inverse Fourier transform are defined as follows :

$$F[H(x,y)](k_x,k_y) = \iint_{-\infty}^{\infty} H(x',y') e^{-i(k_x x' + k_y y')}\, dx'\, dy' \tag{11}$$

$$F^*[F(k_x,k_y)](x,y) = \iint_{-\infty}^{\infty} F(k_x,k_y) e^{i(k_x x + k_y y)}\, dk_x\, dk_y \tag{12}$$

In practical applications, the real space is finite, with dimensions L_x and L_y, and discretized into N_x and N_y points. It's important to note that the FFT yields a set of Fourier coefficients $F_{nx,ny}$, with nx ranging from $-\frac{N_x}{2}$ to $\frac{N_x-1}{2}$ and ny ranging from $-\frac{N_y}{2}$ to $\frac{N_y-1}{2}$, implicitly defining the Fourier space as :

$$\left\{ \frac{2\pi nx}{L_x} \mid nx \in \left[-\frac{N_x}{2}, \frac{N_x - 1}{2} \right] \right\} \tag{13}$$

$$\left\{ \frac{2\pi ny}{L_y} \mid ny \in \left[-\frac{N_y}{2}, \frac{N_y - 1}{2} \right] \right\} \tag{14}$$

It is possible to implement a "brute" force FT. This is a good exercise to understand the basics of Fourier analysis. We can start with a 1D model and then implement a 2D model. Here are some explanation about this two models :

— In a 1D model, you have a signal f(x) where x is the independent variable. The Fourier Transform of f(x), denoted as F(k), is given by :

$$F(k) = \int_{-\infty}^{\infty} f(x) \cdot e^{-2\pi i k x} \, dx \tag{15}$$

For a discrete implementation, we approximate this integral by a sum :

$$F(k) \approx \sum_{n=0}^{N-1} f(n\Delta x) \cdot e^{-2\pi i k n \Delta x} \tag{16}$$

Where Δx is the spacing between the sample points and N is the total number of sample points.

— For the 2D case, where you have a function f(x,y), the Fourier Transform $F(k_x, k_y)$ is given by :

$$F(k_x, k_y) = \iint_{-\infty}^{\infty} f(x, y) \cdot e^{-2\pi i (k_x x + k_y y)} \, dx \, dy \tag{17}$$

For a discrete implementation, similar to the 1D case, you'll have :

$$F(k_x, k_y) \approx \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m\Delta x, n\Delta y) \cdot e^{-2\pi i (k_x m \Delta x + k_y n \Delta y)} \tag{18}$$

One can make a comparison between an FFT-based solution and a basic 2D example. Regretfully, our algorithms did not function as intended ; however, the FFT-based solution outperforms the brute-force method in terms of speed, particularly for bigger datasets.

3.2 Near Field and Far Field

A scalar approximation of the vectorial electromagnetic wave can be derived using the Fresnel transform. Assuming uniform wave propagation (with a wavelength of $\lambda = \sqrt{\epsilon_r}\lambda_0$), the fields at any given position can be inferred from the field at a location situated at z=0.

$$E(x, y, z) = \frac{1}{i\lambda} \iint_{-\infty}^{\infty} E(x', y', z = 0) \frac{e^{ikr} z}{r^2} dx' dy' \tag{19}$$

Where $E(x', y', z = 0)$ represents the near field, and $r = \sqrt{(x - x')^2 + (y - y')^2 + z^2}$. Part 3.6 illustrates how this equation is put into practice.

3.3 Fresnel integrals as a convolution

Given that the Near Field region, where the electric field is non-zero, has dimensions considerably smaller than z, the Fresnel approximation $r \approx z + \frac{(x-x')^2+(y-y')^2}{2z}$ can be employed for the exponential term (and $r \approx z$ in the $\frac{1}{r}$ denominators) :

$$E(x,y,z) = e^{ikz}\frac{i}{\lambda z}\iint_{-\infty}^{\infty} E(x',y',z=0)e^{ik\frac{(x-x')^2+(y-y')^2}{2z}}\,dx'\,dy' \tag{20}$$

This equation represents a convolution that can be effectively computed in reciprocal space by multiplying the Fourier transforms of the two terms and then performing an inverse Fourier transform. For efficiency purposes, one can utilize the Fourier transform of the impulse response of the free space propagation kernel.

$$H_{Fresnel}(x,y) = e^{i\frac{\pi}{\lambda z}(x^2+y^2)} \tag{21}$$

$$\mathcal{F}\{e^{i\frac{\pi}{\lambda z}(x^2+y^2)}\}(k_x,k_y) = i\lambda z e^{-i\frac{\lambda z}{4\pi}(k_x^2+k_y^2)} \tag{22}$$

When considering the reciprocal space defined with the (2π) factors, by redefining the pre-exponential factor in the kernel, one can express the Fourier transform of Equation 20 as :

$$F[E(x,y,z)](k_x,k_y) = e^{iz\left[\frac{2\pi}{\lambda}+\frac{\lambda(k_x^2+k_y^2)}{4\pi}\right]} \times F[E(x,y,z=0)](k_x,k_y) \tag{23}$$

3.4 Helmhotz as a convolution

It's worth noting that this kernel closely resembles another more precise kernel, with which the convolution satisfies the Helmholtz equation. We can write the Equation 23 as :

$$F[E(x,y,z)](k_x,k_y) = e^{i\frac{z}{2\pi}\sqrt{(\frac{2\pi}{\lambda})^2-k_x^2-k_y^2}} \times F[E(x,y,z=0)](k_x,k_y) \tag{24}$$

3.5 Fresnel integrals with a canonical transformation

An alternative way of evaluating the Fresnel integral can be performed using a Fourier transform and a canonical transformation. Using :

$$(x-x')^2 = x^2 + x'^2 - 2xx' \tag{25}$$

$$(y-y')^2 = y^2 + y'^2 - 2yy' \tag{26}$$

and $k = \frac{2\pi}{\lambda}$, the expression becomes :

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z}e^{\frac{i\pi}{\lambda z}(x^2+y^2)}\iint_{-\infty}^{\infty} E(x',y',z=0)e^{\frac{i\pi}{\lambda z}(x'^2+y'^2)}e^{-i2\pi\left(\frac{x}{\lambda z}x'+\frac{y}{\lambda z}y'\right)}\,dx'dy' \tag{27}$$

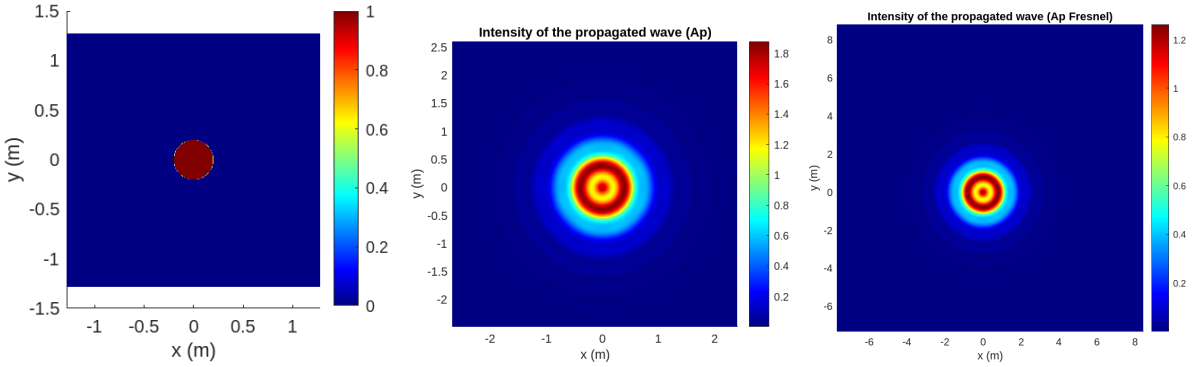
We recognize a Fourier transform with wave numbers $k_x = \frac{2\pi x}{\lambda z}$ and $k_y = \frac{2\pi y}{\lambda z}$. It ends with :

$$E(x,y,z) = \frac{e^{\frac{i2}{\lambda}}}{i\lambda z}H(x,y)F\left[E(x',y',z=0)H(x',y')\right](k_x,k_y) \tag{28}$$

One notices the implicit dependence of x and y in k_x and k_y.

3.6 Pinhole and Far Field

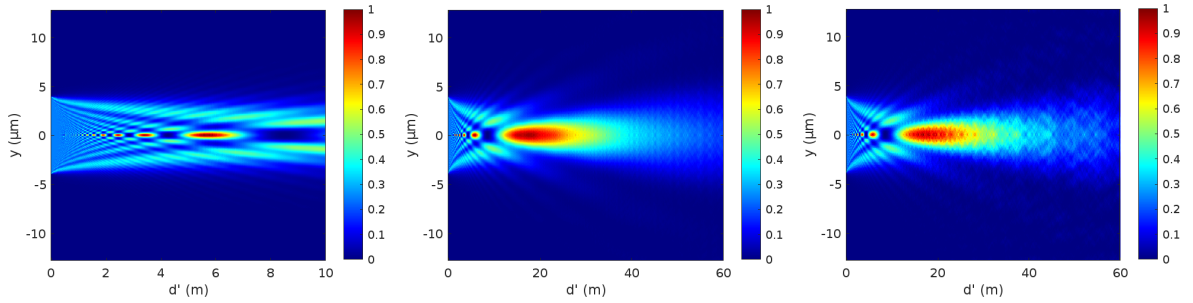
To further proceed, let's define two planes, namely the Near Field plane at z = 0 and the Far Field plane at a given z. We have observed that the far field plane and the reciprocal Fourier space are linked by a canonical transformation. Recalling the definition of the reciprocal wave vectors, one can therefore define the set of points in the Far field space (x*,y*) := $(\lambda z k_{x,y})$. Considering the Fourier space definition, the Far Field space $(x_n*) := \frac{n\lambda z}{L_x}$, where n $\in [\frac{-N_x}{2}, \frac{N_x-1}{2}]$, differs from the Near field space $(x_n) := \frac{nL_x}{N_x}$, where n $\in [\frac{-N_x}{2}, \frac{N_x-1}{2}]$. One also notes that the size along the X direction of the far field (x*) is smaller than the near field one L_x when z$<\frac{N_x^2}{L_x\lambda}$. Three Figures are shown in Figure 12, which we obtained by using the code from section B.



(a) Pinhole of 4 μm diameter (b) With the Fresnel method (c) The Convolutional method

FIGURE 12 – Propagation of a pinhole at $z = 10\,\mu$m

The boundaries for the Fourier transform integration are theoretically infinite. However, in practice, padding with zeros outside the near field plane can be employed to effectively increase its size. Figures 13 provide a comparison between convolution and canonical transform, highlighting the impact of padding.



(a) 8 μm pinhole for $z \leq 10\,\mu$m (Fresnel transforms)

(b) 8 μm pinhole for $z \leq 60\,\mu$m (Fresnel transforms with zero-padd)

(c) 8 μm pinhole for $z \leq 60\,\mu$m (Fresnel transforms without zero-padd)

FIGURE 13 – Near field domain with a pinhole, vertical cut using convolution with FFT

The Figure [14] is quite different because wa still had to implement the interpolation for the variable change but it still seems correct.

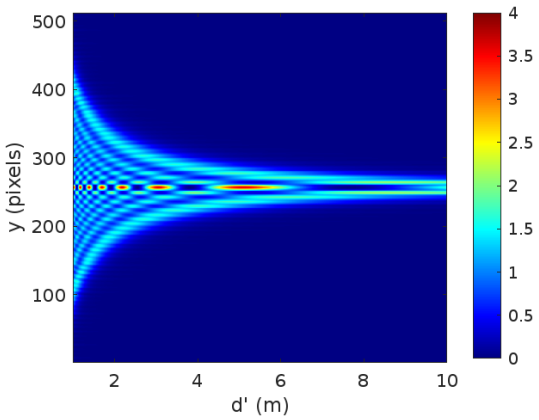


FIGURE 14 – Near field domain with a pinhole, vertical cut using canonical transform with FFT

4 Step 3 : Design of a lens

We begin by revisiting the theoretical lens equation to solidify our understanding of how lenses function to manipulate light paths. This formula is crucial for any practical application involving lens design :

$$\frac{1}{f'} = \frac{1}{x'} + \frac{1}{x} \tag{29}$$

Following this, we will simulate the in-plane phase shift that occurs just after light passes through the lens. This involves plotting how the phase of the light changes across the plane of the lens, providing a visual representation of the optical effect induced by the lens :

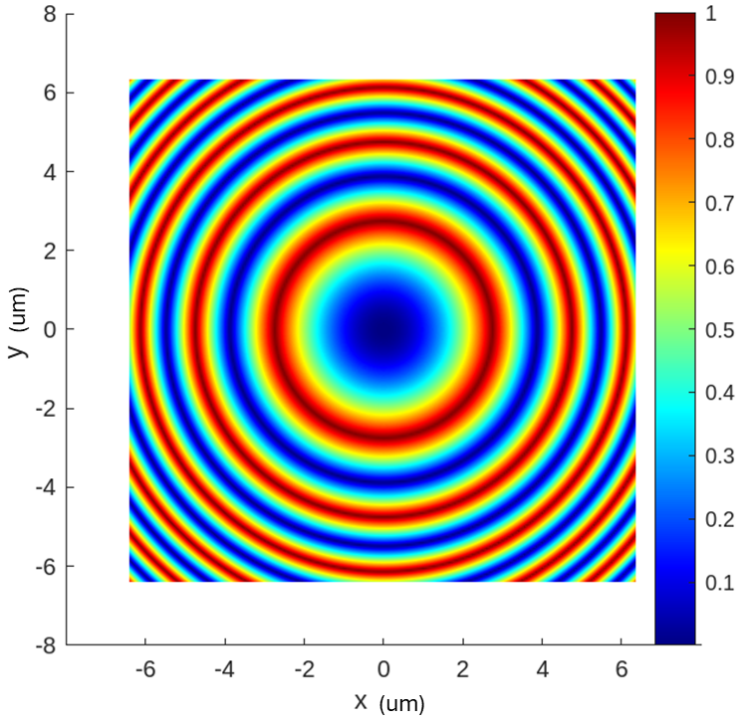
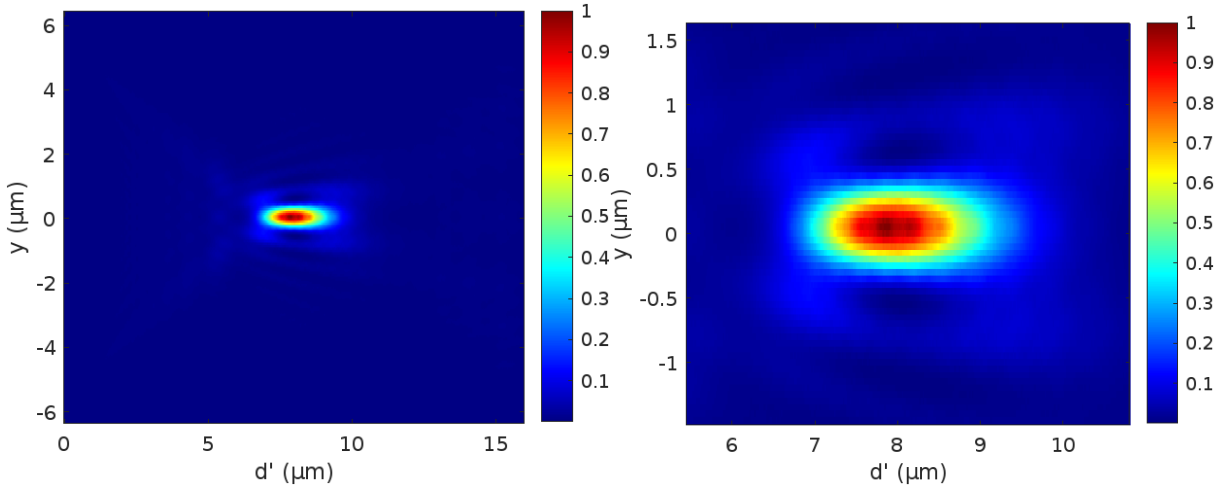


FIGURE 15 – In plane phase shift after the lens

Then we examine the vertical central-cut of the propagating fields. This involves plotting both the modulus (or magnitude) and phase of the light field as it moves through space. This plot will help us identify the focal point of the lens, which is where the light rays converge, creating a sharp focus. An example of these plots can be found in Figure [16].



(a) Vertical central-cut (modulus)

(b) Zoom aroud the focal point

FIGURE 16 – Vertical central-cut of the propagating fields (modulus), focal length of $f = 8\,\mu$m at 940 nm

However, we can notice that the focal point is not well resolved : we can measure $2\,\mu$m on the Figure[16b]. We can resolve this by increasing the number of pixels on the image. Here is the result of it after going from 256 to 1000 pixels on each side :

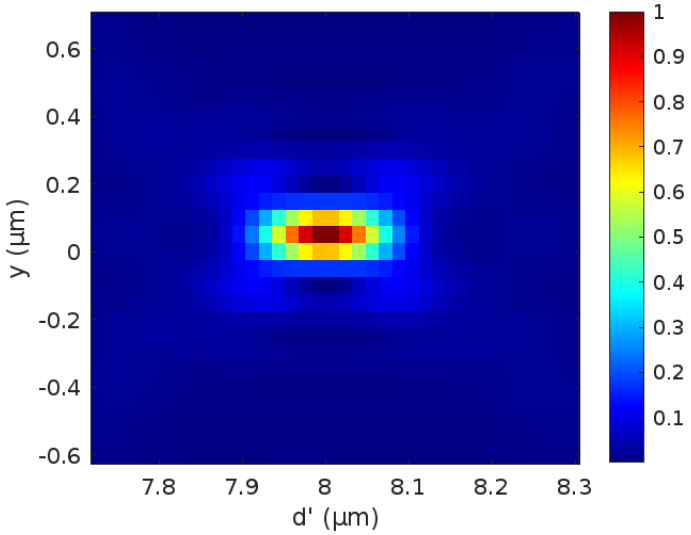


FIGURE 17 – Vertical central-cut of the propagating fields (modulus) with a better resolution

Here is what we obtain for the phase of the propagating fields : we can also notice that that at $z = 8\,\mu$m, we have a symmetry for the converging incident wave.

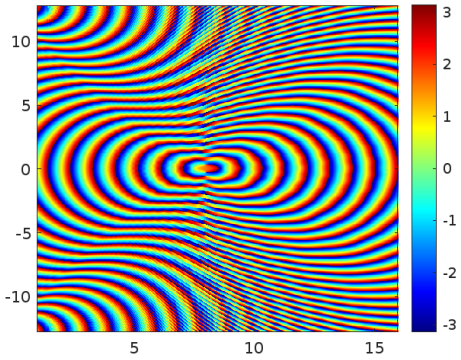
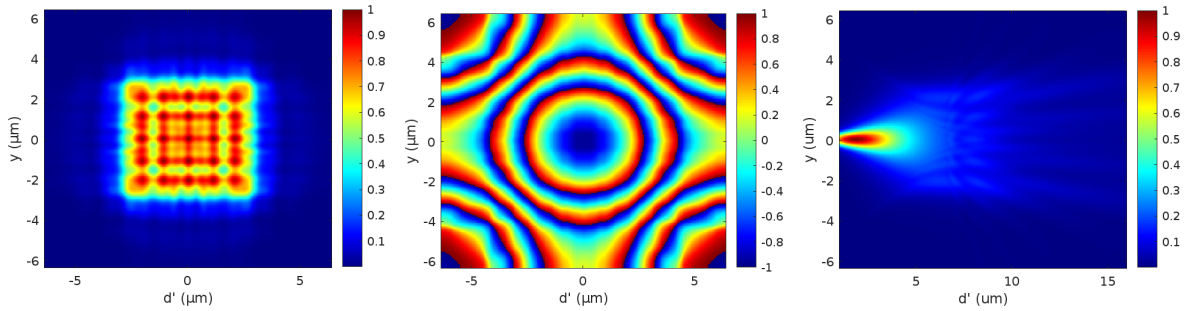


FIGURE 18 – Vertical central-cut of the propagating fields (phase)

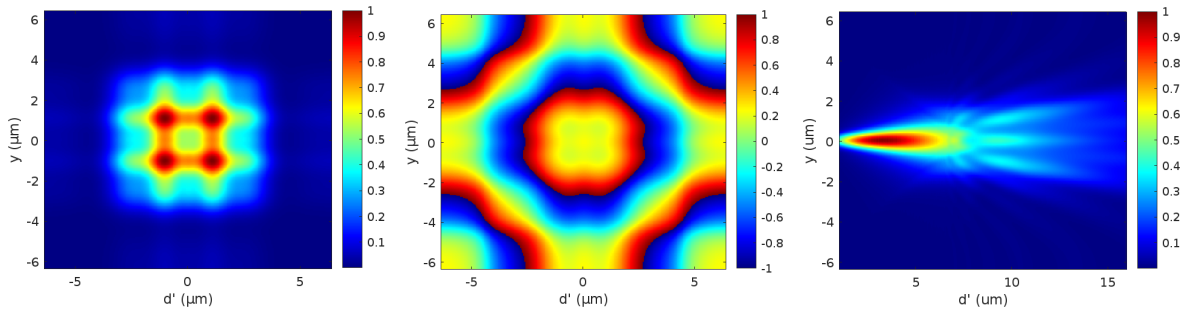
Once we are comfortable with these concepts and the basic lens equation, we proceed to study the propagation of light after passing through such a lens using the Helmholtz equation. This step involves a deeper dive into wave optics, allowing us to predict and visualize how light behaves in more complex scenarios. We will conduct simulations to propagate light at various positions beyond the lens and display 2D-cut planes at positions that are deemed relevant based on the optical setup and the specific experiment objectives. Additionally, we plot a vertical cut perpendicular to the lens to examine the cross-sectional distribution of the light field :



(a) Modulus (b) Phase (c) Vertical cut

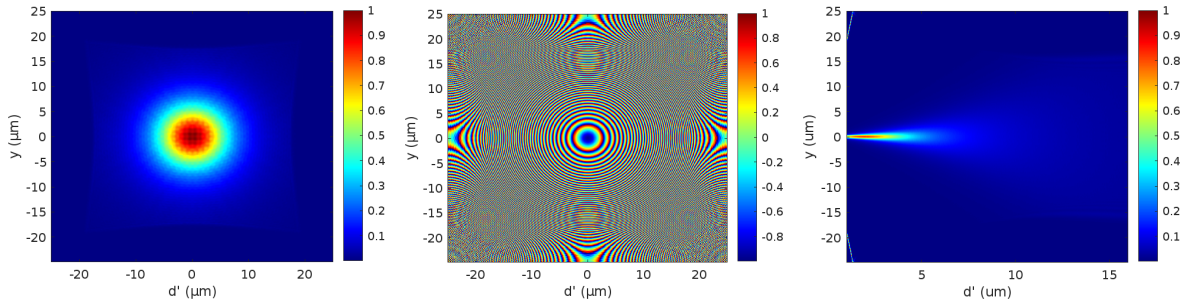
FIGURE 19 – Helmholtz method : Vertical central-cut of the propagating fields at the focal point, focal length of $f = 8\,\mu$m

Finally, we will vary the focal length and the lateral size of the lens to explore how these changes affect the focusing properties and the overall light distribution :



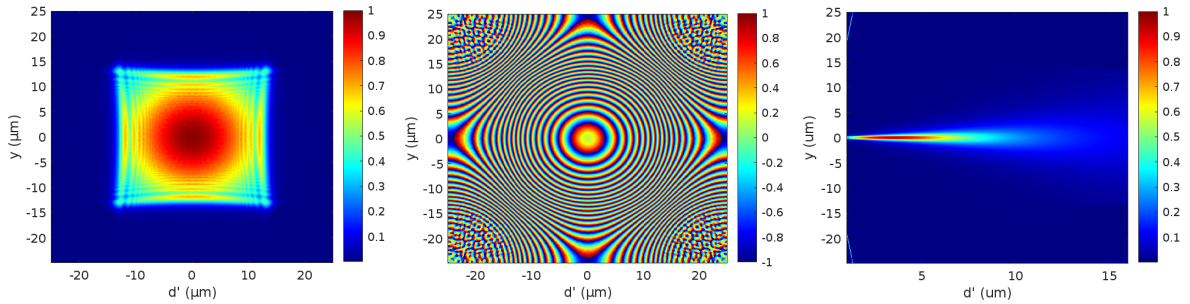
(a) Modulus (b) Phase (c) Vertical cut

FIGURE 20 – Helmholtz method : Vertical central-cut of the propagating fields at the focal point, focal length of $f = 20\,\mu$m



(a) Modulus (b) Phase (c) Vertical cut

FIGURE 21 – Helmholtz method : Vertical central-cut of the propagating fields at the focal point, focal length of $f = 8\,\mu$m and lateral size of $50\,\mu$m



(a) Modulus (b) Phase (c) Vertical cut

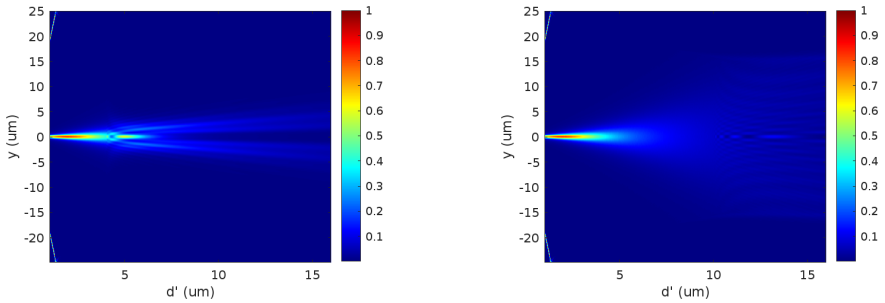
FIGURE 22 – Helmholtz method : Vertical central-cut of the propagating fields at the focal point, focal length of $f = 20\,\mu$m and lateral size of $50\,\mu$m

We can easily notice here that the focal length f and grid size L significantly impact the results. The focal length f determines the curvature of the phase profile in the complex transmittance function, affecting how sharply the wavefront focuses ; a shorter f enhances focusing power, leading to a more pronounced convergence, while a longer f produces a flatter wavefront that converges over greater distances. Meanwhile, the grid size L influences both the resolution and the field of view of the simulation, with larger values providing better spatial sampling that can capture more detailed wavefront behaviors and reduce aliasing in Fourier transforms. However, increased L also raises computational demands, impacting processing time for example. These parameters should be carefully selected based on the specific requirements of the simulation to balance accuracy and computational efficiency.

However, we could still improve these results by adjusting the zero-padding.

Finally, we can combine these lens configurations with a pinhole aperture to study the effects of aperture size on light diffraction and focus quality (see Figure 23).

Smaller apertures seem to have a higher tendency to show pronounced diffraction effects due to their closer approach to the wavelength of light used. The diffracted waves interfere more significantly, potentially creating complex patterns such as a central minimum or "hole" in the center of the main diffraction spot.



(a) Diameter pinhole : 8 μm (b) Diameter pinhole : 20 μm

FIGURE 23 – Helmholtz method : Vertical central-cut of the propagating fields at the focal point, focal length of $f = 8\,\mu$m for different apertures

5 Step 4 : Toward a metasurface

There are three sections here. The first describes how to depict a three-dimensional meta surface ; the second describes the characteristics of the phase and transmission as a function of a pilar width ; and the third and final section examines Supoptique's logo.

5.1 Three-dimensional meta surface

Using a metasurface to implement the previously designed optical function is the last step in this course. As seen in Figure 24, where the silicon tube—which is incapable of floating in the air—is represented in black and the SiO2 is depicted in blue—we usually start by generating a 3D image. This figure represents a gaussian light-illuminated pillar. The metasurface, which turned into a miror, is this pillar. When the metasurface is fully lit, resonance occurs.

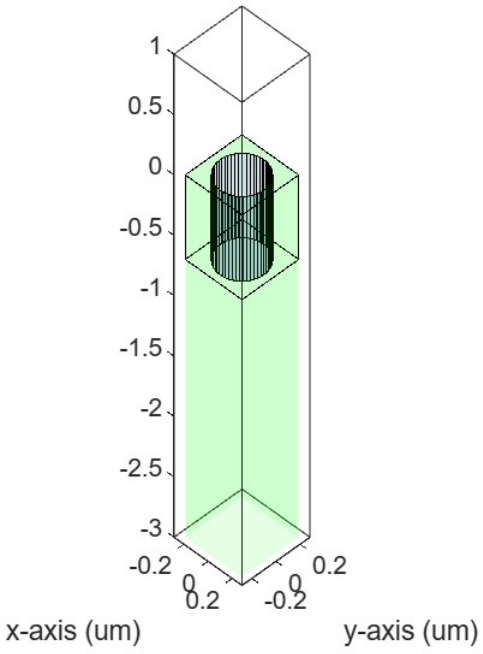


FIGURE 24 – Eight-mode 3D depiction of silicon in SiO2

5.2 Phase and transmission as a function of a pillar width

As seen in Figures 25 and 26, we can have a look-up table, also called a library, that shows the phase shift versus the width of the diffracting element (in this case, a pillar). We apply the code from Annexe C to obtain these Figures. The codes employ functions to alter the pillar's diameter dimension and to depict the transmission and phase as a function of diameter. The interval between each pillar can also be altered ; in Figure 25, it is 0,47 μm, while in Figure 26, it is 0,6 μm.

Alternatively, the mode can be altered by using the "OPTION.halfnpw =" 8 or 0 in the code included in the C Annexe. Two curves are depicted in Figure 25 : one with mode 0, which is the Fundamental of FFT and corresponds to a basic average of the structure permittivity, and the other with eight modes.

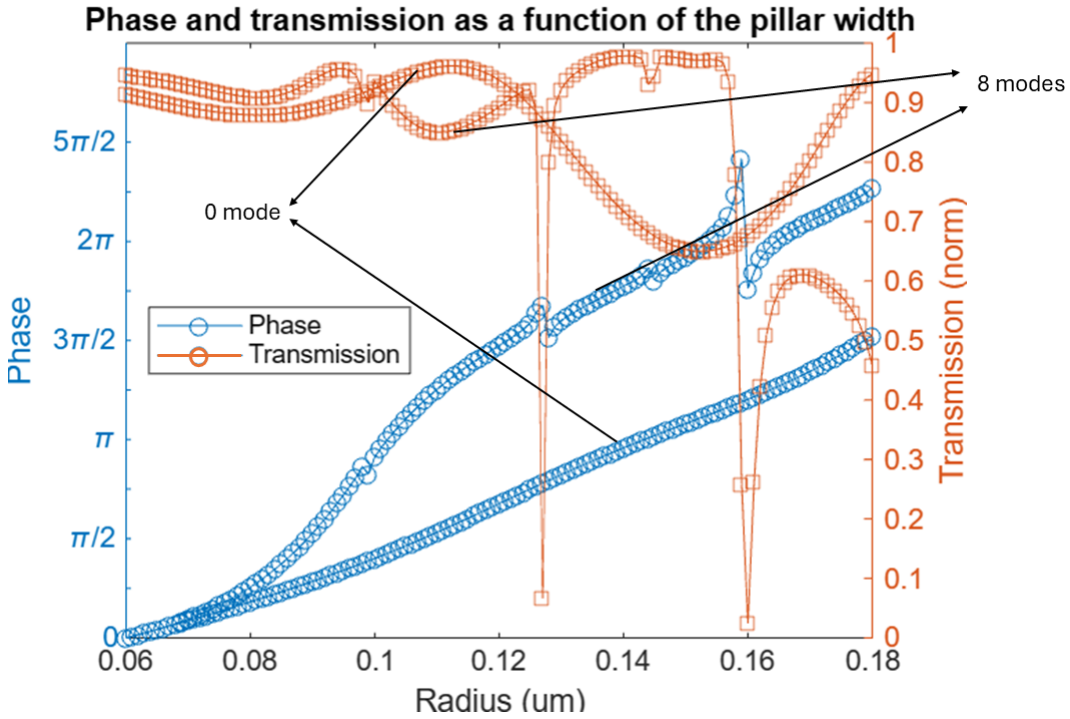


FIGURE 25 – Phase and Transmission : 0 and 8 modes

Explanation : both modalities have the same commencement of the phase and the transmission. After that, we can observe that the phase doubles with the addition of modes. That can be explained by the fact that the optical path grows (doubles in our case) as the wave bounds on the surface. This explains why the phase curve for the eight modes increases quickly.

We may observe that there are no points in the transmission where the fundamental is 0. This is a good news because it is challenging if we need to add modes if the wave failed to pass for the fundamental mode. The value of the transmission for eight modes decreases to zero for certain levels. It indicates that the surface is reflective. Finding the ideal radius for a suitable transmission is therefore feasible.

We are now altering the interval between the pillars. We entered 0.6 μm and put 8 modes. Figure 26 illustrates that a large number of points in the transmission are nulle. It is evident that there is no transmission between π and 2π. This is the plage where the pillar is shown (see Figure 24), which makes it problematic. Thus, the separation between each pillar is crucial.

These numbers are displayed following 215 seconds. Therefore, it will be ideal to optimize the code in order to get the results faster.

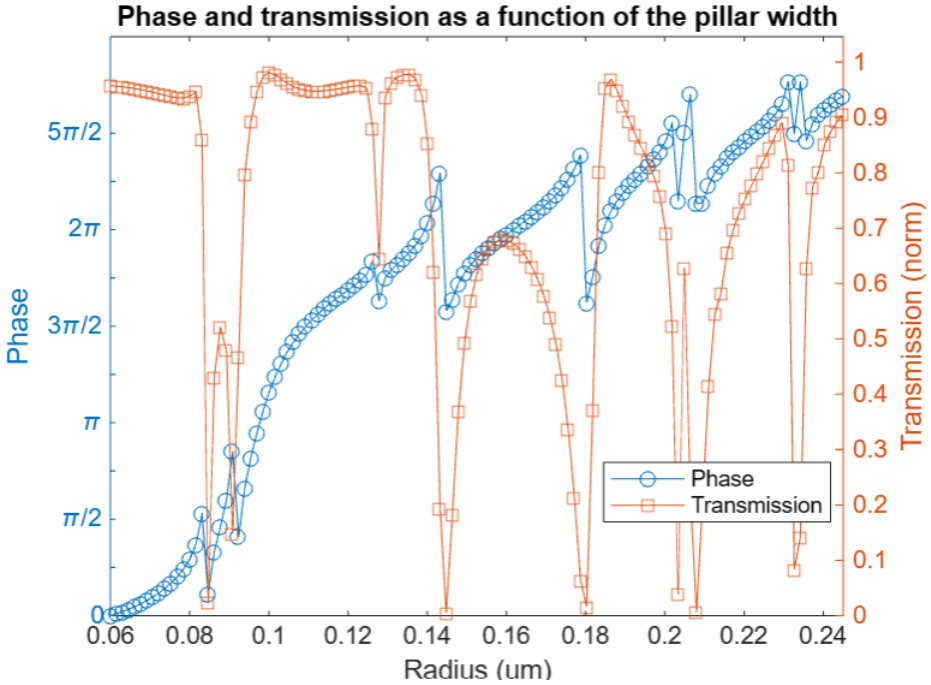


FIGURE 26 – Period of 0.6, instead of 0.47, between the pillar

5.3 Examination of Supoptique Logo

Projecting the Supoptique logo in the far field is the ultimate objective (see the logo in Figure 27).

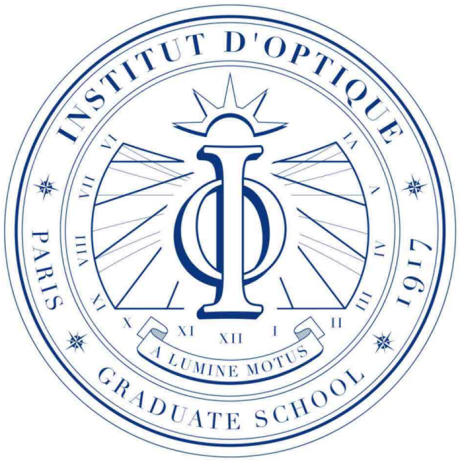
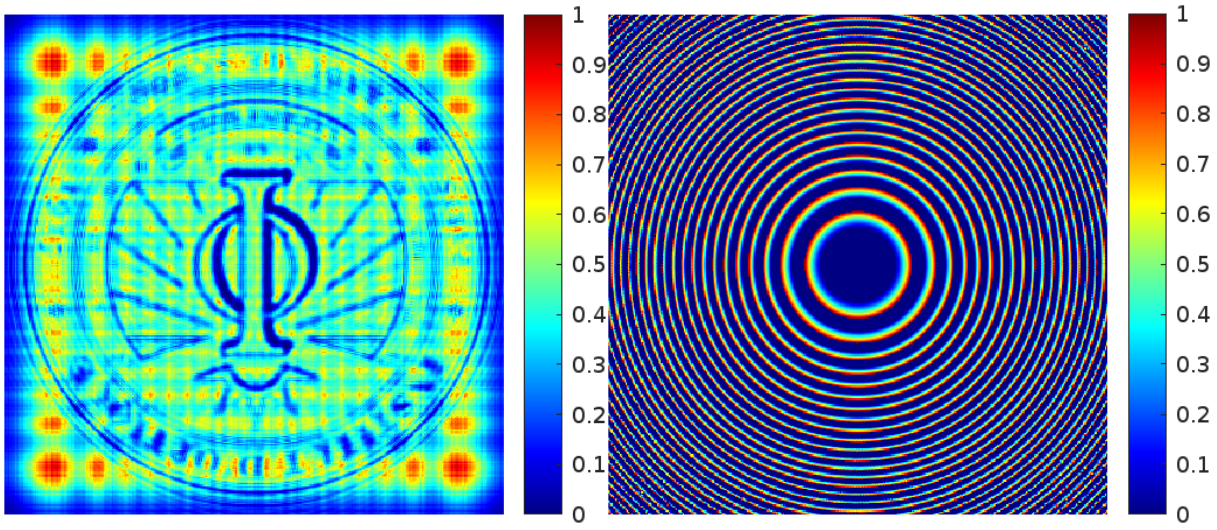


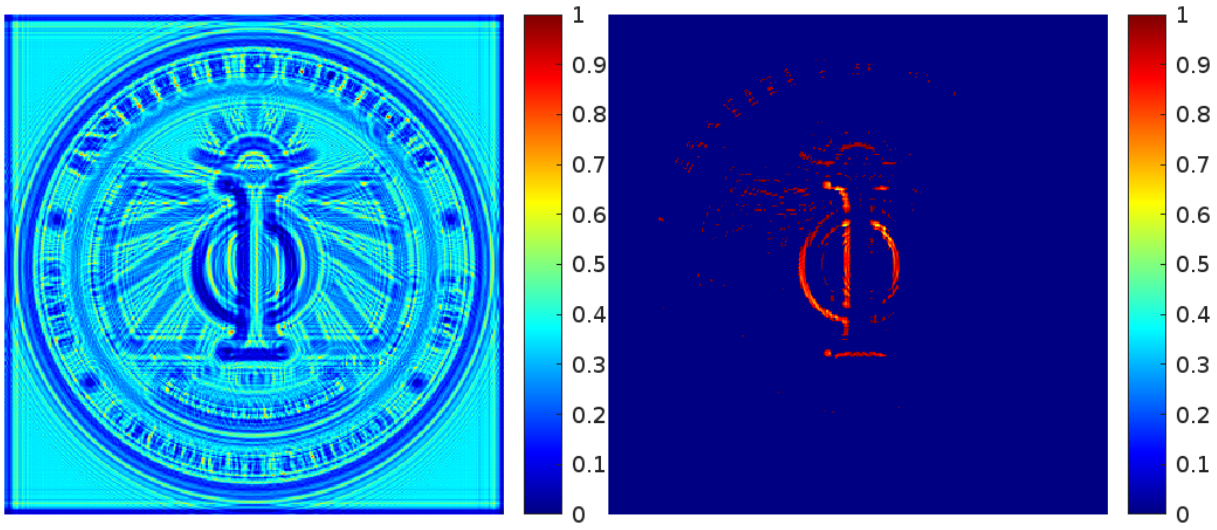
FIGURE 27 – Supoptique Logo

Here is the images we obtain by using the [C] code :



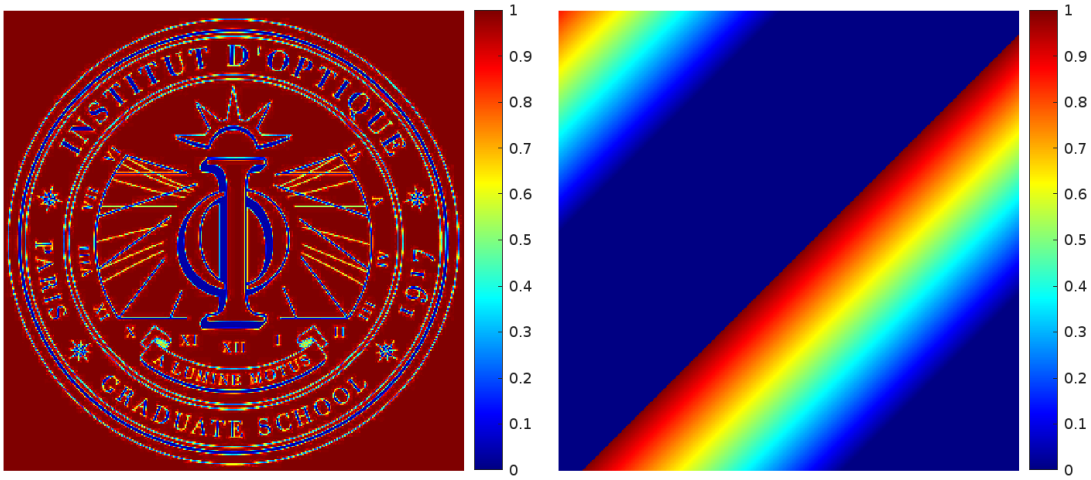
(a) Modulus (b) Phase

FIGURE 28 – Fresnel transforms : far field propagating fields, lens with a focal length of $f = 8\,\mu$m at $16\,\mu$m from both the object and the image



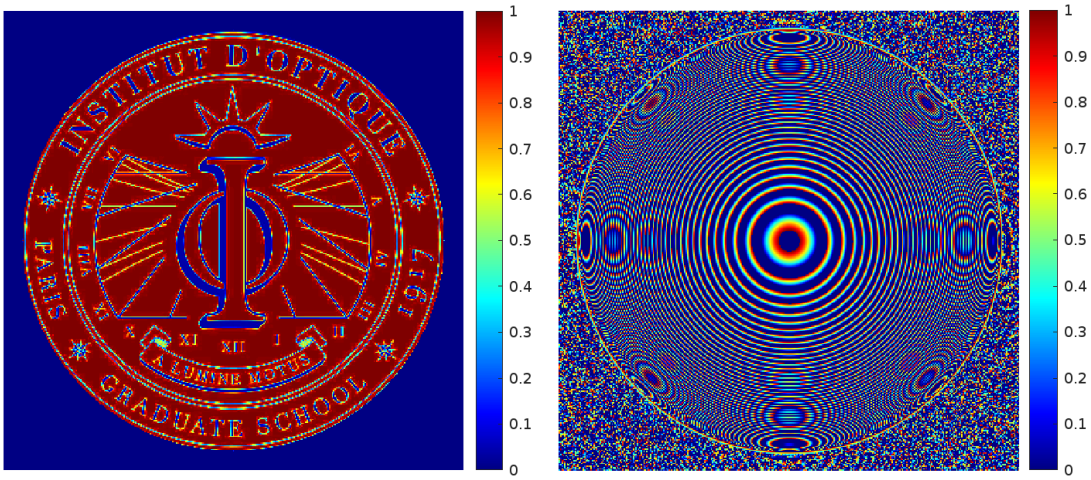
(a) Modulus (b) Phase

FIGURE 29 – Canonical transforms : far field propagating fields, lens with a focal length of $f = 8\,\mu$m at $16\,\mu$m from both the object and the image



(a) Modulus (b) Phase

FIGURE 30 – Fresnel propagation using Helmholtz method : far field propagating fields, lens with a focal length of $f = 8\,\mu$m at $16\,\mu$m from both the object and the image



(a) Modulus (b) Phase

FIGURE 31 – Fraunhofer propagation using Helmholtz method : far field propagating fields, lens with a focal length of $f = 8\,\mu$m at $16\,\mu$m from both the object and the image

In summary, the choice of propagation method and the precise positioning of the lens relative to the object and image planes critically influence the clarity, resolution, and fidelity of the projected images. The Helmholtz method, particularly with optimal lens placement, shows superior performance in maintaining the integrity and sharpness of the Supoptique logo in the far field projection.

6 Conclusion

In summary, this project allowed us to discover meta surfaces and further specialize on meta lenses. Every step was required to comprehend the work we did, and the end product truly satisfied us. It was beneficial to work with Bruno and Benjamin at the beginning to avoid starting off alone. Making a lot of Matlab code at last was a great task.

A Matlab codes Step 1

```
% Création de figures reliants les index n et k à la longueur d'onde ou à l'énergie pour des angles d'incidences pouvant changer
close all
% Constantes
h = 6.63e-34;
c = 3.00e8;
e = 1.6e-19;
%chargement des données
load(('Si_Palik.txt'),'ascii');%changer Si_Palik avec SiO2 quand necessaire
lambda=Si_Palik(:,1);%nm
E=h.*c./e./lambda;%eV
n=Si_Palik(:,2);
k=Si_Palik(:,3);
figure;
plot(lambda/1000,Si_Palik(:,2))%micromètre
hold on
plot(lambda/1000,Si_Palik(:,3))%micromètre
xlabel('Lambda (um)'),ylabel('n, k[1]');
legend('n','k');
figure;
plot(E,Si_Palik(:,2))%eV
hold on
plot(E,Si_Palik(:,3))%eV
xlabel('Photon Energy (eV)'),ylabel('n, k[1]');
legend('n','k');
theta_=deg2rad(1:10:40)
icol=0;for  theta=theta_;icol=icol+1 %degré

z=n.^2-k.^2-sin(theta).^2;
a=sqrt(z.^2+4*n.^2.*k.^2);

rs=((sqrt(a+z)-sqrt(2)*cos(theta)).^2+a-z)./((sqrt(a+z)+sqrt(2)*cos(theta)).^2+a-z);
rp=rs.*((sqrt(a+z)-sqrt(2)*sin(theta)*tan(theta)).^2+a-z)./((sqrt(a+z)+sqrt(2)*sin(theta)*tan(theta)).^2+a-z);

figure(3300)
h=plot(lambda,rs,'--');set(h,'linewidth',3)
set(h,'color',[icol/max(size(theta_)) 1-icol/max(size(theta_)) 0])
hold on
h=plot(lambda,rp,'-');set(h,'linewidth',3)
set(h,'color',[icol/max(size(theta_)) 1-icol/max(size(theta_)) 0])

xlabel('Lambda (um)'),ylabel('RS, RP (a.u.)');
legend('RS','RP');
title('RS and RP fonction of lambda with a step of 10 degree from 1 to 40')

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This test aims to launch RCWA 3D solver on simple layered system
format longE;
clear;
```

```
addpath(genpath('/MATLAB Drive/pcode'));
% Number of mode in fourier space (MANDATORY)
% Recommended : 18
OPTION.halfnpw = 18;
% TM or TE mode
% TM=0 -> TE mode.
% TM=1 -> TM mode.
OPTION.TM = 1;
% Transverse resolution
OPTION.FIELDnx = 100;  % Number of dots in x direction, to reconstruct E and H vector fields
OPTION.FIELDny = 100;  % Number of dots in y direction, to reconstruct E and H vector fields
OPTION.FIELD = 0;  % Plotting fields

% Number of frequency (you can defime om as a frequency range)

numom = 1;
om = 1e3*1.24/940;  % photon energy (hbar omega) in  eV
numom = 100;
if OPTION.halfnpw>0;numom = 10;end;
om = linspace(1e3*1.24/1000,1e3*1.24/250 ,numom); % photon energy (hbar omega) in  eV
lambdaall = linspace(.3,.94 ,numom); % photon energy (hbar omega) in  eV

% monitor: layer index to measure the phase (with 0 being the lower one)

monitor = 0;

% More parameters
% Incident light angle (NEVER SET ZERO)

theta = 0;  % Angle with z-axis
phi = 0;  % Angle with x-axis

% Visualization options
OPTION.SlideType = ['x', 'z'];  % Type of slices.
OPTION.SlideValue = [0 ,  -0.05];  % value of slices.
OPTION.display_colormap = 1;  % 1 -> Display colormap on plots of fields.
        OPTION.BEAM.TYPE='PW';
      %OPTION.BEAM.TYPE='GA';OPTION.OPTION.BEAM.PARAM=[.05 0 0];
disp(" ");
disp("STARTING calculation");
disp(" ");
warning off
Nb_sim=1
% Cycle over angles
for j = 1:Nb_sim
    disp(' ')
    disp(['Step: ', num2str(j), '/', num2str(Nb_sim)])
    %parfor stuff
    Lambda_all =zeros(1,numom);RR = zeros(1,numom);TT = zeros(1,numom);AA = zeros(1,numom);
    % RCWA Solving : cycle over frequency
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for i = 1:size(lambdaall, 2)
        ARG= struct;
```

```
        % Wavelength
        ARG.lambda =lambdaall(i);% 1.24/om(i);  % (um)
        ARG.t = j;
        % Size of the simulated region
        ARG.ax = 1.7;
      %  ARG.ax = 2.36;
        ARG.ax = 2.5;
     %   ARG.ax = 20;
        ARG.Material_Sub='silicon';
       % ARG.Material_Sub='SiO2';
% Multi coating
%ARG.d_ML1=0.06; ARG.Mat_ML1='SiO2_Palik';
%ARG.d_ML1=0.08; ARG.Mat_ML1='NewTiO2';
%ARG.d_ML2=0.03;ARG.Mat_ML2='NewTiO2';
%ARG.d_ML3=0.013;ARG.Mat_ML3='Al2O3_refractiveindex';
%DSA
      ARG.DSA='y';ARG.DSA_Thickness=0.07;
        % DEVICE creation
        DEVICE = struct_LayerCoating(ARG);
% Visualization
% OPTION.SlideType = [ 'z'];  % Type of slices.
% OPTION.SlideValue = [   -0.05];  % value of slices.
%  visualization_structure3D(DEVICE, OPTION);
if 0==1
       figure
                OPTION_3D=OPTION;
                OPTION_3D.SlideType=[ ];          % type of slides.
                OPTION_3D.SlideValue=[ ];      % value of slides.
                OPTION_3D.PlotEdge='no';
                OPTION_3D.PlotEdge='yes';   OPTION_3D.Plotfast='on';
   OPTION_3D.epsilon_Visible='no';
                visualization_structure3D_Perpective(DEVICE, OPTION_3D,
   get(gcf,'number'));
                set(gca,'ZLim',[-1 8])
                %remove air  Visualisation3D_Remove(get(gcf,'number'),[
   1 0 0]);set(gca,'zlim',[-3 1])
end
%         % RCWA3D on frequency om(i)
        SOL = RCWA_3D_singlecore(DEVICE, ARG.lambda, theta, phi, OPTION)
   ;
        if OPTION.FIELD==1
           OPTIONPF=OPTION; OPTIONPF.Type='n'
            PlotField3D(SOL, DEVICE, OPTIONPF);
        end
      %  ModalPhase(j, :) = SOL.ModalPhase;
        disp(['  Step duration: ' num2str(round(SOL.Time0_All/60, 2)) '
   min'])
        % Retrieving result
        Lambda_all(i) =ARG.lambda;
        RR(i) = SOL.RR;
        TT(i) = SOL.TT;
        AA_  = SOL.AA;
        AA(i) = AA_(end);
    end
end
figure(2)
hold on
h=plot(Lambda_all,100*TT,'.-b')
```

```
h=plot(Lambda_all,100*AA,'.-r')
%set(h,'DisplayName',[': ' num2str(OPTION.halfnpw)]);
hold on
h=plot(Lambda_all,100*RR,'.-g')
xlabel('Lambda (um)')
ylabel('T, A ,R (%)')
legend('T',' A' ,'R')
box on
set(gcf,'color',[1 1 1])
figure(20)
hold on
h=plot(Lambda_all,TT,'.-b')
h=plot(Lambda_all,AA,'.-r')
%set(h,'DisplayName',[': ' num2str(OPTION.halfnpw)]);
hold on
h=plot(Lambda_all,RR,'.-g')
xlabel('Lambda (um)')
ylabel('T, A ,R (1)')
legend('T',' A' ,'R')
box on
set(gcf,'color',[1 1 1])
%%
% Air to Glass vs angle
clear Theta_all TT AA RR ARG
for TM = 0:1;
    OPTION.TM=TM;
i=0;for theta=0:90;i=i+1;
    ARG.lambda = 0.5;  % (um)
    ARG.t = j;
    % Size of the simulated region
    ARG.ax = 10;%ARG.lambda * 1 / 2;  % (um)
    ARG.Material_Sub='silicon';
   % ARG.Material_Sub='si02';
    % DEVICE creation
    DEVICE = struct_LayerCoating(ARG);
    % Visualization
    %  visualization_structure3D(DEVICE, OPTION);
    % RCWA3D on frequency om(i)
    %OPTION.BEAM.TYPE='GA'
    OPTION.BEAM.TYPE='PW';
    OPTION.OPTION.BEAM.PARAM=[.1 0 0];
    SOL = RCWA_3D_singlecore(DEVICE, ARG.lambda, theta, phi, OPTION);
    % Retrieving result
    Theta_all(i) =theta;
    RR(i) = SOL.RR; TT(i) = SOL.TT;  AA_  = SOL.AA;
   % AAall(i) = sum(AA_(2:end));
   % AA(i) = AA_(L);
end
figure(3)
hold on
h=plot(Theta_all,100*TT,'.-b')
if OPTION.TM==1;set(h,'LineWidth',3);end
h=plot(Theta_all,100*RR,'.-g')
if OPTION.TM==1;set(h,'linewidth',3);end
end
xlabel('Theta  (deg)')
ylabel('T, R (%)')
legend('T TE','R TE','T TM','R TM' )
```

```
box on
set(gcf,'color',[1 1 1])
```

B Matlab codes Step 2 and 3

```
%% Conception of the pinhole
diameter_pinhole = 4; % um
lambda = 0.940e-6/sqrt(12.95); % m,
pp = 0.05*1e-6; % m
L = 512; C = 512;
Dp = diameter_pinhole * 1e-6 / pp;
U0 = zeros(L, C);
[x, y] = meshgrid((-C/2):(C/2-1), (-L/2):(L/2-1));
rho = sqrt(x.^2 + y.^2);
U0(rho < (Dp / 2)) = 1;
x_m = x * pp;
y_m = y * pp;
figure
hsurfaces = surface(x_m, y_m, U0);
set(hsurfaces, 'FaceColor', 'interp', 'EdgeColor', 'none');
set(gcf, 'color', [1 1 1]);
daspect([1 1 1]);
colorbar;
colormap(jet);
xlabel('x (m)');
ylabel('y (m)');
title('Image of a pinhole');

%% Propagation at z=10 um
d_prime = 10e-6;
    % Using the Fresnel method :
Ap_fresnel = Propagation_zero_padd(U0, lambda, d_prime, pp, 1, 0);
imshow(abs(Ap_fresnel).^2);
    % Using the convolutional method :
Ap_fr_conv = Propagation_zero_padd_conv(U0, lambda, d_prime, pp);
imshow(abs(Ap_fr_conv).^2);

%% Near field domain
    % First, we redo the pinhole with the right diameter:
diameter_pinhole = 8; % um

lambda = 0.940e-6; % m,
pp = 0.05*1e-6; % m
w = 0;
L = 512/2; C = L;
Dp = diameter_pinhole * 1e-6 / pp;
    % pinhole
U0 = zeros(L, C);
[x, y] = meshgrid((-C/2):(C/2-1), (-L/2):(L/2-1));
rho = sqrt(x.^2 + y.^2);
U0(rho < (Dp / 2)) = 1;
t_a_prime = U0;
    % For z <= 10  m
d_prime_values = linspace(1e-6, 10e-6, 1000);
    % For z <=  60  m
%d_prime_values = linspace(1e-6, 60e-6, 1000);
```

```
phase_circles = ones(L, length(d_prime_values));
for i = 1:length(d_prime_values)
    d_prime = d_prime_values(i);
        % With zero-padd
    Ap = Propagation_zero_padd(t_a_prime, lambda, d_prime, pp, 1, 0);
        % Without zero-padd
    %Ap = Propagation_zero_padd(t_a_prime, lambda, d_prime, pp, 0, 0);
    phase_circles(:, i) = abs(Ap(L/2, :)).^2;
end
phase_circles = phase_circles/max(phase_circles,[],"all");
y_pixels = ((1:L) - L/2) * pp * 1e6;
figure;
imagesc(d_prime_values * 1e6, y_pixels, phase_circles);
colormap(jet);
colorbar;
axis xy;
xlabel('d'' (um)');
ylabel('y (um)');

%% In plane phase shift after the lens
[x, y] = meshgrid((-C/2):(C/2-1), (-L/2):(L/2-1));
x_m = x * pp;
y_m = y * pp;
f = 8e-6;
t_a_prime=complex_transmittance(lambda,L, C, pp,f);
d_prime = 0;
Ap = Propagation_zero_padd(t_a_prime, lambda, d_prime, pp, 1, 0);
phase_circles = angle(Ap);
phase_circles = phase_circles/max(phase_circles,[],"all");
% imshow(phase_circles);
% colormap(jet);
figure
hsurfaces = surface(x_m, y_m, phase_circles);
set(hsurfaces, 'FaceColor', 'interp', 'EdgeColor', 'none');
set(gcf, 'color', [1 1 1]);
daspect([1 1 1]);
colorbar;
colormap(jet);
xlabel('x (m)');
ylabel('y (m)');
zlabel('Intensity');

%% Vertical central-cut of the propagating fields, focal length of f=8um
    at 940nm
L = 512; C = L;
U0 = ones(L, C);
[x, y] = meshgrid((-C/2):(C/2-1), (-L/2):(L/2-1));
x_m = x * pp;
y_m = y * pp;
t_a_prime = complex_transmittance(lambda, L, C, pp, f);
d_prime_values = linspace(1e-6, 16e-6, 1000); % Range of d_prime values
phase_circles = ones(L, length(d_prime_values));
for i = 1:length(d_prime_values)
    d_prime = d_prime_values(i);
    Ap = Propagation_zero_padd(t_a_prime, lambda, d_prime, pp, 1, 0);
        % Modulus :
    phase_circles(:, i) = abs(Ap(L/2, :)).^2;
        % Phase :
```

```
    % phase_circles(:, i) = angle(Ap(L/2, :));
end
phase_circles = phase_circles/max(phase_circles,[],"all");
y_pixels = ((1:L) - L/2) * pp * 1e6;
figure;
imagesc(d_prime_values * 1e6, y_pixels, phase_circles);
colormap(jet);
colorbar;
axis xy;
xlabel('d'' (um)');
ylabel('y (um)');

%% Helmholtz method: vertical central-cut at the focal point
clear all
lambda = 0.940e-6; %m
C = 256;
L = C;
k = 2*pi/lambda;
pp = 0.05e-6; %m
[x, y] = meshgrid((-C/2):(C/2-1), (-L/2):(L/2-1));
x_m = x * pp;
y_m = y * pp;
f = 8e-6;
t_a_prime=complex_transmittance(lambda,L, C, pp,f);
d_prime = f;
[UH_RS,UH_F,UH_Fraun] = propagation_H(t_a_prime,lambda,d_prime,pp);
phase_circles = angle(UH_F);
phase_circles = phase_circles/max(phase_circles,[],"all");
figure
y_pixels = ((1:L) - L/2) * pp * 1e6; % Conversion de l'index des pixels
   en micromètres
figure;
imagesc(y_pixels, y_pixels, phase_circles); % Conversion de m en  m
   pour l'affichage
colormap(jet);
colorbar;
axis xy;
xlabel('d'' ( m )');
ylabel('y ( m )');


%%
%%
%%
%%
%% Functions
    % Fresnel propagation
function Ap=Propagation_zero_padd(tim,lambda,z,pp,zero_padd,w)
% Usage :Ap=PropagationZP(tim,lambda,z,pp,zero_padd,w);
% Goal : Simulation of free propagation of coherent light from the plane
    1 of transmittance tim to a plane 2 distant of
% z. Depending on the z value the program uses impulse response hz or
   transfer function Hz
% Note : the uniform phase term exp(-i*2*pi*z:lambda) can be considered
   to be equal to 1 without changing the results in some cases.
% input parameters :
% tim : transmittance of the plane
% lambda : wavelength
```

```
% z : propagation distance along the optical axis
% pp : pixel pitch
% zero_padd
% if zero_pad=0 no zero padding
% if zero_pad=1 zero padding and output image of size equal to the input
% image
% if zero_pad=2 zero padding and output image of size equal to two times
    the
% input image size
% w=0, if we do not want warnings to be written in the command window
% w=1, warnings are written in the command window

[L,C]=size(tim);
if L~=C
    if w==1,fprintf('columns number is not equal to rows number \n
   Warning : to select the method "impulse response" or "transfer
   function" the number of rows is used\n'),end
end
zNS=L*(pp^2)/lambda;% z0 is computed without zero padding because in
   that case what interests us is the in field of view reconstruction

if zero_padd==0
        Hz=FresnelFunctionFT(L,C,z,lambda,pp);% Matlab coordinate origin
        if w>=1,fprintf('use of the transfer function for the
   propagation computation\n'), end
        fftt=fft2(tim);% % Matlab coordinate origin
        Ap=ifft2(Hz.*fftt);
else % (zero_padd>=1)

    tim2=ones(2*L,2*C)*mean(tim(:));tim2(L-L/2:L+L/2-1,C-C/2:C+C/2-1)=
   tim;
    if w==2,figure;imshow(tim2,[],'XData',[1:size(tim2,2)]*1000*pp,'
   YData',[1:size(tim2,1)]*1000*pp);colorbar;title(' 0 padded image');
   axis on;xlabel('x(mm)');ylabel('y(mm)'); end
        Hz=FresnelFunctionFT(2*L,2*C,z,lambda,pp);
        if w>=1,fprintf('use of the transfer function for the
   propagation computation\n'), end
    end
    fftt=fft2(tim2);
    Ap0=ifft2(Hz.*fftt);

    if zero_padd==1
        Ap=Ap0(L-L/2:L+L/2-1,C-C/2:C+C/2-1);
    else
        Ap=Ap0;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    function hz=FresnelFunction(L,C,z,lambda,pp);
        %% Fresnel function
        % input parameters :
        % L,C : number of lines and columns of the Fresnel function (
   output parameter)
        % z : distance of propagation in meter
        % lambda : wavelength in meter
        % pp : pixel pitch in meter
        % output parameter :
        % hz : matrix of L lines and C columns of complexe values of the
```

```
    Fresnel function
210
211         %% Usage : hz=FresnelFunction(L,C,z,lambda,pp);
212         hz=zeros(L,C);
213         [Ki,Et]=meshgrid((-floor(C/2):ceil(C/2)-1),(-floor(L/2):ceil(L
    /2)-1)); % units in pix-1
214         Ki=Ki*pp;% units in m
215         Et=Et*pp;% units in m
216         c1=1/lambda/z/1i;
217         c2=1i*pi/lambda/z;
218         hz=ifftshift(exp(1i*2*pi*z/lambda)*c1*exp(c2*( Ki.^2+Et.^2)  ));
    % the phase term exp(i*2*pi*z/lambda  ) can be omited in some cases
219         % the phase c2*( Ki.^2+Et.^2)  ) is positive , it is the sign
    convention considered in the lectures
220
221     end
222 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
223     function Hz=FresnelFunctionFT(L,C,z,lambda,pp);
224         %% Fresnel function Fourier transform
225         % input parameters :
226         % L,C : number of lines and columns of the Fresnel function (
    output parameter)
227         % z : distance of propagation in meter
228         % lambda : wavelength in meter
229         % pp : pixel pitch in meter
230         % output parameter :
231         % Hz : matrix of L lines and C columns of complexe values of the
     Fresnel
232         % function Fourier Transform
233
234         %% Usage : Hz=FonctionDeFresnelFT(L,C,z,lambda,pp);
235
236         Hz=zeros(L,C);
237         [u,v]=meshgrid((-floor(C/2):ceil(C/2)-1)/C,(-floor(L/2):ceil(L
    /2)-1)/L);
238         % for images with an even number of columns and rows, the
    geometrical center is C/2+1,L/2+1
239         % for images with an odd number of columns and rows, the
    geometrical center
240         % is ceil (C/2+1),ceil(L/2+1)
241         % The frequencies varies from -0.5 to 0.5-frequency sampling
    step, There are LxC
242         % samples. The unit is pix-1
243         u=u/pp; % unit : m-1
244         v=v/pp; % unit : m-1
245
246         ipilambdaz=-i*pi*lambda*z;
247         Hz=ifftshift(exp(1i*2*pi*z/lambda)*exp(ipilambdaz*( u.^2+v.^2) )
    );
248     end
249 end
250
251 % Fresnel propagtion by using the convolutional method
252 function Ap_Fc =Propagation_zero_padd_conv(tim,lambda,z,pp);
253 % Propagation selon la méthode de Fresnel sous sa forme canonique
254
255 [L,C] = size(tim);
256 Ap_Fc = zeros(L,C);
```

```
[X,Y] = meshgrid((-floor(C/2):ceil(C/2)-1),(-floor(L/2):ceil(L/2)-1));
Xp = X/(C);
Yp = Y/(C);
X = X.*pp;
Y = Y.*pp;
Xp = Xp.*pp.*lambda.*z;
Yp = Yp.*pp.*lambda.*z;
c1 = (1/lambda/z/1i).*exp(1i*2*pi*z/lambda);
c2 = 1i.*pi/lambda/z;
%Ap_Fc = c1.*exp(c2.*(Xp.^2 + Yp.^2)).*fftshift(fft2(tim.*exp(c2.*(X.^2
    + Y.^2))));
Ap_Fc = ifftshift(c1 .* exp(c2 .* (X.^2 + Y.^2))) .* fft2(tim .* exp(c2
    .* (X.^2 + Y.^2)));
Ap_Fc = ifft2(Ap_Fc);
end

% Lens
function tL = complex_transmittance(lambda, L, C, pp, f)
    X = -C/2:C/2-1;
    Y = -L/2:L/2-1;
    [x, y] = meshgrid(X, Y);
    Delta = (x.^2 + y.^2) .* (pp^2 / (2*f));
    Phi = (2*pi/lambda) .* Delta;
    tL = exp(-1i .* Phi);
end

%% Propagation using Helmholtz method :
function [ M ] = MaskDisk( L,C,Dp,pp);
% Usage :[ M ] = MaskDisk( L,C,Dp,pp );
% Goal : Create a binary mask of size LxC with a disk of diameter D.
% Its center is located in (L/2+1,C/2+1) if L,C are even.
% Input Parameters :
% - Dp : Disk diameter (type : double, scalar, unit=meter)
% - L,C : size of the mask (type : double, scalar, unit=pixel)
% Output Parameters :
% - M : Mask (type : double, size : LxC)
% pp : pixel pitch

x=-C/2:C/2-1;
y=-L/2:L/2-1;
x=x.*pp;y=y.*pp;
[X,Y]=meshgrid(x,y);
rho=sqrt(X.^2+Y.^2);
M=double(rho<(Dp/2));
end

function [Ap_Fc]=propagation_Fc(tim,lambda,z,pp);
% Propagation selon la mÃ thode  de Fresnel sous sa forme canonique

[L,C] = size(tim);
Ap_Fc = zeros(L,C);
[X,Y] = meshgrid((-floor(C/2):ceil(C/2)-1),(-floor(L/2):ceil(L/2)-1));
Xp = X/(C);
Yp = Y/(C);
X = X.*pp;
Y = Y.*pp;
Xp = Xp.*pp.*lambda.*z;
Yp = Yp.*pp.*lambda.*z;
```

```
c1 = (1/lambda/z/1i).*exp(1i*2*pi*z/lambda);
c2 = 1i.*pi/lambda/z;
Ap_Fc = c1.*exp(c2.*(Xp.^2 + Yp.^2)).*fftshift(fft2(tim.*exp(c2.*(X.^2 +
    Y.^2))));
end
function [Ap_H_RS,Ap_H_F,Ap_H_Fraun]=propagation_H(tim,lambda,z,pp)
% Propagation selon la mÃ thode  de Helmholtz

[L,C] = size(tim);
Ap_H_RS = zeros(L,C);
Ap_H_F = zeros(L,C);
Ap_H_Fraun = zeros(L,C);
[X,Y] = meshgrid((-floor(C/2):ceil(C/2)-1),(-floor(L/2):ceil(L/2)-1));
X = X.*pp;
Y = Y.*pp;
Xp = X/(C);
Yp = Y/(C);
Xp = Xp.*pp.*lambda.*z;
Yp = Yp.*pp.*lambda.*z;

% DÃ finition  de la solution des Ã quations  d'Helmholtz

k0 = 2*pi/lambda;
kx = (2*pi/lambda).*(X./z);
ky = (2*pi/lambda).*(Y./z);
EH = tim.*exp(1i.*z.*sqrt(k0.^2 - kx.^2 - ky.^2));

% propagation suivant les diffÃ rentes  mÃ thodes  :

% Rayleigh Sommerfeld

R = sqrt(X.^2+Y.^2+z.^2);
c1 = z./(lambda*1i*(R.^2));
c2 = 1i*2*pi/lambda;
hzRS = ifftshift(c1.*exp(c2.*R));
TF_RS = fft2(hzRS);
Ap_H_RS = ifft2(fft2(EH).*TF_RS);

% Fresnel

% c3 = (1 / (lambda * z * 1i)) .* exp(1i * 2 * pi * z / lambda);
% c4 = 1i * pi / (lambda * z);
% pre_transform = tim .* exp(c4 .* (X.^2 + Y.^2));
% hzF = fftshift(fft2(pre_transform));
% Ap_H_F = c3 .* ifft2(hzF);
 c3 = (1 / (lambda * z * 1i)) .* exp(1i * 2 * pi * z / lambda);  %
    Common phase factor
c4 = 1i * pi / (lambda * z);  % Scaling for the quadratic phase factor
pre_transform = EH .* exp(c4 .* (X.^2 + Y.^2));  % Modifying the input
    field with quadratic phase
hzF = fftshift(fft2(pre_transform));  % Applying fft to the phase-
    modified field
% Ap_H_F = c3 .* ifft2(hzF);  % Applying inverse fft and scaling with c3
Ap_H_F = ifft2(hzF);  % Applying inverse fft and scaling with c3

% Fraunhofer

c5 = (1/lambda/z/1i).*exp(1i*2*pi*z/lambda);
```

```
367 c6 = 1i.*pi/lambda/z;
368 Ap_H_Fraun = ifftshift(c5.*exp(c6.*(Xp.^2 + Yp.^2)).*fftshift(fft2(EH)))
    ;
369 Ap_H_Fraun = ifft2(Ap_H_Fraun);
370 end
371
372 function [Ap_F]=propagation_F(tim,lambda,z,pp);
373 % Propagation selon la mÃ thode  de Fresnel (approximation de la mÃ
     thode  de Rayleigh Sommerfeld)
374
375 [L,C] = size(tim);
376 hzF = zeros(L,C);
377 [X,Y] = meshgrid((-floor(C/2):ceil(C/2)-1),(-floor(L/2):ceil(L/2)-1));
378 X = X.*pp;
379 Y = Y.*pp;
380 c1 = (1/lambda/z/1i).*exp(1i*2*pi*z/lambda);
381 c2 = 1i.*pi/lambda/z;
382 hzF = ifftshift(c1.*exp(c2.*(X.^2 + Y.^2)));
383 TF_F = fft2(hzF);
384 Ap_F = ifft2(fft2(tim).*TF_F);
385 end
386 function [Ap_Fraun]=propagation_Fraun(tim,lambda,z,pp)
387
388 [L,C] = size(tim);
389
390 Ap_Fraun = zeros(L,C);
391 [X,Y] = meshgrid((-floor(C/2):ceil(C/2)-1),(-floor(L/2):ceil(L/2)-1));
392 Xp = X/(C);
393 Yp = Y/(C);
394 X = X.*pp;
395 Y = Y.*pp;
396 Xp = Xp.*pp.*lambda.*z;
397 Yp = Yp.*pp.*lambda.*z;
398 c1 = (1/lambda/z/1i).*exp(1i*2*pi*z/lambda);
399 c2 = 1i.*pi/lambda/z;
400 Ap_Fraun = c1.*exp(c2.*(Xp.^2 + Yp.^2)).*fftshift(fft2(tim));
401 end
402
403 function [Ap_RS]=propagation_RS(tim,lambda,z,pp);
404 % Propagation selon la mÃ thode  de Rayleigh Sommerfeld
405
406 [L,C] = size(tim);
407 hzRS = zeros(L,C);
408 [X,Y] = meshgrid((-floor(C/2):ceil(C/2)-1),(-floor(L/2):ceil(L/2)-1));
409 X = X.*pp;
410 Y = Y.*pp;
411 R = sqrt(X.^2+Y.^2+z.^2);
412 c1 = z./(lambda*1i*(R.^2));
413 c2 = 1i*2*pi/lambda;
414 hzRS = ifftshift(c1.*exp(c2.*R));
415 TF_RS = fft2(hzRS);
416 Ap_RS = ifft2(fft2(tim).*TF_RS);
417 end
```

C Matlab codes Step 4

```
1
2  %% Propa logo supop
3
4  clear all
5  close all
6
7  lambda = 0.940e-6; %m
8  k = 2*pi/lambda;
9  pp = 0.05e-6; %m
10
11
12 tim =imread('Logo Supop.png');
13 tim = rgb2gray(tim);
14 % imagesc(tim)
15
16 tim = imresize(tim, [699 699]);
17 imshow(tim);
18
19 [L,C] = size(tim);
20 [x, y] = meshgrid((-C/2):(C/2-1), (-L/2):(L/2-1));
21 x_m = x * pp;
22 y_m = y * pp;
23 f = 8e-6;
24 d_prime = 2*f;
25 Dp = 600;
26
27 tim_propa=Propagation_zero_padd(tim, lambda, d_prime, pp, 1, 0);
28 % tim_propa = Propagation_zero_padd_conv(tim, lambda, d_prime, pp);
29 tim = double(tim);
30
31 tL=complex_transmittance(lambda,L, C, pp,f);
32 t_lens_transmited=tim_propa.*tL;
33 t_lens=t_lens_transmited;
34
35 Ap_propa=Propagation_zero_padd(t_lens, lambda, d_prime, pp, 1, 0);
36 % Ap_propa = Propagation_zero_padd_conv(t_lens, lambda, d_prime, pp);
37
38 phase_circles = abs(Ap_propa).^2;
39 % phase_circles = angle(Ap_propa);
40 phase_circles = phase_circles/max(phase_circles,[],"all");
41
42 imshow(phase_circles);
43 colormap(jet); colorbar;
44
45
46 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
47
48 function [Layers, current_z, ind_current_layer,nz]=ADD_STructured_Layer2
       (Layers, current_z, ind_current_layer,nz,epsSi,epsSiO2,ALLcenter,
       ALLcentery,ARG)
49 % A layer of structuration
50 % Parameters:
51 % into ARG
52 % Mandatory PARAMETERS:
53 % % definition of the centers of all pillars as rectanglura grid:
54 %ALLcenter=[-1 0 1] ;
55 %ALLcenter=[ 0 ] ;
56 %ALLcentery= ALLcenter ;
```

```
%OR like a list !

Nduplicate=(size(ALLcenter,1));

if ~isempty(ALLcentery)

    Nduplicate=max(size(ALLcenter));
    Nduplicatey=max(size(ALLcentery)); %ALLcentery is a 1 vector. We are daling with rectangular pillar grids
    for icenterx= 1:Nduplicate
        for icentery= 1:Nduplicatey
            %    centrecercle=[x_min + (x_max - x_min)/2, y_min + (y_max - y_min)/2 ]+[ALLcenter(icenterx), ALLcenter(icentery)];
            centrecercle(icentery,icenterx,:)=[ALLcenter(icenterx), ALLcentery(icentery)];
        end
    end
else
    Nduplicatey=1; % provide pillar position and other parameters as a list
    for icenterx= 1:Nduplicate
        %    centrecercle=[x_min + (x_max - x_min)/2, y_min + (y_max - y_min)/2 ]+[ALLcenter(icenterx), ALLcenter(icentery)];
        centrecercle(1,icenterx,:)=[ALLcenter(icenterx,1), ALLcenter(icenterx,2)];
    end
end

if isfield(ARG,'Pillar_Rotation'); Pillar_Rotation=ARG.Pillar_Rotation;
else; Pillar_Rotation=zeros(Nduplicatey,Nduplicate);
end   %rotate the pillars (matrix of rotation)

if isfield(ARG,'FINERGRID'); FINERGRID=ARG.FINERGRID;
else; FINERGRID=1;
end   %not used but do not remove

%other parameters (default if not spacified)
% Pillar width of the bottom circle of the pillar (the largest). The light is comming from the top
% if Pillar width2 not specified it is egual to width

if isfield(ARG,'PillarWidth'); PillarWidth=ARG.PillarWidth;else; PillarWidth=zeros(Nduplicatey,Nduplicate)+.2; end
if isfield(ARG,'PillarWidth2'); PillarWidth2=ARG.PillarWidth2;else; PillarWidth2=zeros(Nduplicatey,Nduplicate)+.2; end

% SPECIFIC Pillars

if isfield(ARG,'Height_pillar'); Height_pillar=ARG.Height_pillar;else; Height_pillar=0.5;     end
if isfield(ARG,'angle_pillar'); angle_pillar=ARG.angle_pillar;else; angle_pillar=0+zeros(Nduplicate,Nduplicate);     end

%deg / SLicing vertical des pilliers (number of cuts)

if isfield(ARG,'Npillar');
    Npillar=ARG.Npillar;
else;
```

```
104     if min(min(angle_pillar))>0
105         Npillar=20;  else
106         Npillar=1;
107     end
108 end
109 
110 Top_Pillar_Overetch=0;
111  Bot_Pillar_Overetch=0;
112  N_overetch=0;
113 
114 if isfield(ARG,'Pillar_Overetch');
115     Top_Pillar_Overetch=ARG.Pillar_Overetch.dtop;
116     Bot_Pillar_Overetch=ARG.Pillar_Overetch.dbot;
    Top_Pillar_Overetch_eps=ARG.Pillar_Overetch.top_eps;
117     Bot_Pillar_Overetch_eps=ARG.Pillar_Overetch.bot_eps;
118     N_overetch=2;
119 
120     Npillar=Npillar+N_overetch; % add Two layers for the overetch
121 end   %overetch the pillars (material of the outer is different)
122 
123 %random parameters : h=.015,clx=.3,cly=1.5;
124 
125 if isfield(ARG,'ISRAND'); ISRAND=ARG.ISRAND;else; ISRAND=0;     end
126 if ISRAND==1;
127     if isfield(ARG,'RMS'); RMS=ARG.RMS;else; RMS=0.015;     end
128     if isfield(ARG,'Lradial'); Lradial=ARG.Lradial;else; Lradial=0.3;
     end
129     if isfield(ARG,'Lvertical'); Lvertical=ARG.Lvertical;else; Lvertical
    =1.5;     end
130     if Npillar<2
131         disp('Cannot have ISRAND with Npillar<2')
132         OUTPUT=-1;
133         return;
134     end
135 end
136 
137 if isfield(ARG,'PLOTALL'); PLOTALL=ARG.PLOTALL;else; PLOTALL=0;     end
138 if ~isempty(ALLcentery)
139 
140 %A snamity check: in case of a grid metasurf
141 
142     if max(size(PillarWidth,2)) ~=Nduplicate
143         disp('size(PillarWidth) does not match with ALLcenter definition
    ')
144         OUTPUT=-1;
145         return;
146     end
147 
148     if max(size(PillarWidth,1)) ~=Nduplicatey
149         disp('size(PillarWidth) does not match with ALLcenter definition
    ')
150         OUTPUT=-1;
151         return;
152     end
153 
154     if max(size(PillarWidth2,2)) ~=Nduplicate
155         disp('size(PillarWidth2) does not match with ALLcenter
    definition')
```

```
156         OUTPUT=-1;
157         return;
158     end
159 
160     if max(size(PillarWidth2,1)) ~=Nduplicatey
161         disp('size(PillarWidth2) does not match with ALLcenter
    definition')
162         OUTPUT=-1;
163         return;
164     end
165 end
166 
167 %create some variable for the parfor
168 
169 current_z_(1)=current_z;ind_current_layer_0=ind_current_layer;for
    iNpillar=1:(1+Npillar)
170 
171     if iNpillar<=N_overetch/2
172         d_(iNpillar) = Top_Pillar_Overetch;
173     elseif iNpillar>(1+Npillar-N_overetch/2)
174         d_(iNpillar) = Bot_Pillar_Overetch;
175     else
176         d_(iNpillar) =  (Height_pillar-Top_Pillar_Overetch-
    Bot_Pillar_Overetch)/(1+Npillar-N_overetch); %hauteur des pilliers
177     end
178     ind_current_layer = ind_current_layer + 1; nz=[nz round(100/(1+
    Npillar)) ];
179     current_z_(iNpillar+1) = current_z_(iNpillar) -d_(iNpillar);
180 end
181 
182 %for rand Pillars
183 
184 if ISRAND==1 %reported here for parfor reason
185     Widthextracted=[];Widthextracted_botom = [];Nshape=0; ALLf=[];
186     %if   strcmp(ARG.Pillar_type, "circleRand")
187     N=(1+Npillar);rL=1;% Warning corelation lenth are in rL units
188     for icenterx= 1:Nduplicate
189         for icentery= 1:Nduplicatey
190             % RO=PillarWidth(icenterx,icentery)/2; %/2 because it is a
    radius
191             %[f,x,y] =rsgene2D_Loumi(N,rL,RMS,Lradial*RO*2*pi,Lvertical*
    Height_pillar,Seeds_maps(icenterx,icentery));  %rescaled correlation
    lengh
192             [f,x,y] =rsgene2D(N,rL,RMS,Lradial,Lvertical*Height_pillar);
193             ALLf(icentery,icenterx,:,:)=f;
194         end
195     end
196 
197     Nshape=N; % DO NOT CHANGE Nshape=N !
198     %buid a circle as polynoms (radius is unity)
199     P=[];
200     for thetha_grid=1:Nshape
201         P=[P; [ cos(2*pi*thetha_grid/Nshape) sin(2*pi*thetha_grid/Nshape
    )]];
202     end
203 
204     if PLOTALL==1;figure(66);hold off;end
205     %Widthextracted=[];Widthextracted_botom = [];
```

```
    %end
end

%end rand Pillars
Layers_(1)=Layers(1);
for iNpillar=1:(1+Npillar)

    %eps_outpillar = epsSiO2;%outer material _pillier;
    if iNpillar<=N_overetch/2
        eps_outpillar =Top_Pillar_Overetch_eps; %overetch
    elseif iNpillar>(1+Npillar-N_overetch/2)
        eps_outpillar =Bot_Pillar_Overetch_eps;%overetch
    else
        eps_outpillar = epsSiO2;%outer material _pillier;
    end
    eps_square = eps_outpillar; %outer square
    Array_of_circle=[];Array_of_Polygon=[];
    %Pillar with side angle
    Array_of_circle=[];    Array_of_Polygon=[];
    if strcmp(ARG.Pillar_type, "perfectcircle")
        for icenterx= 1:Nduplicate
            for icentery= 1:Nduplicatey
                %centrecercle=[x_min + (x_max - x_min)/2, y_min + (y_max - y_min)/2 ]+[ALLcenter(icenterx), ALLcenter(icentery)];
                %centrecercle=[ALLcenter(icenterx), ALLcentery(icentery)];
                centrecercle_=[centrecercle(icentery,icenterx,1), centrecercle(icentery,icenterx,2)];
                radiusofcircle=(PillarWidth(icentery,icenterx)-2*Height_pillar*tan(2*pi/360*angle_pillar(icentery,icenterx))*((Npillar-iNpillar+1)/(Npillar+1)))/2;  %bug
                Array_of_circle = [Array_of_circle Disk_2D(radiusofcircle, centrecercle_,epsSi)];
            end
        end
    elseif strcmp(ARG.Pillar_type, "none")
        %do noting
    else
        for icenterx= 1:Nduplicate
            for icentery= 1:Nduplicatey
                if  strcmp(ARG.Pillar_type, "circle")
                    Theta=0:(2*pi)/51:2*pi;multfact= 1;
                elseif  strcmp(ARG.Pillar_type, "losange")
                    Theta=0:(pi/2):2*pi;multfact= 1;
                elseif  strcmp(ARG.Pillar_type, "hexagon")
                    Theta=0:(pi/3):2*pi;multfact= 1;
                elseif  strcmp(ARG.Pillar_type, "octogon")
                    Theta=0:(pi/4):2*pi; multfact= 1;
                elseif  strcmp(ARG.Pillar_type, "circSquare")
                    Pillar_circSquareangle= ARG.Pillar_circSquareangle(icentery,icenterx);
                    Theta_0=(pi/4):(pi/2):(2*pi);
                    cutangle=-pi/Pillar_circSquareangle:(pi/Pillar_circSquareangle)/11:pi/Pillar_circSquareangle;
                    Theta=[];for icutangle=1:max(size(Theta_0))
                        Theta=[Theta Theta_0(icutangle)+cutangle];
                    end
                    multfact= 1/cos(pi/4-pi/Pillar_circSquareangle); %
```

```
    increase the radius to get the square part at the good dimention
    givent by pillarwidth
256                 end
257                 %    centrecercle=[x_min + (x_max - x_min)/2, y_min + (
    y_max - y_min)/2 ]+[ALLcenter(icenterx), ALLcentery(icentery)];
258                 %   centrecercle=[ALLcenter(icenterx), ALLcentery(
    icentery)];
259                 centrecercle_=[centrecercle(icentery,icenterx,1),
    centrecercle(icentery,icenterx,2)];
260                 radiusofcircle=(PillarWidth(icentery,icenterx)-2*
    Height_pillar*tan(2*pi/360*angle_pillar(icentery,icenterx))*((Npillar
    -iNpillar+1)/(Npillar+1)))/2; %bug
261                 % Building the radius:
262                 RRR = [];
263                 tmpR = 0;
264                 Nb_Period=ARG.Pillar_NbPeriod(icentery,icenterx);
265                 if Nb_Period>0
266                     PeriodSi = (ARG.PillarInnerWidth(icentery,icenterx)
    +2*Height_pillar*tan(2*pi/360*angle_pillar(icentery,icenterx))*((
    Npillar-iNpillar+1)/(Npillar+1)))/2;%bug
267                     PeriodSiO2 = PillarWidth(icentery,icenterx)/2-
    PeriodSi-Height_pillar*tan(2*pi/360*angle_pillar(icentery,icenterx))
    *((Npillar-iNpillar+1)/(Npillar+1));%bug
268                     for ir = 1:Nb_Period
269                         if ir == 1
270                             RRR = [RRR  (tmpR +PeriodSi)];
271                             tmpR = tmpR + PeriodSi;
272                             RRR = [RRR (tmpR + PeriodSiO2)];
273                             tmpR = tmpR + PeriodSiO2;
274                         else
275                             RRR = [RRR  (tmpR +PeriodSi)];
276                             tmpR = tmpR + PeriodSi;
277                             RRR = [RRR (tmpR + PeriodSiO2)];
278                             tmpR = tmpR + PeriodSiO2;
279                         end
280                     end
281                 else % juste one regular pillar
282                     RRR= radiusofcircle;
283                 end
284                 RRR = multfact*RRR;
285                 RRR2 = PillarWidth2(icentery,icenterx)/PillarWidth(
    icentery,icenterx)*RRR; %this is a rectangular shape
286                 for ir = 1:(max(size(RRR)))
287                     tmp = centrecercle_+[ RRR(ir)*cos(Theta)' RRR2(ir)*
    sin(Theta)'];
288                     if ISRAND==1
289                         Pm=zeros(Nshape,2);
290                         %linear interpolation of the shapes (prio to
    distortion)
291                         [pt,dudt,foft] = interparc(0.1,RRR(ir)*cos(Theta
    ),RRR2(ir)*sin(Theta),'linear')
292                         P=foft(linspace(0,1,Nshape));
293                         %distortion fiels
    Rescale_distortion_to_Perimeter=perimeter(polyshape(P));
294                         if iNpillar ==1                [distortion_field(
    icentery,icenterx,:,:)] =rsgene2D(N,rL,RMS,Lradial/
    Rescale_distortion_to_Perimeter,Lvertical/Height_pillar);
295                         end
```

```
296                         ishape=1;
297                         VP=[P(ishape,:)-P(end,:)];VP=[VP(2) -VP(1)];VP=
    VP/norm(VP);% normal direction
298                         Pm(ishape,:)= P(ishape,:)+ALLf(icentery,icenterx
    ,iNpillar,ishape)*VP;
299                         for ishape=2:Nshape
300                             VP=[P(ishape,:)-P(ishape-1,:)];VP=[VP(2) -VP
    (1)];VP=VP/norm(VP);% normal direction
301                             Pm(ishape,:)= P(ishape,:)+distortion_field(
    icentery,icenterx,iNpillar,ishape)*VP;
302                         end
303                         Pm(1,:)= 0.5*(Pm(2,:)+Pm(end,:));
304                         %plot(P(:,1),P(:,2),'.');hold on
305                         %plot(Pm(:,1),Pm(:,2),'.')
306                         tmp = centrecercle+Pm;
307                     end
308 %rotate pillars:
309 %polygoneC_coordinate = Rot2D(polygoneC_coordinate, pi/2, [0, 0]);
310                     if (Pillar_Rotation(icentery,icenterx))>0;
311                         tmp = Rot2D(tmp, Pillar_Rotation(icentery,
    icenterx), centrecercle_);
312                      end
313                     if Nb_Period>0
314                         if mod(ir,2)==1 % Starting with eps_outpillar
315                             material = eps_outpillar;
316                         else
317                             material = epsSi ;
318                         end
319                     else % juste one regular pillar
320                         material = epsSi ;
321                     end
322                     Array_of_Polygon=[Array_of_Polygon  Polygon2D(tmp,
    material)];
323                 end
324                 if Nb_Period>0
325 %OUTPUT.Area_pillar=  -area(polyshape(Array_of_Polygon(1).P))+area(
    polyshape(Array_of_Polygon(2).P));% area of the area of the pillars
326                 else
327 %  OUTPUT.Area_pillar=  area(polyshape(Array_of_Polygon(1).P));% area of
     the area of the pillars
328                 end
329             end
330         end
331     end
332     % d =  .94/(sqrt(real(epsSi))-sqrt(real(eps_square))); %hauteur des
    pilliers theorique
333     %Bug was d a,d not d_(iNpillar)
334     Layers_(iNpillar) = [layer_2D(d_(iNpillar), eps_square,
    Array_of_Polygon, Array_of_circle, current_z_(iNpillar)-d_(iNpillar),
     current_z_(iNpillar))];
335     %ind_current_layer = ind_current_layer + 1; nz=[nz round(100/Npillar
    ) ]; current_z = current_z -d;
336 end
337
338 Layers(ind_current_layer_0+[0:Npillar]) =Layers_(1+[0:Npillar]);
339 current_z= current_z_(end);  %ici peut etre un bug
340
341 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
342
343 function  [ALLcenter,ALLcentery,ax,ay]=Create_Grid_Structuration(ax,ay,
    ARG);
344
345 % create a grid for structuration with or without periodic square
    lattice
346
347 if isfield(ARG,'RectangularGrid'); RectangularGrid=ARG.RectangularGrid;
    else; RectangularGrid=1;     end
348 if RectangularGrid==1
349
350     % STRUCTURATION:
351     % definition of the centers of all pillars
352
353     if isfield(ARG,'ALLcenter'); ALLcenter=ARG.ALLcenter;
354     else
355         %ALLcenter=[-1 0 1] ;
356         ALLcenter=[ 0 ] ;
357     end
358     if isfield(ARG,'ALLcentery'); ALLcentery=ARG.ALLcentery;
359     else
360         ALLcentery= ALLcenter ;
361     end
362     ALLcenter=ALLcenter*ax;
363     ALLcentery=ALLcentery*ay;
364     Nduplicate=max(size(ALLcenter,2));
365     Nduplicatey=max(size(ALLcentery)); %ALLcentery is a 1 vector
366     % the total size of the device
367     ax=Nduplicate*ax;ay=Nduplicatey*ay; %
368 else % list of absolute values for pillar positions
369     if isfield(ARG,'ALLcenter'); ALLcenter=ARG.ALLcenter;
370     else
371         ALLcenter=[ 0 ; 0] ;
372     end
373     ALLcentery=[]; % not used ==> a list of absolute pillar positions
374 end
375
376 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
377
378 function OUTPUT = struct_LayerCoating(ARG)
379     if isfield(ARG,'ax')
380         ax=ARG.ax;
381     else
382         ax=1;
383     end
384     if isfield(ARG,'ay')
385         ay=ARG.ay;
386     else
387         ay=ax;
388     end
389     if isfield(ARG,'PML')
390         PML=ARG.PML;
391     else
392         PML = 0;
393     end
394     if isfield(ARG,'N_PML')
395         N_PML=ARG.N_PML;
396     else
```

```
        N_PML = 0;
    end
    if isfield(ARG, 'thickness_PML')
        thickness_PML = ARG.thickness_PML;
    else
        thickness_PML = 1;
    end
    if isfield(ARG, 'max_eps_PML')
        max_eps_PML = ARG.max_eps_PML;
    else
        max_eps_PML = 20;
    end
    if isfield(ARG, 'DSA')
        DSA = ARG.DSA;
    else
        DSA ='no';
    end
   %DSA Thickness  ;
    if isfield(ARG, 'DSA_Thickness')
        DSA_Thickness = ARG.DSA_Thickness;
    else
        DSA_Thickness =0.1;
    end
% multilayer
NBML=0;for iML=1:10
   if isfield(ARG, ['d_ML' num2str(iML)])
      NBML=NBML+1;
      eval(['d_ML' num2str(iML) '= ARG.d_ML' num2str(iML) ';']);
      eval(['Mat_ML' num2str(iML) '= ARG.Mat_ML' num2str(iML) ';']);
    else
        eval(['d_ML' num2str(iML) '= 0;']);
    end
end
OUTPUT.NBML=NBML;
if isfield(ARG, 'Material_Sub')
        Material_Sub = ARG.Material_Sub;
    else
        Material_Sub = 'Silicon';
    end
    % Define the structure
    % Period in x in micrometer
    x_min = -ax / 2;
    x_max = x_min + ax;
    %ay = 0.5;  % Periode in y in micrometer
    y_min = -ax / 2;
    y_max = y_min + ay;
    MAXCOORD.xmin = x_min;
    MAXCOORD.xmax = x_max;
    MAXCOORD.ymin = y_min;
    MAXCOORD.ymax = y_max;
    % PERMITTIVITY
    lambda = ARG.lambda;
% Palik data were used in fdtd and dgtd simulations.
% load('Ta205.txt','-ascii')  % wl espilon
% Eps_Ta205 = Ta205(:,2)+1j*Ta205(:,3); % Eps_R Eps_I
% wl_Ta205 = Ta205(:,1);
%load('C:\Users\lecal\OneDrive\Documents\SUPOP\2AS\TP Projet\SiO2_Palik
   ','-ascii')  % wl n  k
```

```
load(('Si_Palik.txt'),'-ascii')  % wl n  k
load(('SiO2_Palik.txt'),'-ascii')  % wl n  k
wl_Si_Palik = Si_Palik(:,1) ; wl_SiO2_Palik = SiO2_Palik(:,1);
Eps_Si_Palik = (Si_Palik(:,2).^2 - Si_Palik(:,3).^2) + 1i*(2*Si_Palik
   (:,2).*Si_Palik(:,3));  % Eps_R Eps_I
Eps_SiO2_Palik = (SiO2_Palik(:,2).^2 - SiO2_Palik(:,3).^2) + 1i*(2*
   SiO2_Palik(:,2).*SiO2_Palik(:,3)); % Eps_R Eps_I
%Eps_Ta2O5 = (Ta2O5_refractiveindex(:,2).^2 - Ta2O5_refractiveindex(:,3)
   .^2) + 1i*(2*Ta2O5_refractiveindex(:,2).*Ta2O5_refractiveindex(:,3));
     % Eps_R Eps_I
%wl_Ta2O5 = 1e3*Ta2O5_refractiveindex(:,1);
% interp at wavelengh
  epsSiO2 = interp1(wl_SiO2_Palik, Eps_SiO2_Palik, lambda*1e3);
  epsSi = interp1(wl_Si_Palik, Eps_Si_Palik, lambda*1e3);
%epsTa2O5 = interp1(wl_Ta2O5, Eps_Ta2O5, lambda*1e3);
epsAl2O3=3.02295;
    epsAir = 1;
if   strmatch(Material_Sub, 'silicon', 'exact')  ;
    epsMap = [ epsSi, epsAir];
else
    epsMap = [ epsSiO2, epsAir];
end
%multilayer
for iML=1:NBML
    eval(['tmpMat=' 'Mat_ML'  num2str(iML) ';'])
    tmpMattxt=[tmpMat '.txt'];
    eval(['load(''' tmpMattxt ''','-ascii');'])
    eval([ 'tmpMat =' tmpMattxt(1:end-4) ';'])
    tmpepsilon=nk2epsilon(tmpMat(:,2),tmpMat(:,3));
    supermicronlambda=lambda*(1e3*~isempty(findstr('Palik',tmpMattxt))
   +1*isempty(findstr('Palik',tmpMattxt)));
    if supermicronlambda<tmpMat(1,1);supermicronlambda=tmpMat(1,1);end
    if supermicronlambda>tmpMat(end,1);supermicronlambda=tmpMat(end,1);
   end
    tmpepsilon=interp1(tmpMat(:,1), tmpepsilon, supermicronlambda); %
   normlisation of the W axis
    if isnan(tmpepsilon);tmpepsilon=1;end
    eval(['epsMat_ML' num2str(iML) '=tmpepsilon;'])
    epsMap = [epsMap tmpepsilon];
end
epsMap=unique(epsMap);
%Attribute the top and bottom material (must be real)
    epssup = epsAir;
    epssub = real(epsSi);  % super- and sub-strate permeabilities
    if   strmatch(Material_Sub, 'silicon', 'exact')  ; epssub = real(
   epsSi); else epssub = real(epsSiO2);end
    MAX_Z = 10; % Max z of your structure
%SUP-Strate
    nz=1;
    d_sup = [0.01];
    current_z = MAX_Z;
    ind_current_layer = 1;
%% 4 um Air
        Array_of_Polygon = [];
        eps_square = epsAir;
    Array_of_circle = [];
    d = 10;
    Layers(ind_current_layer) = [layer_2D(d, eps_square,
```

```
    Array_of_Polygon, Array_of_circle, current_z-d, current_z)];
504     ind_current_layer = 1 + ind_current_layer;
505     nz = [nz 60];
506     current_z = current_z - d;
507 %ML
508 for iML=1:NBML
509    Array_of_Polygon = [];
510     eval(['eps_square =epsMat_ML' num2str(iML) ';'])
511    Array_of_circle = [];
512     eval(['d = d_ML' num2str(iML) ';'])
513     Layers(ind_current_layer) = [layer_2D(d, eps_square,
    Array_of_Polygon, Array_of_circle, current_z-d, current_z)];
514     ind_current_layer = 1 + ind_current_layer; nz=[nz 10 ]; current_z =
    current_z -d;
515 end
516 %DSA
517 if DSA(1)=='y'
518 ARG_DSA.PLOT=0;ARG_DSA.DSA_Thickness=DSA_Thickness;
519 %realstructure=imread('selfasembly.PNG');
520             % One level
521             %DSA_Shape=realstructure(:,:,1)>60;
522 %realstructure=imread('Mouche_zoom.PNG');
523 realstructure=imread('Mouche.png');
524 %realstructure=imread('Mouche_Nice.PNG');
525           % One level
526             DSA_Shape=realstructure(:,:,1)>100;
527             %DSA_Shape=DSA_S
528             ape(20:end-20,20:end-20);
529            % figure;imagesc(DSA_Shape)
530 ARG_DSA.DSA_Min_Nb_Edges=5; %nim size of poly
531 ARG_DSA.DSA_Min_Around_Space=0.95; % border guard
532 Layers_DSA=ADD_DSA_Layer(current_z,x_max,x_min,y_max,y_min,epsSi,epsAir,
    DSA_Shape,ARG_DSA);
533 Layers(ind_current_layer) =Layers_DSA;
534     ind_current_layer = 1 + ind_current_layer; nz=[nz 10 ]; current_z =
    current_z -ARG_DSA.DSA_Thickness;
535 end
536 %% 10um of Si
537     Array_of_Polygon=[];
538 if   strmatch(Material_Sub, 'silicon', 'exact')  ; eps_square = epsSi;
    else eps_square = epsSiO2;end%outer square
539     Array_of_circle = [];
540     d = 10;
541     Layers(ind_current_layer) = [layer_2D(d, eps_square,
    Array_of_Polygon, Array_of_circle, current_z-d, current_z)];
542     ind_current_layer = 1 + ind_current_layer;
543     nz=[nz 60];
544     current_z = current_z -d;
545 %% SUB-strate
546     d_sub = 0.01;
547 %     d_sub = 10;
548     nz = [nz 1];
549     % Constructing usual variable : d and L
550     L = max(size(Layers));
551     d=0; tmp_d = [];
552     for i = 1:max(size(Layers))
553         tmp_d = [tmp_d , Layers(i).d];
554     end
```

```
555     d = [d_sup, tmp_d, d_sub];
556     BOT_Z = MAX_Z - sum(d(2:(end-1)));
557     % disp(Layers)
558     st = dbstack;
559     OUTPUT.name = st.name;
560     OUTPUT.ax = ax;
561     OUTPUT.x_min = x_min;
562     OUTPUT.x_max = x_max;
563     OUTPUT.ay = ay;
564     OUTPUT.y_min = y_min;
565     OUTPUT.y_max = y_max;
566     OUTPUT.epssup = epssup;
567     OUTPUT.epssub = epssub;
568     OUTPUT.MAX_Z = MAX_Z;
569     OUTPUT.BOT_Z = BOT_Z;
570     OUTPUT.epsMap = epsMap;
571     OUTPUT.nz = nz;
572     OUTPUT.d = d;
573     OUTPUT.L = L;
574     OUTPUT.Layers = Layers;
575 end
576 
577 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
578 clear all
579 
580 format longE;
581 
582 addpath(genpath('/MATLAB Drive/pcode'));
583 %RCWA Library generation
584 %Generate a phase/radius library using RCWA solver for a single pillar
585 %This is a demo script, where you just have to set up few parameters.
586 %It uses another script to generate the device structure called:
587 %struct_Pilier_ODIF_PML_robin.m
588 %The structure is as follow (in nanometer):
589 
590 clear;
591 % TM or TE mode
592 %TM=0 ==> H is along x  [0 Ey Ez(theta)] [Hx 0 0]
593 %TM=1 ==> E is along x  [Ex 0 0] [0 Hy Hz(theta)]
594 
595 OPTION.TM = 0; % sweep in Lambda
596 numom = 1; %nb of points to sweep
597 % if OPTION.halfnpw>0;numom = 16;else;numom =100;end;
598 % om = linspace(1e3*1.24/940,1e3*1.24/300 ,numom); % photon energy (hbar
        omega) in  eV
599 % lambdaall = 1.24./om; %micrometer
600 %lambdaall = linspace(.3,.94 ,numom); %micrometer
601 lambdaall = linspace(.94,.94 ,numom); %micrometer
602 % Incident light angle (NEVER SET ZERO)
603 theta = 0;  % Angle with z-axis
604 phi = 0;  % Angle with x-axis
605 % Number of mode in fourier space (RCWA MANDATORY)
606 % Recommended 18 in case of non uniform layers: 18
607 OPTION.halfnpw = 8; % On change le nombre de mode ici
608 % Visualization options
609 OPTION.SlideType = ['x', 'z'];  % Type of slices.
610 OPTION.SlideValue = [0 ,  -0.05];  % value of slices.
611 OPTION.display_colormap = 1;  % 1 -> Display colormap on plots of fields
```

```
disp(" ");disp("STARTING calculation");disp(" ");

warning off
% Device

period_of_Pillar_Grid=0.47 ; % um
Nb_sim=121; %mettre bcp de point pour ne pas manquer la résonance
Wmin=0.12; Wmax=period_of_Pillar_Grid-0.11;%
Width_ = linspace(Wmin, Wmax, Nb_sim);
[PX PY]=meshgrid([-1:1]); %in um directly
[PX PY]=meshgrid([0]); %in um directly
N_grid=size(PX,1);
    % RCWA Solving : cycle over frequency
     for i = 1:size(lambdaall, 2)
           %parfor stuff
    Lambda_all =zeros(1,numom);RR = zeros(1,numom);TT = zeros(1,numom);
   AA = zeros(1,numom);
tic
    % Cycle over devices
            for j = 1:Nb_sim
        % Create the Device to be simulated
        ARG= struct;
      % Wavelength
        ARG.lambda =lambdaall(i);% 1.24/om(i);  % (um)
      %  ARG.t = j;
        % Size of the simulated region
        ARG.ax = period_of_Pillar_Grid*N_grid;
        ARG.Material_Sub='silicon';
        ARG.Material_Sub='SiO2';
        %pillar grid
        ARG.STRUCT='y';
            ARG.ALLcenter=period_of_Pillar_Grid*[PX(:) PY(:)];
            ARG.ALLcentery=[];
            ARG.RectangularGrid=0;
            ARG.PillarWidth=Width_(j)+zeros(1,(size(ARG.ALLcenter,1)));
            ARG.PillarWidth2=Width_(j)+zeros(1,(size(ARG.ALLcenter,1)));
            ARG.PillarInnerWidth=ARG.PillarWidth-.12;
            ARG.Pillar_type="circSquare" ;
             ARG.Pillar_type="circle" ;           ARG.Pillar_NbPeriod
   =0+0*ARG.PillarWidth;ARG.Pillar_circSquareangle=10+0*ARG.PillarWidth;
    %from circle to sqr
            ARG.Npillar=0;
            ARG.angle_pillar=0+0*ARG.PillarWidth;
% Multi coating
%ARG.d_ML1=0.06;
ARG.Mat_ML1='SiO2_Palik';
%ARG.d_ML2=0.047;ARG.Mat_ML2='Ta205_refractiveindex';
%ARG.d_ML3=0.013;ARG.Mat_ML3='Al2O3_refractiveindex';
        % DEVICE creation
        DEVICE = struct_LayerCoating2(ARG);
% Visualization at the last frequency (do not work with a parfor)
%  visualization_structure3D(DEVICE, OPTION);
             if (j==Nb_sim)
                 figure(4)
% Possibilité de faire run section ici
                       OPTION_3D=OPTION;
                       OPTION_3D.SlideType=[ ];% type of slides.
                       OPTION_3D.SlideValue=[ ];% value of slides.
```

```
667                         OPTION_3D.PlotEdge='no';
668                         OPTION_3D.PlotEdge='yes';
669                         OPTION_3D.Plotfast='on';
670                         OPTION_3D.epsilon_Visible='no';
671                         visualization_structure3D_Perpective(DEVICE,
    OPTION_3D,get(gcf,'number'));
672                         set(gca,'ZLim',[-1 8])
673 % Pour visualiser La figure 2 faire run section ici %remove air
674                         Visualisation3D_Remove(get(gcf,'number'),[ 1 0
    0]);set(gca,'zlim',[-3 1])
675                         drawnow
676             end
677 % RCWA3D Simulation
678         SOL = RCWA_3D_singlecore(DEVICE, ARG.lambda, theta, phi, OPTION)
    ;
679         ModalPhase(j, :) = SOL.ModalPhase;
680 % Retrieving result
681         RR(j) = SOL.RR;
682         TT(j) = SOL.TT;
683 %         AA_  = SOL.AA;
684 %         L = DEVICE.L;
685 %         AA(j) = AA_(L);
686     end
687     toc
688 ALLPhase( :, :) = ModalPhase;
689 ALLTrans( :) = TT;
690 % Figure of the phase/transmission versus pillar width
691 monitor=0;
692 figure(40)
693 Phh = ModalPhase(:, end-monitor) - ModalPhase(1, end-monitor);
694 %Phh(find(Phh<0)) = Phh(find(Phh<0))+2*pi;
695 Phh=unwrap(Phh);
696 %Phh = abs(Phh-2*pi);
697 yyaxis left
698 title('Phase and transmission as a function of the pillar width')
699 plot(Width_(1:size(Phh,1))/2, Phh, '-o')
700 xlabel('Radius (um)')
701 yticks([0 pi/4 pi/2 3*pi/4 pi 5*pi/4 3*pi/2 7*pi/4 2*pi 9*pi/4 10*pi/4
    ])
702 yticklabels({'0', ' ', '\pi/2', ' ', '\pi', ' ', '3\pi/2', ' ', '2\pi',
    ' ','5\pi/2'})
703 axis([Wmin/2 Wmax/2 0 3*pi])
704 ylabel('Phase')
705 hold on
706 yyaxis right
707 plot(Width_(1:size(Phh,1))/2, TT, '-s')
708 legend({'Phase','Transmission'},'Location','west')
709 ylabel('Transmission (norm)')
710 ylim([0 1])
711 set(gcf,'color',[1 1 1])
712 % eval([' savefig( '''  NAME_SIMULATION '_halfnpw-' num2str(OPTION.
    halfnpw) ''')'])
713 end
714 return
715 load /prj/ai4odiff/users/rd10/MPW6_AI_BENCH_2023/
    DS_Odiff2_etchV4_1_insensitive_220122.mat
716 PHASE=(unwrap(ds.phiT - ds.phiT(1)  ));
717 PillarW=2*ds.scatterer.scatterer_radius_top_x;
```

```
718 yyaxis left
719 plot(0.5*PillarW,PHASE,'--')
720 legend({'Phase','Transmission','Phase Ref'},'Location','west')
721 NAME_SIMULATION='ODIFF_LIB'
722 eval([' savefig( '''  NAME_SIMULATION '_halfnpw-' num2str(OPTION.halfnpw
    ) ''')'])
723 %% Plot fields after library calculation
724 % Index of the radius to plot
725 idx = Nb_sim / 2;
726 r = radius(idx);
727 SOL = ALL_SOL(idx);
728 if OPTION.FIELD == 1
729     OPTION.PLOTPHASE = 0;
730     PlotField3D(SOL, DEVICE, OPTION);
731     OPTION.PLOTPHASE = 1;
732     PlotField3D(SOL, DEVICE, OPTION);
733 end
```

D Sources

[1] Denis Rideau, "Don't miss the meta-surface revolution!" (2024).

[2] Nanxi Lia, Zhengji Xua, Yuan Dong, Ting Hu, Qize Zhong, Yuan Hsing Fu, Shiyang Zhu and Navab Singh, "Large-area metasurface on CMOS-compatible fabrication platform : driving flat optics from lab to fab", Nanophotonics (2020).

[3] Hemant Kumar Raut, V. Anand Ganesh, A. Sreekumaran Nairb and Seeram Ramakrishna, "Anti-Reflective Coatings : A Critical, In-Depth Review", Energy and Environment Science (2011).

[5] Robin Feuillassier, "Numerical simulation for optical metasurfaces design", Internship report (2022).

[6] https ://empossible.net/critical angle/

[7] Mathieu Hébert, "Optical Models for Material Appearence" (2022)