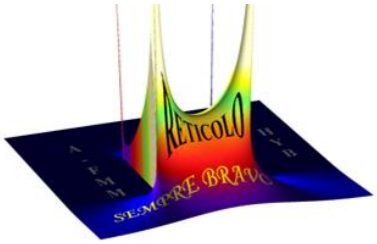


Analysis of stacks of anisotropic uniform layers: user guide for the RETICOLOfilm-stack program

Jean-Paul Hugonin and Philippe Lalanne
Institut d'Optique Graduate School, CNRS
version V5
DOI: 10.5281/zenodo.7429030
(last update: March 2024)

Abstract: The present document is the documentation of the 'RETICOLOfilm-stack' program, a free software operating under Matlab. The program computes the reflection and transmission of arbitrary stacks of anisotropic thin films. RETICOLOfilm-stack is vectorialized and thus treats several wavelengths and incidences in a single instruction. Compared to the state of the art, see Section XI.2 of the present document, RETICOLOfilm-stack appears highly reliable, stable, effective and general. We expect that it may be useful.

To install the software, download it from Zenodo and add the downloaded folder in the Matlab path.

Technical questions and help should be preferentially addressed to the main author: jean-paul.hugonin@universite-paris-saclay.fr.

We kindly ask that you acknowledge the package and its authors in any publication/report for which you use it. The preferred citation is:

> J.-P. Hugonin and P. Lalanne, RETICOLOfilm-stack, DOI: 10.5281/zenodo.7429030 (2022).

Outline

Introductory remarks

RETICOLOfilm-stack is related to another freeware, RETICOLO [11], which analyzes the diffraction by gratings with the Rigorous Coupled Wave Analysis. The version V9 of RETICOLO discusses the possibility of studying thin-film stacks, simply by retaining a single component of the RCWA Fourier extension. However, it is not optimized in computation time, because loops for scanning the wavelengths and incidences have to be implemented. This is the reason why we propose the present simplified version, which is fully-vectorial and thus much faster.

We advise the readers to familiarize themselves with the notations, by initially reading the help of RETICOLO V9 [11], while keeping in mind that only a single Fourier component (nn=0) is needed for uniform layers with the Rigorous Coupled Wave Analysis.

The symbol

labels paragraphs that can be skipped in first reading and may be useful by advanced users to speed up the computation.

The $\exp(-i\omega t)$ convention is used hereafter. Please note two interesting programs that are computing the guided modes of stacks in the guided-mode and quasinormal-mode representations, $\tilde{k}(\omega)$ and $\tilde{\omega}(k)$. These two programs are pole_interface_air_biaxe_4x4_nv.m and pole_omega_kpar_fixed_2x2_nv.m. They both consider the hyperbolic modes of an interface between two half spaces, air and an hyperbolic material.

I. Reminder: electromagnetism of stacks of uniform layers

1. Two cases

A thin-film stack is characterized by a series of 3×3 matrices defining the permittivity and permeability tensors, **ε** et **μ**, of every layer. The axis z denotes the normal to the stack, oriented from 'bottom' to 'top'.

The electric field can be written

$$\boldsymbol{E}(z)\, exp\left[i\left(K_{//,x}\, x + K_{//,y} y\right)\right], \tag{1}$$

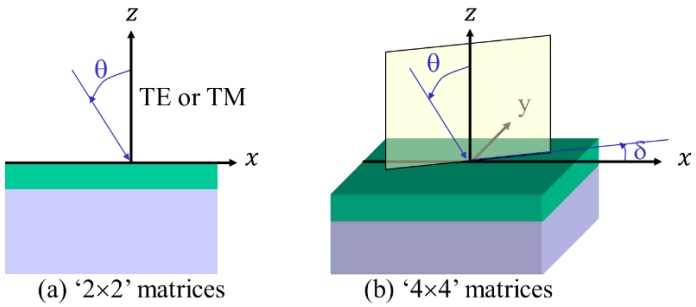
and a similar expression exists for the magnetic field. $\mathbf{K}_{//} = K_{//,x}\hat{\mathbf{x}} + K_{//,y}\hat{\mathbf{y}}$ is the same in all layers (it is an invariant). In the programs, $\mathbf{K}_{//}$ is denoted by 'beta0'.

For a given $\mathbf{K}_{//}$, Maxwell equations reduce to the integration of a differential equation in z

$$\frac{d}{dz}\begin{pmatrix} E_T \\ H_T \end{pmatrix} = \boldsymbol{A}_{K_{//}}(z) \times \begin{pmatrix} E_T \\ H_T \end{pmatrix}, \tag{2}$$

where the unknows E_T and H_T denote the tangential (or parallel) contributions of $\mathbf{E}(z)$ and $\mathbf{H}(z)$.

We distinguish two cases (Fig. 1).



(a) '2×2' matrices (b) '4×4' matrices

Fig. 1 (**a**) If **ε** et **μ** are diagonal matrices and $\varepsilon_{xx} = \varepsilon_{yy}$ and $\mu_{xx} = \mu_{yy}$, Maxwell equations split in two sets of equations, one for TE (E//Oy) and the other for TM (H//Oy); the computation then relies on '2×2' matrix algebra. (**b**) Otherwise, the computation relies on '4×4' matrix algebra.

The problem illustrated in Fig. 1a is solved with 2×2 matrices, that of Fig. 1b with 4×4 matrices. In the following, the two cases will be referred to as '2×2' and '4×4'. Only the case '4×4' makes it possible to differently couple right and left circular polarizations.

At the top (up or top) and at the bottom (bottom), we must satisfy outgoing wave conditions. For that purpose, we write the continuity conditions for the tangential field components of the plane waves of the bottom and top layers.

The integration of the differential equation and the continuity conditions make it possible to calculate the complex amplitudes of the reflected and transmitted plane waves.

In every uniform layer for which **ε** and **μ** are independent of z, the integration of the differential equation requires exponentiating a matrix, which is achieved thanks to a diagonalization of the matrix $\mathbf{A}_{K_{//}}$ (of size 2×2 or 4×4) associated with the layer.

2. Vectorialisation

It is crucial for the calculation time to 'vectorize', i.e. to process several frequencies and several incidences at the same time.

Two important parameters will intervene in the code to set up this vectorialization and the dimensioning of the associated arrays:

- **nb_k0**, the number of wavevectors ($k_0 = \frac{2\pi}{\lambda}$) and thus of wavelengths (λ).
- **nb_inc**, the number of incidence angles.

II. First step: initialization

The 'res0_0D' function feeds global variables (invisible to the user) which will be used throughout the remainder of the program without the user having to worry about them.

An important variable is the $K_{//}$ wavevector array, called beta0. If one illuminates by various incidence angles and various wavelengths (as is often the case), $K_{//}$ is 2*pi./ld(:).*n_inc(ld).*sin(teta). Then we have: size(beta0) =[nb_k0, nb_inc] for the 2×2 case and size(beta0) =[nb_k0, nb_inc, 2] for the 4×4 case, beta0(:,:,1) being the x-components and beta0(:,:,2) the y-components.

By default, the angles are expressed in radians. To express them in degrees (as in reticolo1D and reticolo2D), it is necessary to add a structure with field parm=struct('option_degre',1) at the end of the arguments of res_0D.m. This structure also makes it possible to modify the cut, angle_cut in radians or in degrees according to option_degre, of the square root function (see Section X.4). By default, we have: parm= struct('option_degre',0,'angle_cut',pi/2).

For instance, we may have: res0_0D (pol,k0,beta0,struct('option_degre',1,'angle_cut',10)

Why may we need to modify angle_cut? The incident medium is isotropic and lossless, but the outgoing medium can be arbitrary. In some cases, if the outgoing medium has a 'complicated' anisotropy, the classical 90° cutoff can lead to instabilities. It is then necessary to choose a lower angle to give priority to the attenuation, see Section X.4.

Several formats are possible for the function **res0_0D**:

1. '2×2'

The TE et TM polarisations are separated: pol=1 (TE) or pol=-1 (TM). The convention pol=0 (TE) or pol=2 (TM) is also possible.

a. One directly enters the $K_{//}$ array

res0_0D(pol, k0, beta0) with size(beta0)=[nb_k0, nb_inc].

b. One directly enters wavelengths and angles of incidence separately

res0_0D(pol, k0, theta, n_inc) with

n_inc: refractive index of the incident medium (upper or lower layers); this medium is always isotropic and lossless. For dispersive incident media, n_inc depends on the wavelength and can be a function or a vector of length nb_k0. The output medium may be anisotropic and absorbing.

theta: angles of incidence (Fig. 1a).

2. '4×4'

The parameter 'pol' is not used for '4×4' matrices.

a. One directly enters the $K_{//}$ array

res0_0D (k0, beta0) avec size(beta0)=[nb_k0, nb_inc, 2] et beta0(:,:,1)=$K_{//x}$, beta0(:,:,2)=$K_{//y}$.

b. One directly enters wavelengths and angles of incidence separately

In a more 'friendly' way, one can use the theta and delta angles to define the incidence:

res0_0D (k0, theta, delta, n_inc, meshed), the angles theta et delta being defined in Fig. 1b.

n_inc: refractive index of the incident medium (upper or lower layers); **the incident medium is always isotropic and lossless**. For dispersive incident media, n_inc depends on the wavelength and can be a function or a vector of length nb_k0.

Do not forget the parameter 'meshed'. Its use is required:

- if meshed=0

the theta and delta vectors do not have the same length and are transformed in the code to

[theta, delta]= ndgrid(theta, delta) with nb_inc=length(theta) * length(delta);

example:

delta=5; theta=0:90;

- if meshed=1

We then define a set of incidences, i.e. of couples {theta, delta}. Vectors theta and delta have the same dimension [1, nb_inc] and the incidence number ii is defined by the two angles theta(ii) and delta(ii).

example:

[theta,wt]=retgauss(0,pi/2,10,20);[delta,wd]=retgauss(0,2*pi,-90); % theta and delta have different sizes

[Teta,Delta]=ndgrid(theta,delta);[wt,wd]=ndgrid(wt,wd);

W=wt(:).*wd(:).*sin(Teta(:)); % Teta and Delta have the same size

res0_0D (k0, Teta, Delta, 1.5, 1)

Then, one may plot:

u=sin(Teta).*cos(Delta);v=sin(Teta).*sin(Delta);w=cos(Teta);

figure;

surf(u.*Rh,v.*Rh,w.*Rh,'facecolor',[.5,.5,.5],'LineStyle','none');

set(gca, 'projection','perspective'); axis tight; axis equal; xlabel('x'); ylabel('y'); lighting gouraud; light('position',[-1,0,1],'color','r'); light('position',[-1,-5,1],'color','w'); light('position',[-1,-5,1], 'color','y');

Attention, in this case, res0_0D (k0, Teta, Delta, 1.5, 0) is wrong and often leads to an error: “out of memory”.

III. Second step: layer definition and mode computation

a=res1_0D(n) constructs the 'descriptor' ‘a’ of a layer ‘n’. The descriptor is a cell array that generally contains the result of the diagonalization of the matrix $\mathbf{A}_{K_{//}}$. It can then be used to propagate the field ((step 3), to fulfill the boundary conditions (step 5), or to calculate the fields (step 7). The variable ‘n’ defines the electromagnetic parameters (**ε**, **μ**) and can take various forms. ‘n’ corresponds to the layer refractive index only in some specific cases.

1. ‘2×2’ or ‘4×4’

- without dispersion: n=1.5;
- with dispersion:

n=@(ld) retindice(ld, 2.13);

or

n=[line vector of the refractive indices for every wavelength]; % length(n)=nb_k0

2. ‘2×2’

a. Anisotropic medium with diagonal anisotropy

- **ε** only

n=@(ld) {nx, [], nz}; ny=[] is not needed since for the ‘2×2’ case, nx=ny. nx=@(ld) ... is a function calculating nx as a function of ld (note, we define nx, not $\varepsilon_{xx}= \varepsilon_{yy}= nx^2$ or $\varepsilon_{zz}= nz^2$).

with nx the array containing the values of nx (size(nx)=nb_k0). Idem for nz.

- **ε** and **μ**

n=@(ld) {nx, [], nz, mx, [], mz}; ny=nx and my=mx are not needed.

mx and mz being either fonctions or arrays of length nb_k0 ($\varepsilon_{xx}= \varepsilon_{yy}= nx^2$, $\varepsilon_{zz}= nz^2$, $\mu_{xx}= \mu_{yy}= mx^2$ and $\mu_{zz}= mz^2$) .

exemple: n=@(ld) {n0*1i,[],n0*1i,1i,[],1i}; % perfect lens

3. '4×4'

a. Anisotropic medium with arbitrary anisotropy

- **ε** only

Now, the input 'n' represents the permittivity tensor and not the refractive index as in the diagonal anisotropy case. 'n' is a cell array: n={epsilon};

with dispersion: size(epsilon)=[3,3,nb_k0] ;

without dispersion: size(epsilon)=[3,3] and the matrix is repeated identically nb_k0 times.

- **ε** and **μ**

Now, we have: n={epsilon, mu};

with dispersion: size(mu)=[3,3,nb_k0] ;

without dispersion: size(mu)=[3,3] and the matrix is repeated identically nb_k0 times.

b. Anisotropic medium with diagonal anisotropy

This case is not explicitly provided. The user should refer to the general case. An illustrative example is the following:

without dispersion:

epsilon=diag ([1.2, 1.3, 1.4]);

with dispersion:

epx=retindice(ld, 514).^2;epz=retindice(ld, 515).^2; % hBN

epsilon=zeros(3,3,length(ld));epsilon(1,1,:)=epx;epsilon(2,2,:)=epx;epsilon(3,3,:)=epz;

where 'ld' is a 1xn vector of wavelengths (the unit is not imposed; using the micron unit for any length is a reasonable choice at optical frequencies).

4. We can also introduce an optional parameter p

To fulfill the boundary conditions and define the 'su' and 'sb', we need to differentiate the modes which propagates up from those which propagate down with appropriate cuts. For large arrays with many wavelengths or incidences, the computation might require some time..

a=res1_0D(n, p);

By default, p=1. If p=0, we avoid sorting the eigenvectors and we save time. However, one cannot then implement the boundary conditions for these layers, i.e. these layers cannot be either the top or bottom media.

IV. Third step: S-matrix computation

The integration of Eq. (2) on the height 'h' leads to a matrix **T** linking the tangential components of the fields at the bottom of the layer to the tangential components of the fields at the top of the layer. Because of the large values of some eigenvalues of the $\mathbf{A}_{K_{//}}$ matrix, the matrix **T** can become very large; this leads to instabilities. We prefer to use the matrix formalism **S** which is stable

s=res2_0D(a,h) ;

'a' layer descriptor calculated in the third step

'h' layer thickness

‘s’ is an object of class 'ret_matrice_S'. This allows matrices to be manipulated with the usual operators such as '*' and '^' to perform **S**-matrix products. For example

s=s1*s2*s3; % product of 3 matrices (noncommutative product: layer 1 is above, layer 2 below …)

s=s^5;

V. Forth step: compute the total matrix S_tot

1. Product and power of matrices S

A section is an assembly of contiguous layers of total height ‘h’ described by a matrix **S**. For example, ‘S_123’ is the **S** matrix of the section associated with: S_1, S_2, S_3 (see Fig. 2). The **S** matrix connects the tangential components of the fields at the bottom and at the top of the section.

As these tangential components are continuous, there is no need to write a transition matrix between two sections and 'S_123' can be used everywhere like a lego piece, using the associativity of the product of **S** matrices [1].

In Fig. 2, we show how we reduce 16 matrix products to 6 matrix products (2 products for S_123, 2 others for S_123^4 and another 2 others for **S**), which corresponds to a strong reduction in the computation time. Note that raising to an integer power uses an efficient factorization in powers of 2 [1].

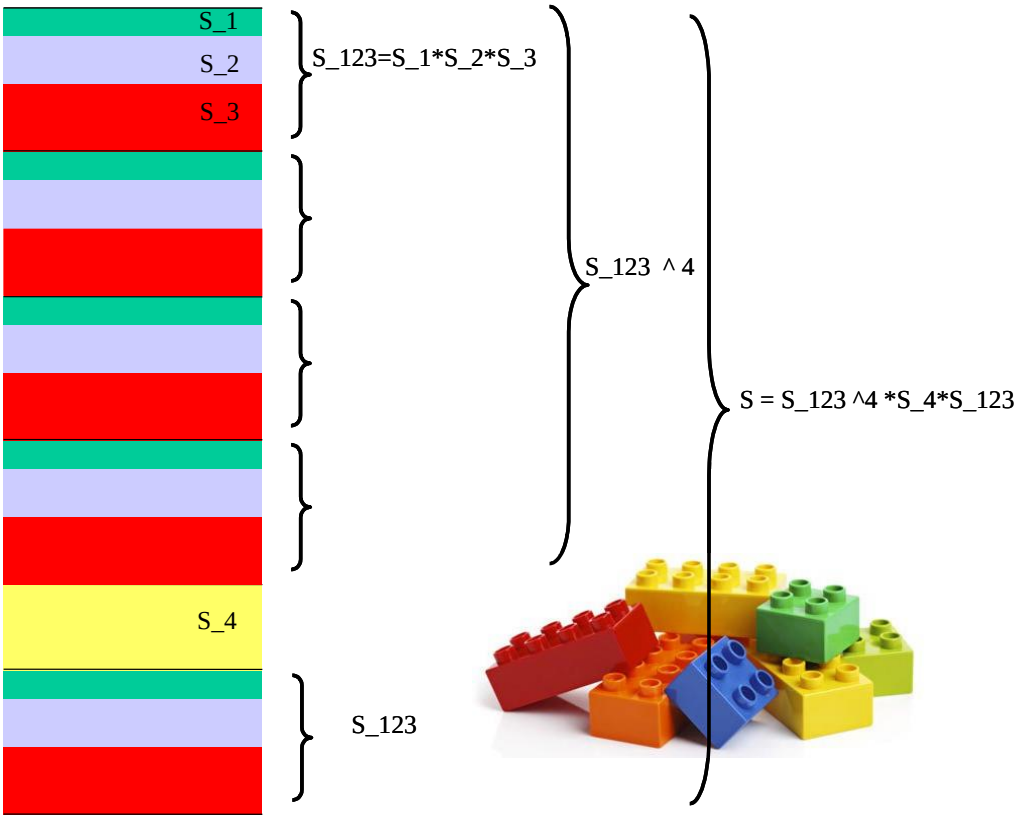


Fig. 2 Product and power of **S** matrices. Attention, the product is done from top to bottom, S=S_123^4*S_4*S_123, like the product of **T** matrices.

2. Bloch modes of a section

To diagonalize the **T** matrix associated to the **S** matrix, we use [10] :

abloch= res1_0D (S,h)

with h=h1+h2+h3 the total thickness of the section

‘abloch’ can be used to write periodic boundary conditions.

If the user wants to know the eigenvalues ‘d’ and the eigenvectors ‘V’:

[abloch,d,V]= res1_0D (S,h); % T=V*diag(exp(d*h)*inv(V).

VI. Fifth step: boundary conditions and final result

1. Reflection and transmission computation (similar to the grating codes)

[result, su, sb]=res2_0D(s, au, ab);

The res2_0D function calculates the matrices ‘su’ and ‘sb’ to satisfy the outgoing wave conditions in the upper and lower media defined with descriptors ‘au’ and ‘ab’.

If we have a single interface between a substrate and a superstrate, we must write:

[result, su, sb]=res2_0D([], au, ab);

Please note that, as for the grating codes, we do not obtain the classic Fresnel formulas (given in many textbooks) because of the normalization used for the plane waves.

Alternatively, one may directly calculate the matrices ‘su’ et ‘sb’ without using res2_0D.m:

su=retb_0D(au,1) ;

sb=retb_0D(ab, -1);

The total matrix S_tot, which connects the plane waves in the substrate and the superstrate is then obtained by

s_tot=su*s*sb ;

2. Results: the structure ‘result’

The 'result' structure is very similar to those of the grating codes (RETICOLO1D and RETICOLO2D), except for the '4×4' case for which we have additional specific outputs for handling polarization conversions between circularly polarized plane waves. For instance:

result.Jones.inc_top_reflected.efficiency_R_2_L

provides the right-to-left circular polarization conversion.

Please also consider Section VIII.3 for reciprocity condition concerning the reflection and transmission coefficients (amplitude and intensity).

a. ‘2×2’

The XXX fields of the ‘result.XXX’ structure are given by the following boxes:

inc_top inc_bottom	PlaneWave_E: [3 x nb_k0 x nb_inc] PlaneWave_H: [3 x nb_k0 x nb_inc]

inc_top_reflected inc_top_transmitted inc_bottom_reflected inc_bottom_transmitted	PlaneWave_E: [3 x nb_k0 x nb_inc] PlaneWave_H: [3 x nb_k0 x nb_inc] efficiency: [nb_k0 x nb_inc] amplitude: [nb_k0 x nb_inc]

b. ‘4×4’

The XXX fields of the ‘result.XXX’ structure are given by the following boxes:

TEinc_top TMinc_top TEinc_bottom TMinc_bottom	PlaneWave_TE_E: [3 x nb_k0 x nb_inc] PlaneWave_TE_H: [3 x nb_k0 x nb_inc] PlaneWave_TM_E: [3 x nb_k0 x nb_inc] PlaneWave_TM_H: [3 x nb_k0 x nb_inc]

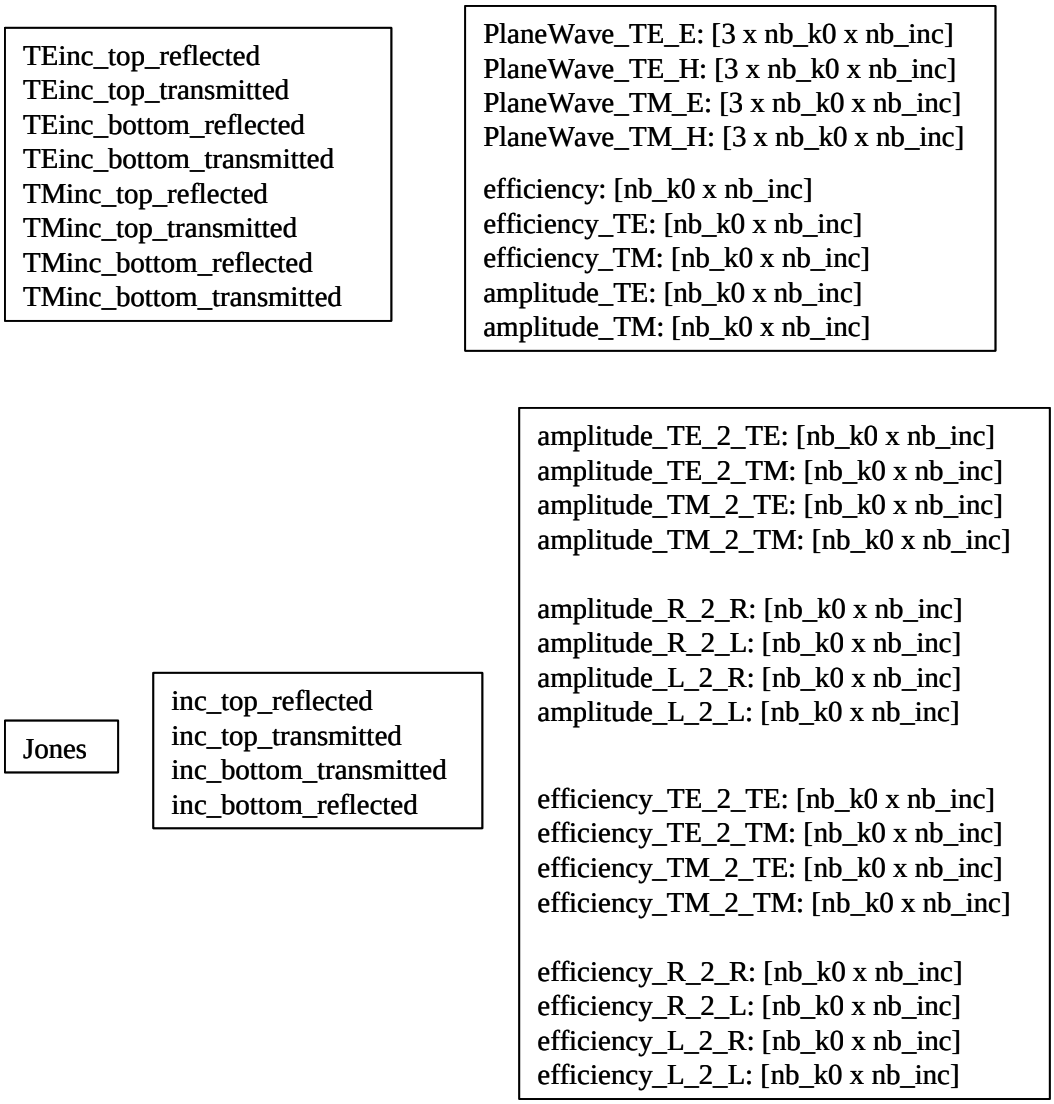


Fig. 3 The different XXX fields of the ‘result.XXX’ structure.

3. Reducing the computational time

The calculation of the structure 'result' can be quite long and can be avoided by executing:

[result, su, sb, S_tot]= res2_0D (s,au,ab);

By introducing a fourth output, S_tot, 'result' is not calculated (the res2_0D function then returns [] instead).

To extract the coefficients of transmission and reflection, it is then necessary to directly use the S_tot matrix

size(S_tot)=[2 ,2,nb_k0,nb_inc] for ‘2×2’

size(S_tot)=[4 ,4,nb_k0,nb_inc] for ‘4×4’

which adopts the conventions defined in Fig. 4.

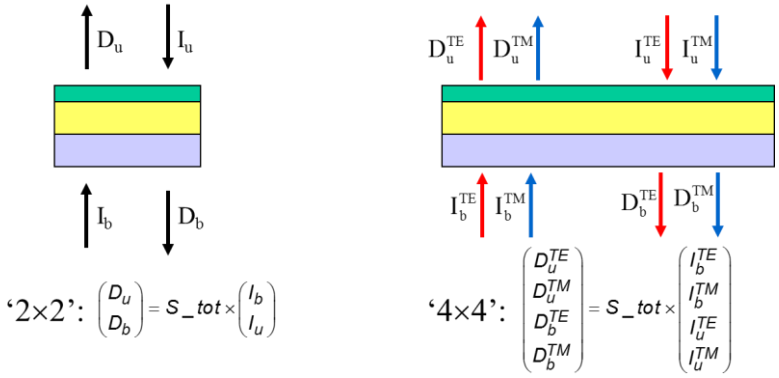


Fig. 4 Definition of the S_tot matrix that links the incident 'I' and diffracted 'D' (reflected or transmitted) plane waves on the top 'u' ('u' means up) and bottom 'b' media, for every individual value of the parallel wavevector.

Reminder: The incident medium must be isotropic and non-absorbent, but the outgoing medium is arbitrary. If the outgoing medium is not isotropic or is absorbing or if the transmitted waves are evanescent, the modal efficiencies given by the 'result' structure in the outgoing medium are set to 0 (but not the amplitudes of the modes).

VII. Field computation and plots, loss-per-layer computation

1. Field computation

a. '2×2'

[e, z, index,wz]=res3_0D(x, aa, Profile, einc, parm);

size(e)=[length(z), length(x), length(y), 3, nb_k0, nb_inc], where '3' corresponds to E_y, H_x, H_z for TE et $H_y, -E_x, -E_z$ for TM.

size(index)=[length(z),length(x),length(y),nb_k0,nb_inc]

'wz' is the vector of Gauss weigth to be used to compute z integrals, see Section X.6.b.

'einc' is the amplitude of the incident plane wave.

b. '4×4'

[e,z,index,wz]=res3_0D(x,y, aa, Profile, einc, parm);

size(e)=[length(z),length(x),length(y), 6, nb_k0,nb_inc], where '6' corresponds to E_x, E_y, E_z, H_x, H_y, H_z.
size(index)=[length(z),length(x),length(y),nb_k0,nb_inc]

'wz' is the vector of Gauss weigth to be used to compute z integrals, see Section X.6.b.

einc=[amplitude incidente TE, amplitude incidente TM]. For example, for a circular polarization, einc=[1, i]/sqrt(2).

Assigning many values to the 'x' and 'y' vectors risks increasing the calculation time. It is recommended to calculate the field in $x = y = 0$ and to manage the variation in $\exp\left(i\mathbf{K}_{//} \cdot (x\hat{\mathbf{x}} + y\hat{\mathbf{y}})\right)$ 'by hand' in the field plots.

2. The 'Profile' variable

Just like for RETICOLO1D and 2D [11], the user should define the 'Profile' variable that contains, starting from the top layer and finishing by the bottom layer, the successive information (thickness and texture-label) relative to every layer. Here is an example (Fig. 5) that illustrates how to set up 'Profile':

Profile = {[h_h,1,0.5,0.5,1,0.5,0.5,2,h_b],[1,3,2,4,3,2,4,6,2]};

From the top to the bottom, we have: the top layer is formed by texture 1 with a thickness h_h, then we have two repetitions of the series of textures 3, 2, 4 with thicknesses 1, 0.5, 0.5 µm, respectively. This repetition is followed by texture 6 with a thickness 2, and finally the bottom layer, formed by texture 2, has a thickness h_b. For h_b=h_h=0, the fields in the top and bottom layers are not plotted.

In the previous example, the structure formed by texture 3 with thickness 1, texture 2 with thickness 0.5 and texture 4 with thickness 0.5 is repeated twice. It is possible to simplify the instruction defining the 'Profile' variable. One may write :

Profile = {{h_h,1},{[1,0.5,0.5], [3,2,4], 2},{[2,h_b],[6,2]}};

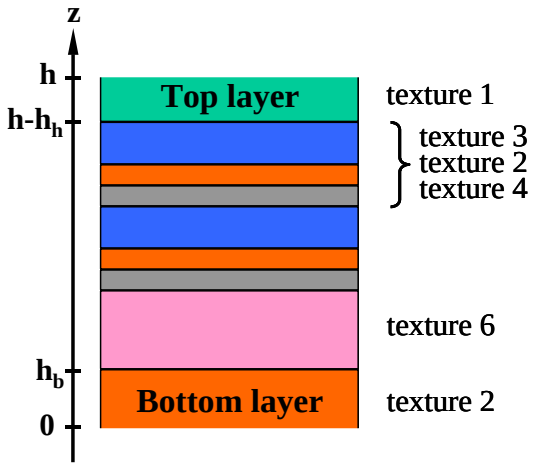


Fig. 5. Texture stacks. The example corresponds to a profile defined by
Profile = {[h_h,1,0.5,0.5,1,0.5,0.5,2,h_b],[1,3,2,4,3,2,4,6,2]};
Just like for gratings [11], we may also have:
Profile={{h_h,1},{[1,0.5,0.5], [3,2,4], 2},{[2,h_b],[6,2]}}.

3. 'parm' variable

parm=struct('sens',1,'npts',10);

is the default value.

Some **important** comments on the **parm**" variable are:

1/ Illuminating the grating from the top or the bottom layer: the user must specify the direction of the incident plane wave. This is specified with variable '**sens**':

sens=1 incidence from top (by default)

sens=-1 incidence from the bottom

2/ As for the gratings [11], the sampling points in z are determined by a Gauss Legendre method. The user may choose the number of subintervals and the degree in every layer using the parameter npts. The number of columns of npts is the number of layers npts =[[10,5,12];[3,1,5]]; means that 3 subintervals with points of degree 10 are used in the first layer, 1 with degre 5 in the second, 5 with degree 12 in the third layer.

npts=[10,5,12] is the same as npts= [[10,5,12];[1,1,1]]; npts=10 is the same as npts= [[10,10,10];[1,1,1]]; The degree must be smaller than 1000; however to have a good distribution of sampling points, it is recommended to keep the degree smaller than 20.

3/ **VERY IMPORTANT**: where is the $z = 0$ plan and what are the z-coordinates of the $z =$ constant plan? The $z = 0$ plan is defined at the bottom of the bottom layer. Thus, the field calculation is performed only for $z > 0$ values. For the example in Fig. 5, if we refer to the Bottom layer as the substrate, the $z = 0$ plane is located in the substrate at the distance h_b under the substrate interface. Note that the z coordinates for the $z =$ constant plans are always given by the second output variable of res3_0D.

4/ How can one specify a given z=constant plan? First, one has to redefine the variable **Profile**. For the example, if one wants to compute the field in the center of texture 6 of Fig. 5, one must take:
Profile={[h_h,1,.05,.05,1,.05,.05,1,0,1,h_b],[1,3,2,4,3,2,4,6,6,6,2]}
npts=[0,0,0,0,0,0,0,0,1,0,0].

4. Electromagnetic-field plots

The computation of the transmission and reflection coefficients is vectorized; the plotting of the fields is done for a specific $K_{//}$ value, 'i_inc', and a specific k0 value, 'i_k0'.

a. 2×2

res3_0D(e(:, :, :,i_k0,i_inc),index(:,:,i_k0,i_inc),x,z,[num_champ,option,1i]));

b. 4×4

res3_0D(e(:, :, :, :,i_k0,i_inc),index(:, :, :,i_k0,i_inc),x,y,z,[num_champ,option,1i]);

'1i' : the last term '1i' allows to trace the contour of the object (recommended otherwise we may also plot a map of the refractive index of the object).

'index' is an array of complex numbers. For homogeneous and isotropic layers, 'index' represents the refractive index. For anisotropic layers, 'index' represents an average of the refractive index tensor; it has no precise meaning but only allows to represent the interfaces between the different layers on the field maps.

'num_champ' makes it possible to draw several components of the fields on the same figure with different subplots. For example, for num_champ=1:3;, one plots all the field components for the '2×2' case.

num_champ	TE	TM
1	E_y	H_y
2	H_x	$-E_x$
3	H_z	$-E_z$

Other example, for num_champ=[4,6]; one plots Hx and Hz for the '4×4' case.

num_champ	
1	E_x
2	E_y
3	E_z
4	H_x
5	H_y
6	H_z

Suppose the user has computed the electromagnetic fields for 1000 wavelengths and 90 angles of incidence. To visualize them, we can make a film. We can also combine them to see the evolution of a pulse as in the example provided in 'resonateur_de_bragg_pulse_Gaussien.m'.

c. Plot options

option	
11	plot abs(XXX).^2, with XXX=E_x, E_y …
12	plot abs(XXX)
13	plot real(XXX)
14	plot imag(XXX)
15	time animation, real(XXX*exp($-i\omega t$))

16	plot log10(abs(XXX))

5. Loss

This section explains how to compute the absorption loss in every layer (please remember that the layer stack is defined by the variable 'Profile'). It also explains how to compute the loss in the top (superstrate) and bottom (substrate) layers. More details can be found in Section 8.2 in [11], keeping in mind that the virtual period is 1.

[Losses_per_layer,Flux_Poynting]=res3_0D(aa,Profile,einc,*parm*);

- size(Losses_per_layer)=[nb_k0,nb_inc,length(Profile{1})]
- size(Flux_Poynting)=[nb_k0,nb_inc, length(Profile{1})+1]
- parm=struct('sens',1) by default
- The losses per layer are computed for an incident flux (on the virtual period 1) of abs(einc)^2 for 2×2 and sum(abs(einc).^2) in 4×4.

Figure 6 summarizes the two possible cases:

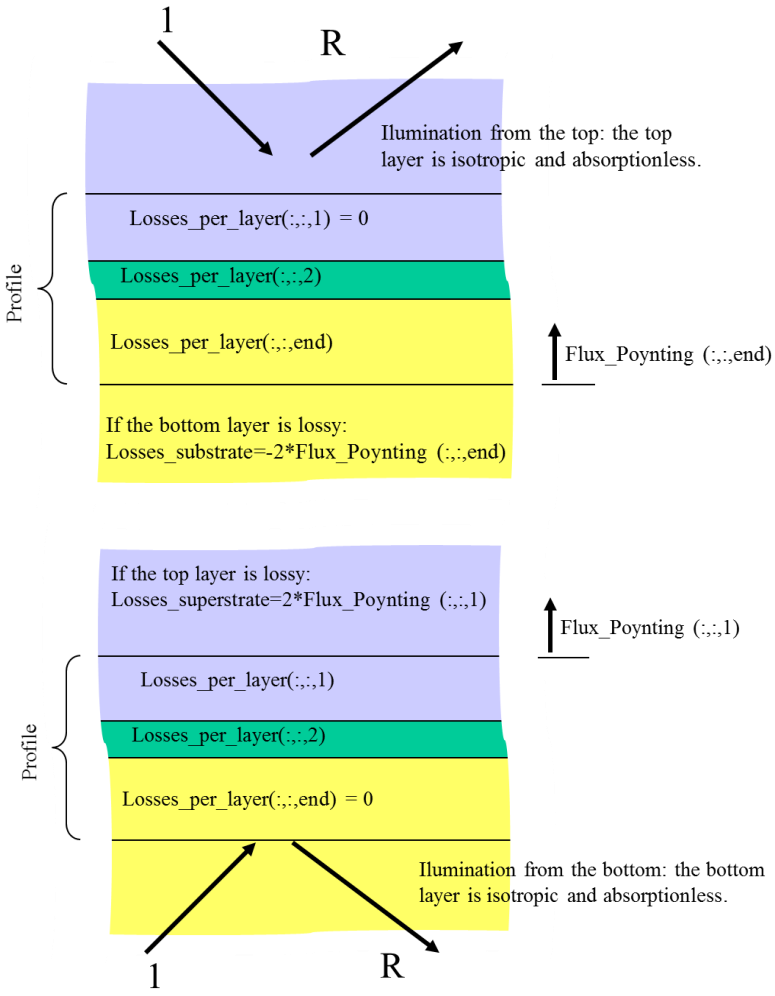


Fig. 6 Computation of the absorption loss in every layer for two usual cases. Upper inset: illumination from the top for a lossy substrate. The absorption in the substrate (bottom layer) is -2*Flux_Poynting(:,:,end). Lower inset: illumination from the bottom layer for a lossy superstrate (top layer). The absorption in the top layer is 2*Flux_Poynting(:,:,1). More details can be found in Section 8.2 in [11].

VIII. Fast computation of ‘ru’, ‘t’ and ‘rb’ (only for ‘2×2’)

In the previous Sections, the sofware was introduced with advanced user-friendly functions. In this Section, we introduce the way the authors often use the software. The ‘result’ structure is not computed; however, the computation can be faster (between 2 and 10 times faster, depending on the case) than the solution presented previously. It is however limited to the ‘2×2’ case and does not allow the computation of the fields nor that of the loss in each layer.

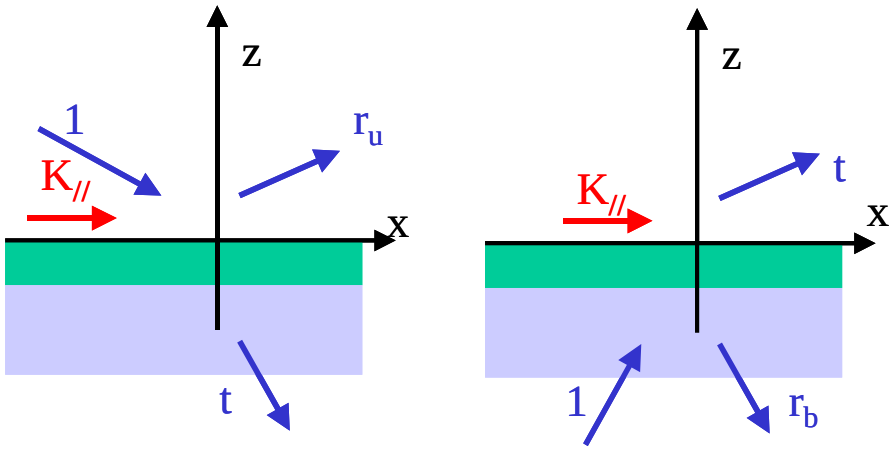


Fig. 7 For a fixed $\mathbf{K}_{//}$, the stack can be illuminated from the top or the bottom and this defines several reflection and transmission complex coefficients, ‘ru’, ‘t’ and ‘rb’. They are all computed together.

The definition of ‘ru’ and ‘rb’ conforms to the conventions of versions V9 and later of RETICOLO1D et 2D [11]. With the plane-wave normalization satisfying the reciprocity theorem, unlike the Fresnel formula of classical textbooks [12], we have tu = tb (ru and rb are the same as in textbooks, maybe to a minus sign).

The function res2_0D requires the following input parameters:

- ‘pol’ : pol=1 TE pol=-1 TM
- ‘n’: layer-index array ordered from top to bottom including superstrate and substrate. ‘n’ is an array of numbers (doubles)
- ‘h’: layer thickness ordered by descending order: h(1) is the thickness of the layer in contact with the top layer and h(end) is the thickness of the layer in contact with the bottom layer; note that length(h)=length(n) – 2 and that h=[] for a single interface.
- ‘beta’: the parameter is defined below (attention, avoid the confusion between beta and beta0=$K_{//}$)

1. Light is incident from the top

```
[ru,t]=res2_0D(pol,n,h,beta,k0,parm); % rb is not computed
```

a. parm=1 (or missing):

beta: array of beta=n(1)*sin(angle d'incidence) ($K_{//}$=k0*beta) ; size(beta)=[nb_k0,nb_inc].

b. parm=0: nb_k0=1;

beta: array of $K_{//}$; size(beta)=[1,nb_inc)].

c. parm=2:

beta: array of $K_{//}$; size(beta)=[nb_k0,nb inc]

For the case where the incident medium is dispersive: beta =(k0(:).*n(1)(ld(:)))*sin(teta). See the subsection 4 hereafter for the meaning of the function n(1).

2. Light is incident from the bottom

[~,t,rb]=res2_0D(pol,n,h,beta,k0,parm);

a. parm=1 (or missing):

beta: array of beta=n(end)*sin(angle d'incidence) (K//=k0*beta) ; size(beta)=[nb_k0,nb_inc].

b. parm=0: nb_k0=1;

beta: array of $K_{//}$; size(beta)=[1,nb_inc)].

c. parm=2:

beta: array of $K_{//}$; size(beta)=[nb_k0,nb inc].

For the case where the incident medium is dispersive: beta=(k0(:).*n(end)(ld(:)))*sin(teta). See the subsection 4 hereafter for the meaning of the function n(end).

3. Very important

'ru', 't' and 'rb' are the complex reflection and transmission coefficients of size [nb_k0, nb_inc].

Attention, $T = |t_u|^2 = |t_b|^2$ represents the transmission in intensity only if the transmitted waves are propagating plane waves. Suppose the plane wave transmitted downwards is evanescent, $|t_u|^2$ has no meaning from an energetic point of view and the absorption is simply $1 - |r_u|^2$, not $1 - |r_u|^2 - |t_u|^2$ as would be the case for a propagating transmitted wave. So, in general, the intensity transmitted downwards (for an illumination from above) is $T = |t|^2$ unless the transmitted wave is evanescent, which happens in 2 cases:

- if imag(n(end)(ld)~=0
- if n(1)(ld)*sin(teta)>n(end(ld)) (total internal reflection).

NB: In the general formalism in which the 'result' structure is computed, this does not have to be taken into account: result.efficiency is automatically set to 0. This is an advantage of the general form which is not implemented in the fast method.

4. To account for dispersion without anisotropy

'n' is an array of numbers; however, we can artificially use it to introduce dispersion and anisotropy.

For example, if the media 1 and 3 are dispersive, n(1) and n(3) are 'relay' numbers (one can provisionally take rand or any number) leading to functions of ld which will be evaluated during the computation.

Before executing the instruction [ru,t]=res2_0D (pol ...) , global variables need to be updated:

res2_0D([n(1),n(3)], @(ld) indice1(ld…), @(ld) indice2(ld…)

Please note that this order must only be executed once because it overwrites the previous ones. In the case of a parfor loop, it must be executed inside the loop and not outside the loop.

Figure 8 clarifies our purpose.

5. To additionally account for anisotropy

We limit ourselves to diagonal **ε** et **μ** tensors with $\varepsilon_{xx}=\varepsilon_{yy}$=indice_par^2, ε_{zz}=indice_per^2.

res2_0D ([n(1), n(3)], @(ld) indice1_par(ld…), @(ld) indice3_par(ld…), …
@(ld) indice1_per(ld…), @(ld) indice3_per(ld…)) ;

One may also introduce the relative permeability mu=indice_mu^2

res2_0D ([n(1), n(3)], @(ld) indice1_par(ld…), @(ld) indice3_par(ld…), …
@(ld) indice1_per(ld…), @(ld) indice3_per(ld…),...
@(ld) indice1_mu(ld…), @(ld) indice3_mu(ld…)) ;

or

```
res2_0D ( [n(1), n(3)], @(ld) indice1_par(ld…), @(ld) indice3_par(ld…), …
                        @(ld) indice1_per(ld…), @(ld) indice3_per(ld…),...
                        @(ld) indice1_mu_par(ld…), @(ld) indice3_mu_par(ld…),…
                        @(ld) indice1_mu_per(ld…), @(ld) indice3_mu_per(ld…) );
```

For example, in the example of case 5 in Section XI.1 (perfect lens),

```
res2_0D(n(3),@(ld)  n0*1i,  @(ld) n0*1i, @(ld)  1i);
```

we are obliged to successively introduce sqrt(eps_par), sqrt(eps_per), sqrt(mu), even if eps_par= eps_per.

Please note that the number of functions is 1*length([n(1), …]), or 2*length([n(1), …]), which means that if a single medium is anisotropic, we have to repeat the functions for all the mediums, even if they are isotropic:

```
res2_0D([n(1),n(2)],...
.@(ld) retindice(ld,505),@(ld) retindice(ld,505)),...  % BK7 isotrope
.@(ld) retindice(ld,516),@(ld) retindice(ld,517));    % hBn anisotrope
```

Again, some other examples:

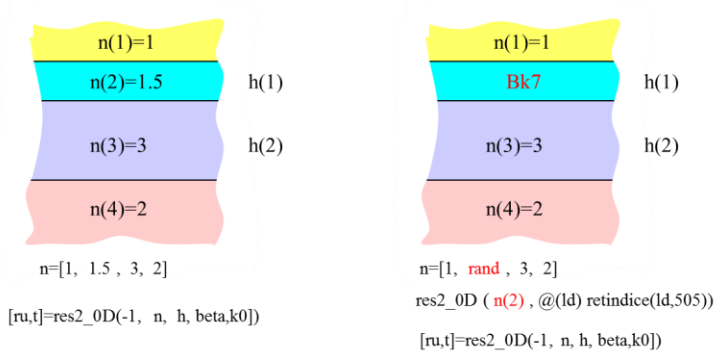


Fig. 8 Left : no dispersion, no anisotropy. Right: Dispersion without anisotropy.

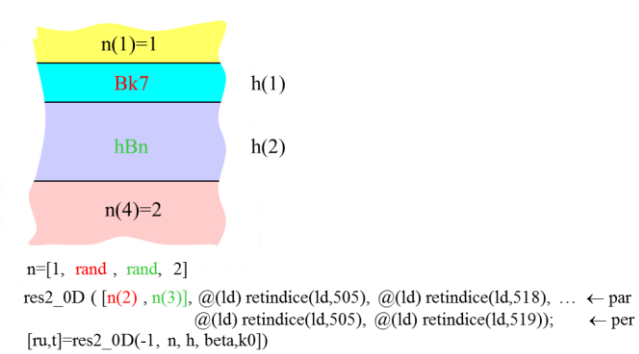
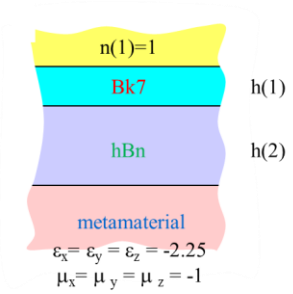
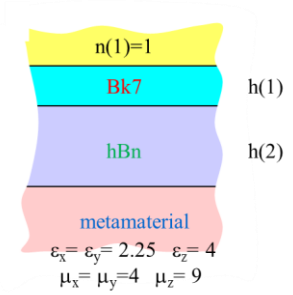


Fig. 9 Dispersion with anisotropy.



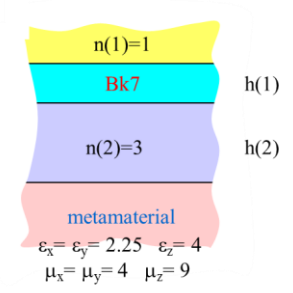

```
n=[1,  rand ,  rand,  rand]
res2_0D ( [n(2) , n(3), n(4)], @(ld) retindice(ld,505),  @(ld) retindice(ld,518),  1.5i…     ← par √ε
                               @(ld) retindice(ld,505),  @(ld) retindice(ld,519),  1.5i…     ← per  √ε
                                          1 ,                    1,                  1i  );        √μ
[ru,t]=res2_0D(-1,  n, h, beta,k0])
```

Fig. 10 Isotropic magnetic materials.




```
n=[1,  rand ,  rand,  rand]
res2_0D ( [n(2) , n(3), n(4)], @(ld) retindice(ld,505),  @(ld) retindice(ld,518),  1.5,…     ← par √εx =√εy
                               @(ld) retindice(ld,505),  @(ld) retindice(ld,519),  2,…       ← per  √εz
                                          1 ,                    1,                  2 ,…      ← √μx =√μy
                                          1,                     1,                  3 );      ← √μz
[ru,t]=res2_0D(-1,  n, h, beta,k0])
```

Fig. 11 Anisotropic magnetic materials.




```
n=[1,  rand ,  3,  rand]
res2_0D ( [n(2) , n(4)], @(ld) retindice(ld,505),   1.5,…     ← par √εx =√εy
                         @(ld) retindice(ld,505),   2,…       ← per  √εz
                                    1 ,             2 ,…      ← √μx =√μy
                                    1,              3 );      ← √μz
[ru,t]=res2_0D(-1,  n, h, beta,k0])
```

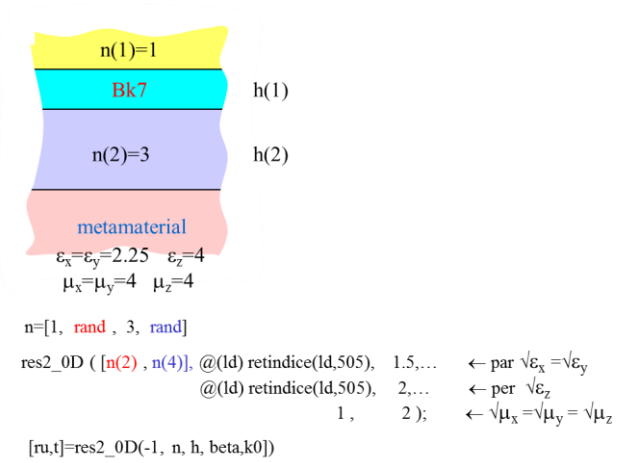


Fig. 12 Anisotropic magnetic and dielectric materials.

IX. Summary

Step 1 **Initialisation** parm=struct('option_degre',1"angle_cut',90)		
res0_0D(pol, k0, beta0, *parm*)	size(beta0)=[nb_k0, nb_inc] size(k0)=[1, nb_k0]	2×2
res0_0D(pol, k0, theta, n_inc, *parm*)	size(theta)=[1, nb_inc] size(k0)=[1, nb_k0]	
res0_0D(k0, beta0, *parm*)	size(beta0)=[nb_k0, nb_inc, 2] size(k0)=[1, nb_k0]	4×4
res0_0D(k0, theta, delta, n_inc, 1 , *parm*)	size(theta)= size(delta)=[1, nb_inc] size(k0)=[1, nb_k0]	
res0_0D(k0, theta, delta, n_inc, 0, *parm*)	nb_inc=length(theta)*length(delta) size(k0)=[1, nb_k0]	

Step 2 **Layer definition**		
a= res1_0D(n, p);	p=1 if the user wants to write themselves the boundary conditions at the top and bottom layers. p=0 otherwise	
n=1.5;		2×2 ou 4×4
n=@(ld) retindice(ld, 2.13);		
n=[vecteur des indices de longueur nb_k0]		
n={@(ld) retindice(ld,2.13), Omegap, Gama, beta};	Nonlocal hydrodynamic model	
n=@(ld){nx, ny, nz};	Diagonal anisotropy epsilon=diag([nx^2,ny^2,nz^2]) nx =cte or depends on ld	2×2
n=@(ld){nx, ny, nz , mx, my,mz};	Diagonal anisotropy epsilon=n^2 mu=m^2	
n={epsilon};	size(epsilon)=[3, 3] (nondispersive) size(epsilon)=[3, 3, nb_k0]	

n={epsilon, mu} ;	size(epsilon)=[3, 3], (nondispersive) size(epsilon)=[3, 3, nb_k0] size(mu)=[3, 3], (nondispersive) size(mu)=[3, 3, nb_k0]	4×4

Step 3 Propagation s=res2_0D(a,h)

Step 4 Assemblage s=s1*s2*(s3*s4*s5)^4*s6

Modes de Bloch abloch= res1_0D (S,h)

Step 5 Boundary conditions at the top and bottom layers		
[result,su,sb]=res2_0D(s,au,ab)	build 'result'	boundary condition
[result,su,sb,s_tot]=res2_0D(s,au,ab)	compute 's_tot' without computing 'result'	
su=retb(au,1); sb=retb(ab,-1); s_tot=su*s*sb	Alternative solution	
[result,su,sb]=res2_0D(s,au,abloch) etc...		periodic boundary condition

Compute and plot the field (option=12 corresponds to the abs(fields))		
[e,z,index]=res3_0D(x,aa,Profile,einc,*parm*);	compute	2×2
res3_0D(e,index,x,z,[1:3,option,1i])	plot	
[e,z,index]=res3_0D(x,y,aa,Profile,einc,*parm*);	compute	4×4
res3_0D(e,index,x,y,z,[1:6,option,1i])	plot	

Absorption loss
[Losses_per_layer,Flux_Poynting]=res3_0D(aa,Profile,einc,*parm*);

X. Additional information

1. T, S, G matrices

The '**T**' matrix formalism leads to numerical instabilities; we thus use the '**S**' matrix formalism [1].

In extreme cases (mixture of gain and loss, very large heights), the '**S**' matrices can become unstable. It is then possible to choose the '**G**' matrices [13] by adding parm=struct(.... 'sog',0) to the parameters of the res_0D function. The calculation time is increased. This option can only be use for the 4×4 case.

a. Propagation

T matrix $$\begin{pmatrix} E_T(z_2) \\ H_T(z_2) \end{pmatrix} = T \times \begin{pmatrix} E_T(z_1) \\ H_T(z_1) \end{pmatrix}$$

S matrix $$\begin{pmatrix} E_T(z_2) \\ H_T(z_1) \end{pmatrix} = S \times \begin{pmatrix} E_T(z_1) \\ H_T(z_2) \end{pmatrix}$$

G matrix $$G_2 \times \begin{pmatrix} E_T(z_2) \\ H_T(z_2) \end{pmatrix} = G_1 \times \begin{pmatrix} E_T(z_1) \\ H_T(z_1) \end{pmatrix}$$

Fig. 13 Different matrices to compute propagation from ‘z1’ to ‘z2’. The **T** matrix is unstable [1]. The **G** matrix [13] is not well known but is highly stable and general.

b. Boundary conditions

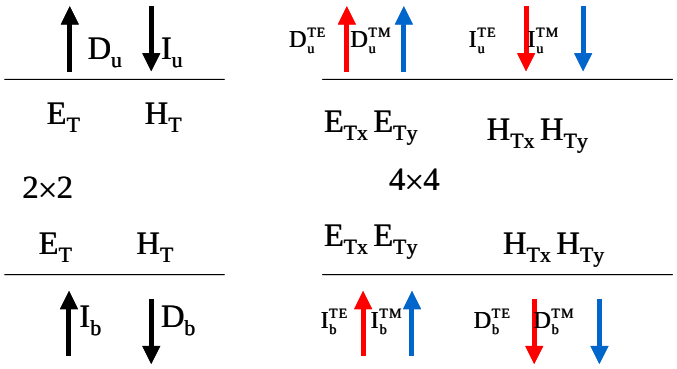


Fig. 14 Definition of the Incident (denoted ‘I’) and Scattered (denoted ‘D’ for diffracted) plane waves.

	Up	bottom
T matrix	$\begin{pmatrix} D_u \\ I_h \end{pmatrix} = T_u \times \begin{pmatrix} E_T(z_u) \\ H_T(z_u) \end{pmatrix}$	$\begin{pmatrix} E_T(z_b) \\ H_T(z_b) \end{pmatrix} = T_b \times \begin{pmatrix} I_b \\ D_b \end{pmatrix}$
S matrix	$\begin{pmatrix} D_u \\ H_T(z_u) \end{pmatrix} = S_u \times \begin{pmatrix} E_T(z_u) \\ I_u \end{pmatrix}$	$\begin{pmatrix} E_T(z_b) \\ D_b \end{pmatrix} = S_b \times \begin{pmatrix} I_b \\ H_T(z_b) \end{pmatrix}$
G matrix	$G_u^2 \times \begin{pmatrix} D_u \\ I_u \end{pmatrix} = G_h^1 \times \begin{pmatrix} E_T(z_u) \\ H_T(z_u) \end{pmatrix}$	$G_b^2 \times \begin{pmatrix} E_T(z_b) \\ H_T(z_b) \end{pmatrix} = G_b^1 \times \begin{pmatrix} I_b \\ D_b \end{pmatrix}$

Fig. 15 Boundary conditions with the T, S and G matrices.

2. Definition of circular polarizations

Figures 16 and 17 define a plane wave circularly polarized right or left according to the direction of the helix described by the vertex of the vector Real(E) (green curve).

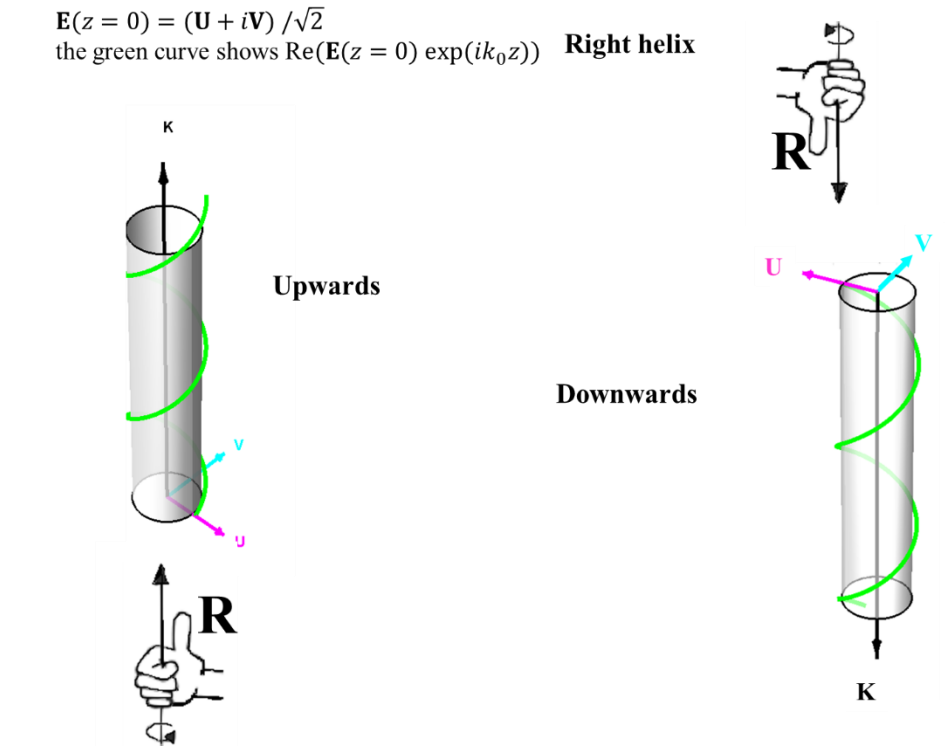


Fig. 16

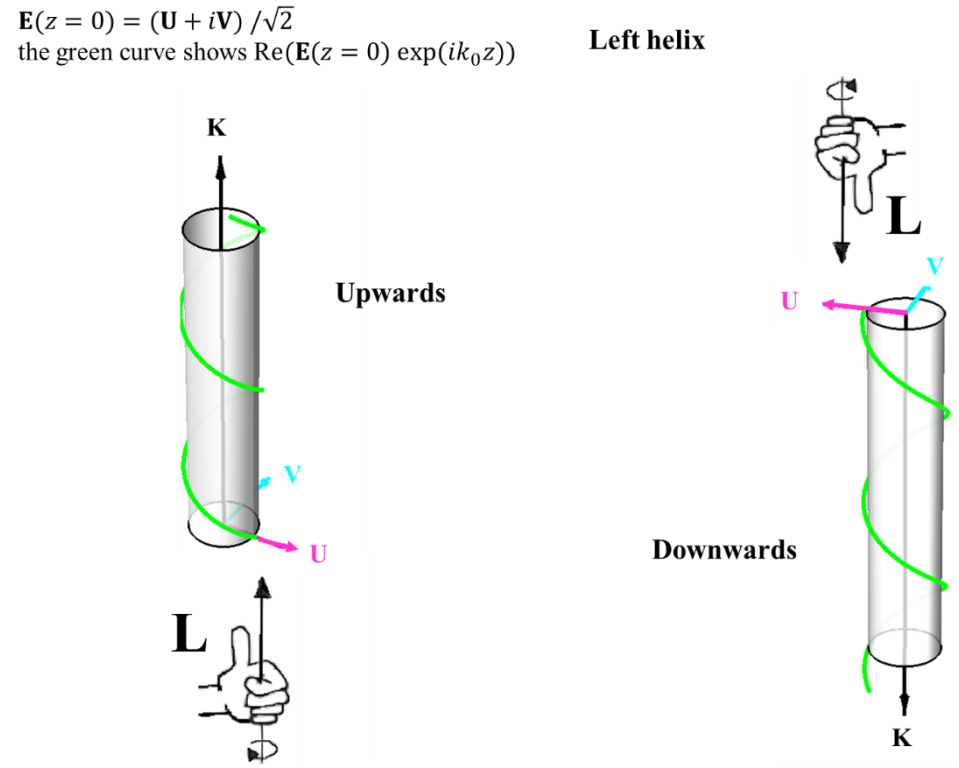


Fig. 17

Figures 18 and 19 make it possible to express the incident or diffracted plane waves (right column) according to the 'PlanesWaves' defined by the 'result' structure (2 left columns). Figures 16 and 17 are limited to right circular polarization. The left circular case is obtained by changing i to $-i$.

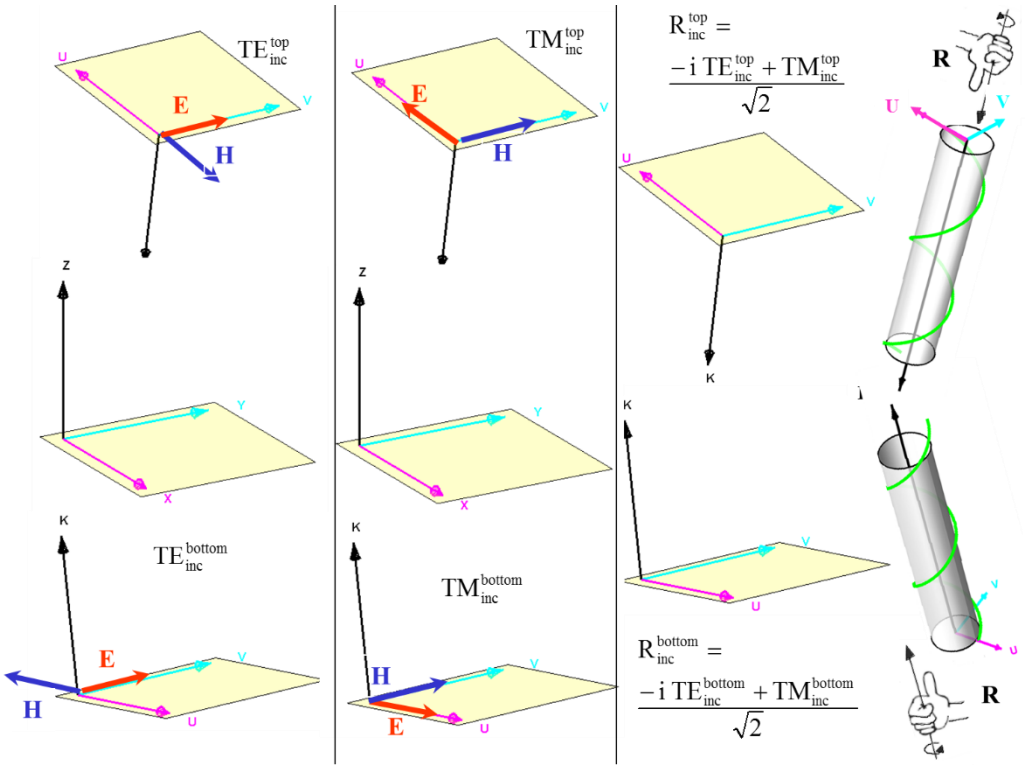


Fig 18

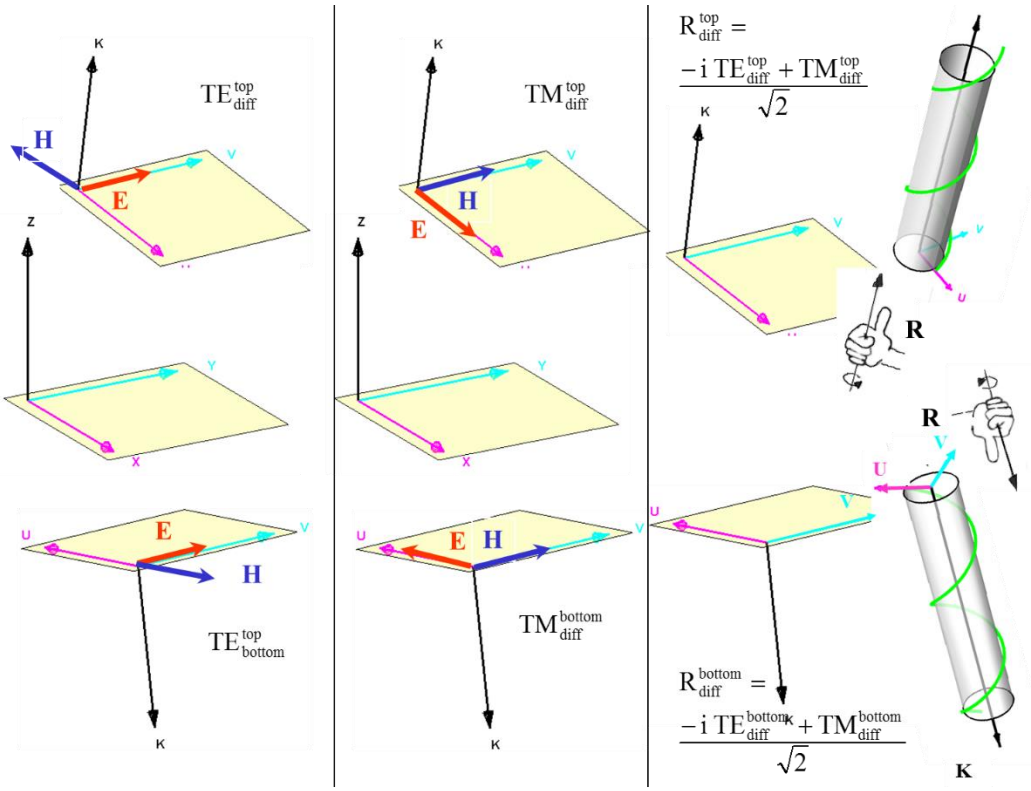


Fig. 19

We therefore have the passage matrices:

$$\begin{pmatrix} TE_{inc}^{bottom} \\ TM_{inc}^{bottom} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} -i & 1 \\ i & 1 \end{pmatrix} \times \begin{pmatrix} R_{inc}^{bottom} \\ L_{inc}^{bottom} \end{pmatrix} \qquad \begin{pmatrix} TE_{inc}^{top} \\ TM_{inc}^{top} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} -i & 1 \\ i & 1 \end{pmatrix} \times \begin{pmatrix} R_{inc}^{top} \\ L_{inc}^{top} \end{pmatrix}$$

$$\begin{pmatrix} TE_{diff}^{bottom} \\ TM_{diff}^{bottom} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} -i & 1 \\ i & 1 \end{pmatrix} \times \begin{pmatrix} R_{diff}^{bottom} \\ L_{diff}^{bottom} \end{pmatrix} \qquad \begin{pmatrix} TE_{diff}^{top} \\ TM_{diff}^{top} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} -i & 1 \\ i & 1 \end{pmatrix} \times \begin{pmatrix} R_{diff}^{top} \\ L_{diff}^{top} \end{pmatrix}$$

Transition from the rectangular Jones matrix **J** to the circular Jones matrix $\mathbf{J_C}$

$$\begin{pmatrix} J_C\{1,1\} \\ J_C\{2,1\} \\ J_C\{1,2\} \\ J_C\{2,2\} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & -i & i & 1 \\ -1 & -i & -i & 1 \\ -1 & i & i & 1 \\ 1 & i & -i & 1 \end{pmatrix} \times \begin{pmatrix} J\{1,1\} \\ J\{2,1\} \\ J\{1,2\} \\ J\{2,2\} \end{pmatrix}$$

$$\begin{pmatrix} TE_{diff} \\ TM_{diff} \end{pmatrix} = \mathbf{J} \times \begin{pmatrix} TE_{inc} \\ TM_{inc} \end{pmatrix}, \qquad \begin{pmatrix} R_{diff} \\ L_{diff} \end{pmatrix} = \mathbf{J_C} \times \begin{pmatrix} R_{inc} \\ L_{inc} \end{pmatrix}$$

3. Stokes parameters

For instance, for the field of the 'result' structure

ch=result.TEinc_top_transmitted

S0=I=abs(ch.amplitude_TE).^2+ abs(ch.amplitude_TM).^2;

S1=Q=abs(ch.amplitude_TE).^2- abs(ch.amplitude_TM).^2;

S2=U=2*real(ch.amplitude_TE.*conj(ch.amplitude_TM));

S3=V= -2*imag(ch.amplitude_TE.*conj(ch.amplitude_TM));

4. Cuts in the complex plane

In the top and bottom media, where outgoing wave conditions are written, the propagation constant in *z* is defined by $\chi^2 = k_0^2 n^2 - \beta_0^2$. This equation has two square root solutions: $\pm\chi$. One of these solutions corresponds to an outgoing wave, the other to an incoming wave. They should be determined to write the boundary conditions.

For that, it remains to define the square root of any complex number Z [14]. A simple solution is to introduce a cutoff (discontinuity of the function z(Z)) in the complex plane from an angle θ ($0 \le \theta < \pi$) , to write $Z = |Z|\, exp(i\phi)$ with $-\theta \le \phi < \pi - \theta$ and to choose:$z = \sqrt{|Z|}\, exp\left(i\frac{\phi}{2}\right) = \sqrt[\theta]{Z}$.

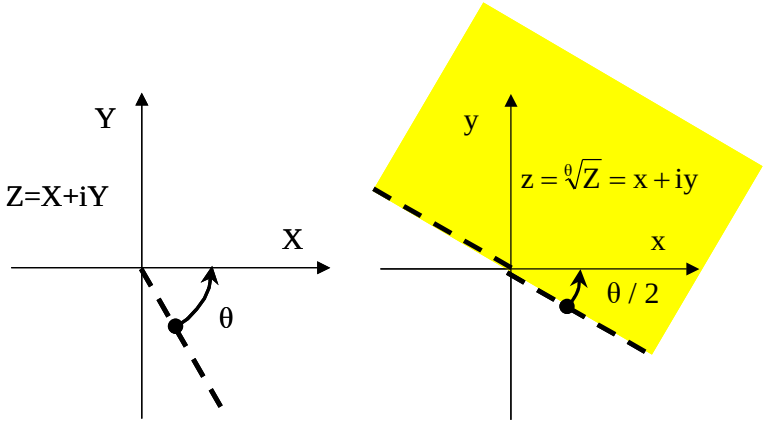


Fig. 20 Square roots of complex numbers (to be used to satisfy the boundary conditions in the top and bottom media).

We thus define: $\chi = \sqrt[\theta]{k_0^2 n^2 - \beta_0^2}$

For $\theta = 0$, we eliminate the increasing waves at infinity; however, this does not allow us to compute the 'leaky' modes like those of the plasmon couplers in the Otto or Kretschmann configurations. The default cut is Maystre's cut with $\theta = \pi/2$ (see Fig. 20 for a definition of θ).

5. General construction of symmetric epsilon tensors without gain

One requires that $\mathrm{Im}(\boldsymbol{\varepsilon})$ is a positive matrix.

For example:

epsilon=randn(3);epsilon=epsilon+epsilon.';H=randn(3);epsilon=epsilon+1i*H'*H;

6. Other programs used in the examples

a. retindice.m : a refractive index database

parameter=4.116; n_Al2O3=retindice(ld, parameter); % ld, the wavelength in microns (Al2O3)

parameter=2; n_Au=retindice(ld, parameter); % ld, the wavelength in microns (Gold)

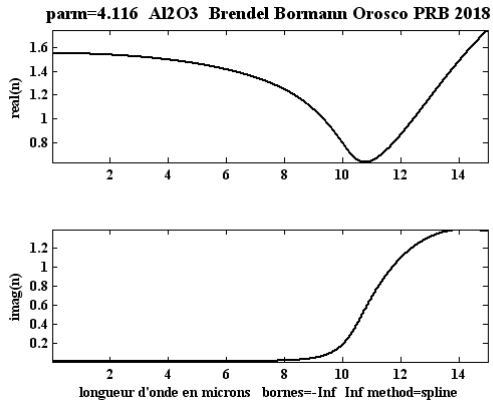
In general, several options per material are available. To list the different options for Al2O3 :

retindice('al2o3')

```
4.11     [-Inf     Inf]  Al2O3 extraordinaire Sellmeier equation, transmis par Mathieu
4.111    [-Inf     Inf]  Al2O3 Sellmeier equation Transmis par S Collin I. H. Malitson
4.112    [-Inf     Inf]  Al2O3 ordinaire Sellmeier equation, transmis par S Collin
4.113    [-Inf     Inf]  Al2O3_ordinaire_Barker_1 PR 132 4 1963 transmis par PBA
4.114    [-Inf     Inf]  Al2O3_extraordinaire_Barker_1 PR 132 4 1963 transmis par
4.115    [-Inf     Inf]  Al2O3_ordinaire_Barker_2 PR 132 4 1963 transmis par PBA
4.116    [-Inf     Inf]  Al2O3  Brendel Bormann Orosco PRB 2018
```

To know to which material corresponds a parameter

retindice([],[4.116])



We decline any responsibility for errors contained in 'retindice.m'.

b. retgauss.m : integral computation with Gauss method

[x,wx]=retgauss(0,pi,10,5); % 50 Gauss points are generated in the interval from 0 to \pi.

sum(wx.*sin(x).^2) - pi/2

ans =

8.8818e-16

c. retcadilhac.m : search for the complex zeros of a complex function zz(z)=0

```
start=0;teta=retcadilhac(@(teta)cos(teta)-(.8+.2i),[],start); cos(teta)
```

d. other fonctions used in the examples of Section XI

```
retfont(gcf,0)       % improve the fonts of the curves
retcolor(x,y,Tab) % similar to pcolor: 2D image of Tab,  size(tab)=[length(y) ,length(x)]
retdb.m              % transform a wavelength in micron into another unit (cm^(-1) THz ...) and vice versa,
retcompare(x,y)   % compute the error between x and y
retcauchy.m        % Visualize poles and cuts in a tabulation of a complex function of complex variable
```

7. Vectorization principe

Let's take as a simple example a product of 2×2 matrices:

$$A=\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad B=\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \quad AB=\begin{pmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} \end{pmatrix}$$

Now the elements are cel arrays and the a and b are arrays of size nb_k0 × nb_inc

$$A=\begin{Bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{Bmatrix} \quad B=\begin{Bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{Bmatrix} \quad AB=\begin{Bmatrix} a_{11}.*b_{11}+a_{12}.*b_{21} & a_{11}.*b_{12}+a_{12}.*b_{22} \\ a_{21}.*b_{11}+a_{22}.*b_{21} & a_{21}.*b_{12}+a_{22}.*b_{22} \end{Bmatrix}$$

We have formulas of the same style for inversions and products of matrices S. For 2×2 S matrix product, s=sss*ss can be computed with the following code

```
s=cell(2,2);
s{1,2}=1./(1-ss{1,2}.*sss{2,1});
ss{1,1}=ss{1,1}.*s{1,2};
sss{2,2}=sss{2,2}.*s{1,2};
s{1,1}=sss{1,1}.*ss{1,1};
s{2,2}=ss{2,2}.*sss{2,2};
s{2,1}=sss{2,1}.*ss{2,2}.*ss{1,1}+ss{2,1};
s{1,2}=sss{1,1}.*ss{1,2}.*sss{2,2}+sss{1,2};
```

8. Remark on the implementation of nonlocal models

RETICOLOfilm-stack can handle spatial dispersion (nonlocality) at the interfaces. It implements the hydrodynamic model of Ref. [6], in which the concepts of lateral waves (‘e_onde_lateral’) and transversal waves (‘e0’) involved in the total field, e=e0+e_onde_lateral, are clearly presented. The program ‘Antoine_Bulk_plasmon.m’ provides an example. Please contact Jean Paul Hugonin if you are interested.

The hydrodynamic model includes the parameters Omegap et Gama (eV), and beta (eV.µm).

```
n={@(ld) retindice(ld,2.13), Omegap, Gama, beta};
```

The function of ld which gives the index can be replaced by a row vector of length nb_k0 or even simply by a constant.

XI. Examples

1. The following examples can be copied and executed in Matlab

a. ‘2×2’ absorption per layer and field plot

In this section, we take the examples of the paragraph 'simplified solution' with the 2×2 formalism that allows the calculation of the losses and field plots.

- fast solution

```
for cas=1:5;
switch cas
case 1; % without dispersion
%
%      1.2
% ***************
% ***** 4+1i **** h
% ***************
% .............
% ....  1  ......
% .............

n=[1.2,4+1i,1];h=.1;ld=linspace(.3,.6,100);k0=2*pi./ld;pol=1;
teta=linspace(0,89,200); % teta=90 is problematical
beta=n(1)*sind(teta);
[ru,t]=res2_0D(pol,n,h,beta,k0);
R=abs(ru).^2;
T=abs(t).^2;T(n(1)+0*ld.'*sind(teta)>n(3))=0;% very important
Losses=1-R-T;
figure;retcolor(teta,ld,R);retfont(gcf,0);
figure;retcolor(teta,ld,Losses);retfont(gcf,0);

case 2; % with dispersion  BK7/Ag/air
%
%      BK7 retindice(ld,505)
% ****************
% ***** Ag ****** h
% ****************
% .............
% ....  1  ......
% .............
n=[rand,rand,1];
%rand is used for initialisation before introducing dispersion
h=.1;ld=linspace(.3,.6,100);k0=2*pi./ld;pol=-1;% TM
teta=linspace(0,89,200);beta=(k0(:).*retindice(ld(:),505))*sind(teta);
res2_0D([n(1),n(2)],@(ld) retindice(ld,505),@(ld)retindice(ld,1));
% dispersion
[ru,t]=res2_0D(pol,n,h,beta,k0,2);
R=abs(ru).^2;
T=abs(t).^2;
T(retindice(ld,505).'*sind(teta)>1)=0;% very important
figure;retcolor(teta,ld,R);retfont(gcf,0);
Losses=1-R-T;
```

```
figure;retcolor(teta,ld, Losses);retfont(gcf,0);

case 3; % Single interface BK7/Ag
%
%      BK7
% ***************
% ***** Ag ******
% ***************

n=[rand,rand];
 % rand is used for initialization, before introducing dispersion
ld=linspace(.3,.6,100);k0=2*pi./ld;pol=-1;
teta=linspace(0,89,200);beta=(k0(:).*retindice(ld(:),505))*sind(teta);
res2_0D([n(1),n(2)],@(ld) retindice(ld,505),@(ld)retindice(ld,1));
% dispersion
[ru,t]=res2_0D(pol,n,[],beta,k0,2);
R=abs(ru).^2;
T=abs(t).^2;T(imag(retindice(ld,1))~= 0,:) = 0;
figure;retcolor(teta,ld,R);retfont(gcf,0);
Losses=1-R-T;
figure;retcolor(teta,ld, Losses);retfont(gcf,0);

case 4; % with dispersion diagonal anisotropy, air hBN glass
%
%      air
% ****************
% ***** hBN ****** h
% ****************
% ..............
% .... 1.5 ......
% ..............

n=[1,rand,1.5];h=.1;ld=linspace(.3,.6,100);k0=2*pi./ld;pol=-1;
teta=linspace(0,89,200);beta=1*sind(teta);
%                  sqrt(\eps_x=\eps_y),     sqrt(\eps_z)
res2_0D(n(2),@(ld) retindice(ld,516),@(ld) retindice(ld,517));
[ru,t]=res2_0D(pol,n,h,beta,k0);
R=abs(ru).^2;
T=abs(t).^2;
figure;retcolor(teta,ld,R);retfont(gcf,0);
Losses=1-R-T;
figure;retcolor(teta,ld, Losses);retfont(gcf,0);

case 5; % Perfect Lens
```

```
h=.1;n0=1.5;
n=[n0,n0,rand,n0,n0];ld=linspace(.3,.6,100);k0=2*pi./ld;pol=-1;
teta=linspace(0,89,200);beta=k0(:).*n0*sind(teta);
%       sqrt(\eps_x=\eps_y), sqrt(\eps_z) ,sqrt(\mu)
res2_0D (n(3),@(ld)  n0*1i,  @(ld) n0*1i, @(ld)  1i);
% or equivalently
% res2_0D (n(3),@(ld)  n0*1i,  @(ld) n0*1i, @(ld)  1i, @(ld)  1i);
[ru,t]=res2_0D(pol,n,[h,h,0],beta,k0,2);
test_r0=max(abs(ru(:)))
test_t1=max(abs(t(:)-1))
end;
end;
```

- ‘2×2’ : absorption per layer and field plot

We take again the examples of the previous section with the 2×2 formalism that allows the loss computation and the plot of the electromagnetic fields, and provide other examples.

```
plot_field=1; % in option
for cas=1:5;
switch cas
case 1; % without dispersion
%
%      1.2         (au)
% ****************
% ***** 4+1i ****  (ac)    h
% ****************
% ................
% .... 1 ........  (ab)
% ................

n=[1.2,4+1i,1];h=.1;ld=linspace(.3,.6,100);k0=2*pi./ld;pol=1;% TE
teta=linspace(0,89,200);beta=n(1)*sind(teta);

res0_0D(pol,k0,teta,n(1),struct('option_degre',1));
au=res1_0D(n(1));
ac=res1_0D(n(2));
ab=res1_0D(n(3));
s=res2_0D(ac,h);
result=res2_0D(s,au,ab);
R=result.inc_top_reflected.efficiency;
T=result.inc_top_transmitted.efficiency;
Losses=1-R-T;
figure;retcolor(teta,ld,R);retfont(gcf,0);
figure;retcolor(teta,ld,Losses);retfont(gcf,0);
Profile={[0,h,0],[1,2,3]}; % compute the loss only in layer 2
```

```
[Losses_per_layer,Flux_Poynting]=res3_0D({au,ac,ab},Profile,1);
test_Losses=retcompare(Losses,sum(Losses_per_layer,3))

if plot_field;% in option
Profile={[h,h,h],[1,2,3]};
npts=[[10,10,10];[5,5,5]];
x=0;[e,z,index]=res3_0D(x,{au,ac,ab},Profile,1,struct('npts',npts));
i_k0=retminabs(ld,.5);% the plot is performed at a wavelength close to 0.5
i_inc=retminabs(teta,60);% for an incidence angle close to 60 degree
res3_0D(e(:,:,:,i_k0,i_inc),index(:,:,i_k0),x,z,[1:3,12,1i]);
end;

case 2; % with dispersion  BK7/Ag/air
%
%      BK7
% ***************
% ***** Ag ******  h
% ***************
% ..............
% .... 1   ......
% ..............


h=.1;ld=linspace(.3,.6,100);k0=2*pi./ld;pol=-1;
teta=linspace(0,89,200);beta=(k0(:).*retindice(ld(:),505))*sind(teta);
nu=@(ld) real(retindice(ld,505));% BK7

res0_0D(pol,k0,teta,nu,struct('option_degre',1));

nc=@(ld) retindice(ld,1);% Ag
nb=1;
au=res1_0D(nu);
ac=res1_0D(nc);
ab=res1_0D(nb);
s=res2_0D(ac,h);
result=res2_0D(s,au,ab);
R=result.inc_top_reflected.efficiency;
T=result.inc_top_transmitted.efficiency;

figure;retcolor(teta,ld,R);retfont(gcf,0);
Losses=1-R-T;
figure;retcolor(teta,ld, Losses);retfont(gcf,0);
Profile={[0,h,0],[1,2,3]};
[Losses_per_layer,Flux_Poynting]=res3_0D({au,ac,ab},Profile,1);
```

```
test_Losses=retcompare(Losses,sum(Losses_per_layer,3))
if plot_field
Profile={[h,h,h],[1,2,3]};
npts=[[10,10,10];[5,5,5]];
x=0;[e,z,index]=res3_0D(x,{au,ac,ab},Profile,1,struct('npts',npts));
i_k0=retminabs(ld,.5);i_inc=retminabs(teta,60);
res3_0D(e(:,:,:,i_k0,i_inc),index(:,:,i_k0),x,z,[1:3,12,1i]);
end;

case 3; %   single interface BK7/Ag
%
%      BK7
% ***************
% ***** Ag ******   h
% ***************

ld=linspace(.3,.6,100);k0=2*pi./ld;pol=-1;
teta=linspace(0,89,200);beta=(k0(:).*retindice(ld(:),505))*sind(teta);

nu=@(ld) real(retindice(ld,505));
nb=@(ld)retindice(ld,1);
res0_0D(pol,k0,teta,nu,struct('option_degre',1));

au=res1_0D(nu);
ab=res1_0D(nb);
result=res2_0D([],au,ab);
R=result.inc_top_reflected.efficiency;
T=result.inc_top_transmitted.efficiency;
figure;retcolor(teta,ld,R);retfont(gcf,0);
Losses=1-R-T;
figure;retcolor(teta,ld, Losses);retfont(gcf,0);

Profile={[0,0],[1,2]};
[Losses_per_layer,Flux_Poynting]=res3_0D({au,ab},Profile,1);
test_Losses=retcompare(Losses,-2*Flux_Poynting(:,:,end))
if plot_field
Profile={[.5,.1],[1,2]};
npts=[[10,10];[5,5]];
x=0;[e,z,index]=res3_0D(x,{au,ab},Profile,1,struct('npts',npts));
i_k0=retminabs(ld,.5);i_inc=retminabs(teta,60);
res3_0D(e(:,:,:,i_k0,i_inc),index(:,:,i_k0),x,z,[1:3,12,1i]);
end;

case 4; % with dispersion & diagonal anisotropy, air/hBN/glass
```

```
%
%      air
% ****************
% ***** hBN ****** h
% ****************
% ................
% .... 1.5   ......
% ................

nu=1;
nc=@(ld) { retindice(ld,516),[], retindice(ld,517)};
nb=1.5;
h=.1;ld=linspace(.3,.6,100);k0=2*pi./ld;teta=linspace(0,89,200);pol=-1; %TM

res0_0D(pol,k0,teta,nu,struct('option_degre',1));

au=res1_0D(nu);
ac=res1_0D(nc);
ab=res1_0D(nb);
s=res2_0D(ac,h);
result=res2_0D(s,au,ab);
R=result.inc_top_reflected.efficiency;
T=result.inc_top_transmitted.efficiency;
figure;retcolor(teta,ld,R);retfont(gcf,0);
Losses=1-R-T;
figure;retcolor(teta,ld, Losses);retfont(gcf,0);
Profile={[0,h,0],[1,2,3]};
[Losses_per_layer,Flux_Poynting]=res3_0D({au,ac,ab},Profile,1);
test_Losses=retcompare(Losses,sum(Losses_per_layer,3))

if plot_field
Profile={[.5,.1],[1,2]};
npts=[[10,10];[5,5]];
x=0;[e,z,index]=res3_0D(x,{au,ab},Profile,1,struct('npts',npts));
i_k0=retminabs(ld,.5);i_inc=retminabs(teta,60);
res3_0D(e(:,:,:,i_k0,i_inc),index(:,:,i_k0),x,z,[1:3,12,1i]);
end;

case 5; % Perfect lens
h=1;n0=1.5;
ld=linspace(.3,.6,100);k0=2*pi./ld;
teta=linspace(0,89,200);
pol=-1; %TM
nu=n0;
```

```
nc=@(ld) {n0*1i,[],n0*1i,1i,[],1i};
nb=n0;

res0_0D(pol,k0,teta,nu,struct('option_degre',1));

au=res1_0D(nu);
ac=res1_0D(nc);
ab=res1_0D(nb);
s=res2_0D(au,h)*res2_0D(ac,h);
result=res2_0D(s,au,ab);
ru=result.inc_top_reflected.amplitude;
t=result.inc_top_transmitted.amplitude;
test_r0=max(abs(ru(:)))
test_t1=max(abs(t(:)-1))

if plot_field
Profile={[h,h,h],[1,2,3]};
npts=[[10,10,10];[5,5,5]];
x=0;linspace(-h,h,100);
[e,z,index]=res3_0D(x,{au,ac,ab},Profile,1,struct('npts',npts));
i_k0=retminabs(ld,.5);i_inc=retminabs(teta,10);
res3_0D(e(:,:,:,i_k0,i_inc),index(:,:,i_k0),x,z,[1:3,13,1i]);
end;
end;
end;
```

b. 4×4 circular Left and Right

```
% Circular_Left_and_Right
%
%       nu=1
% @@@@@@@@@@@@@@@@@@@@
% @@@@@ eps_mat @@@@@@  h_mat
% @@@@@@@@@@@@@@@@@@@@
% .....................
% .....................
% ......n_spacer..... h_spacer
% .....................
% .....................
% ********************
% ********************
% *******  Au  *******
% ********************
```

```
% This code computes the optimisations obtained by Anne Nguyen during her thesis
lld=linspace(3,6,50);k0=2*pi./lld;
n_spacer=@(ld) retindice(ld,4.15);% SiNx
% Drude gold fitted from Palik data
eps0=1;omegap=5.8202;gama=0.0455;
n_Au=@(ld)retsqrt(eps0-omegap.^2*(ld.^2)./(1+1i*gama*ld),-1);
% Drude fit Palik
% optimise Losses L
eps_mat =[[-0.0400+0.2222i -30.3324-3.6779i        0];...
 [-30.3324-3.6779i  -7.3974+60.8745i        0];
 [       0                 0            0.0529+0.8073i]];
h_mat =  0.0250;
h_spacer =  0.6510;

theta=0;delta=0;
nu=1;
nb=n_Au;
n_mat={eps_mat};
res0_0D(k0,theta,delta,nu,0,struct('option_degre',1));

au=res1_0D(nu);
a_mat=res1_0D(n_mat);
a_spacer=res1_0D(n_spacer);
ab=res1_0D(nb);

s_mat=res2_0D(a_mat,h_mat);
s=s_mat*res2_0D(a_spacer,h_spacer);
result=res2_0D(s,au,ab);
% RCP incident light
R=result.Jones.inc_top_reflected.efficiency_R_2_R+...
result.Jones.inc_top_reflected.efficiency_R_2_L;
Losses_R=1-R;L_R=Losses_R(retminabs(lld,4.7))
Profile={[0,h_mat,h_spacer,0],[1,2,3,4]};
[Losses_per_layer,Poynting]=...
res3_0D({au,a_mat,a_spacer,ab},Profile,[1,-1i]/sqrt(2));
er_R=retcompare(Losses_R,sum(Losses_per_layer,3)-2*Poynting(:,:,end))
% LCP incident light;

R=result.Jones.inc_top_reflected.efficiency_L_2_R+...
    result.Jones.inc_top_reflected.efficiency_L_2_L;
Losses_L=1-R;L_L=Losses_L(retminabs(lld,4.7))
Profile={[0,h_mat,h_spacer,0],[1,2,3,4]};
[Losses_per_layer,Poynting]=res3_0D({au,a_mat,a_spacer,ab},Profile,[1,1i]/sqrt(2));
er_l=retcompare(Losses_L,sum(Losses_per_layer,3)-2*Poynting(:,:,end))
```

```
figure;plot(lld,Losses_R,'-k',lld,Losses_L,'-r','linewidth',2);
legend('losses R','losses L');xlabel('ld');retfont(gcf,0);
title(rettexte(h_mat,h_spacer));
```

2. Reproducing some results of the literature with RETICOLOfilm-stack

In this section we present some examples which show how RETICOLOfilm-stack recovers available results from the literature. We thus provide a set of Matlab programs. The numerical data (Matlab figures) generated by these programs are gathered in examples_test.pdf.

a. PyLlama [2]

run visu_PyLlama execute the following .m codes

PyLlama_fig4b

PyLlama_fig4d

PyLlama_fig5

PyLlama_fig6

PyLlama_fig7

PyLlama_figA9

b. Passler [3]

run visu_Passler.m

Passler_Field_and_Losses_1D

Passler_Field_and_Losses_2D

Passler_GaNm45

Passler

c. Antoine Moreau, Moosh [4,6,7,8]

run visu_Antoine_Moreau.m

Antoine_Moreau_miroir_de_bragg_non_local

Antoine_Moreau_refraction_negative (also perfect lens)

Antoine_Bulk_plasmon

d. Other codes

run visu.m

coupleur_Otto

coupleur_Kretschmann

miroir_de_bragg

resonateur_de_bragg

resonateur_de_bragg_pulse_Gaussien

Gaussien_beam_on_PML

MCD [5]

And, also, the codes in the text

fast_solution

Losses_and_fields

Circular_Left_and_Right

XII. Acknowledgements

J.-P. Hugonin thanks Antoine Moreau for fruitful discussions and advice on the hydrodynamic model and Kevin Vynck for his comparisons with the PyLlama software.

XIII. References

1. L. Li, Formulation and comparison of two recursive matrix algorithms for modeling layered diffraction gratings, J. Opt. Soc. Am. A **13**, 1024-1035 (1996).
2. M. Bay, S. Vignolini, K. Vynck, PyLlama: A stable and versatile Python toolkit for the electromagnetic modelling of multilayered anisotropic media, Computer Physics Communications **273**, 108256 (2022).
3. N.C. Passler, M. Jeannin and A. Paarmann, Layer-Resolved Absorption of Light in Arbitrarily Aniotropic Heterostructures, Phys. Rev. B **101**, 165425 (2020), https://doi.org/10.5281/zenodo.3648040 (Matlab)
4. J. Defrance, C. Lemaître, R. Ajib, J. Benedicto, E. Mallet, R. Pollès, J.-P. Plumey, M. Mihailovic, E. Centeno, C. Ciracì, D.R. Smith, and A. Moreau, Moosh: A Numerical Swiss Army Knife for the Optics of Multilayers in Octave/Matlab, Journal of Open Research Software **4**, p.e13 (2016) http://doi.org/10.5334/jors.100
5. O. ElGawhary, M.C. Dheur, S.F. Pereira, J.J.M. Brat, Extension of the classical Fabry-Perot formula to 1D multilayered structures, Appl. Phys. B **111**, 637-645 (2013).
6. A. Moreau, C. Ciraci, and D. R. Smith, Impact of nonlocal response on metallodielectric multilayers and optical patch antennas, Phys. Rev. B **87**, 045401 (2013).
7. Armel Pitelet. Théorie et simulation en nanophotonique : non-localité dans les nanostructures métalliques. PhD Université Clermont Auvergne, 2018.
8. E. Sakat, A. Moreau, and J.-P. Hugonin, Generalized electromagnetic theorems for nonlocal plasmonics, Phys. Rev. B **103**, 235422 (2021).
9. J. B. Pendry, Negative Refraction Makes a Perfect Lens, Phys. Rev. Lett. **85**, 3966 (2000).
10. Q. Cao, P. Lalanne, J.-P. Hugonin, Stable and efficient Bloch-mode computational method for one-dimensional grating waveguides, J. Opt. Soc. Am. A **19**, 335-338 (2002).
11. J.-P. Hugonin, P. Lalanne, Reticolo software for grating analysis, arXiv:2101.00901.
12. P. Yeh, *Optical waves in layered media*, J. Wiley and Sons eds., New York, 1988.
13. J.-P. Hugonin, M. Besbes, P. Lalanne, Hybridization of electromagnetic numerical methods through the G-matrix algorithm, Opt. Lett. **33**, 1590-1592 (2008).
14. https://en.wikipedia.org/wiki/Square_root#Square_roots_of_negative_and_complex_numbers