RETOP: an open-source for retrieving the free-space and guided-mode radiation diagrams of inhomogeneities embedded in thin-film stacks

Jianji Yang[𝕃1], **Jean-Paul Hugonin**[Σ], **and Philippe Lalanne**[𝕃]

[𝕃] LP2N, Institut d'Optique d'Aquitaine, IOGS, Univ. Bordeaux, CNRS.

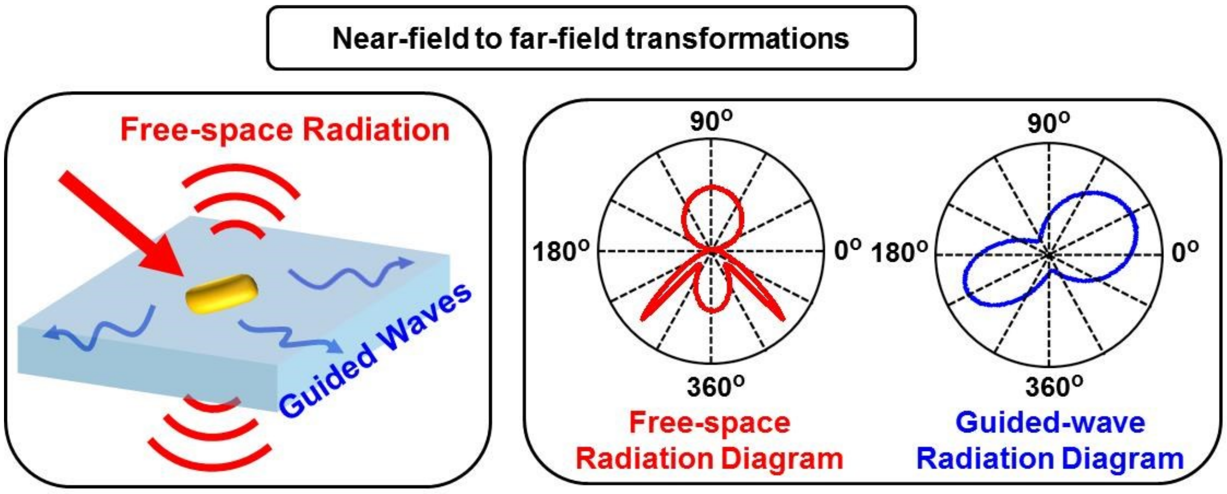
[Σ] LCF, IOGS, Univ. Paris-Sud, CNRS

jianji.yang.photonique@gmail.com & jjyang10@stanford.edu

jean-paul.hugonin@institutoptique.fr

philippe.lalanne@institutoptique.fr

Last revision: July 5, 2018


Near-field to far-field transformations
Free-space Radiation
Guided Waves
90°
180°
0°
360°
Free-space Radiation Diagram
90°
180°
0°
360°
Guided-wave Radiation Diagram

[1] Now at Ginzton Lab, Dept. Electrical Engineering, Stanford University, Stanford, California 94305.

RETOP: an open-source for retrieving the free-space and guided-mode radiation diagrams of inhomogeneities embedded in thin-film stacks

The present documentation provides the user guide of **RETOP**, which is a Matlab-based open-source numerical tool, built for implementing near-to-far-field transform (NFFT) for light scattering or emission problems associated to a local inhomogeneity (scatterers and local sources) embedded in a stratified medium (or uniform background) [1]. **RETOP** can be used to retrieve **the free-space and/or guided-mode radiation diagrams** of light scattering/emission problems.

To perform NFFT, **RETOP** requires the knowledge of the near-zone field (obtained from any full-wave Maxwell's solver) of the scattering/emission problem. The near field (in 3D, a vector composed of 6 electric and magnetic components) has to be calculated by the Maxwell software on a *rectangular* "box" that should fully surround the local inhomogeneity.

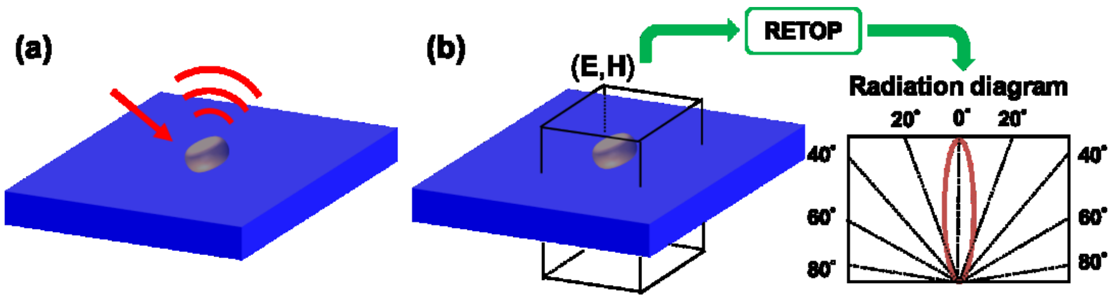


Figure 1. **Illustration of the retrieval procedure of the radiation diagram.** (**a**) Example of a scattering problem: the scatterer on a thin-film stack scatters a driving field (or just emits light). (**b**) Retrieval: the electromagnetic field (**E**, **H**) on a rectangular box that fully surrounds the scatterer, calculated with any full-wave Maxwell solvers, is used by **RETOP** to retrieve the guided or free-space radiation diagrams.

[1] J. Yang, J.P. Hugonin, and P. Lalanne, "Near-to-far field transformations for radiative and guided waves", ACS Photonics, **3**, 395 (2016)

Content

1. Introduction to RETOP

RETOP (named from the Matlab subroutine “**retop.m**”) calculates the radiation diagrams (free-space plane-waves and guided modes diagrams) of local emitters or scatterers embedded in a stratified medium, at each angle. The outputs of **RETOP** are the angular distributions of the phase, amplitude, and intensity of these modes.

In this section, basic information about **RETOP** is provided.

1.1 Download & Installation

RETOP is a freeware that operates under Matlab environment with a set of Matlab subroutines (the users uses directly “retop.m”). To install it, copy and decompress the companion folder “RETOP-Package.zip” and add the folder in the Matlab path.

1.2 User Environment

Using RETOP with a COMSOL environment

In this user guide we provide two examples as documented in Sections 3 and 4, which have been implemented using COMSOL Multiphysics (**version 5.2a**) and correspond to the first two examples in [1]. Each example contains three files:

(i) a COMSOL model sheet where the light scattering/emission problem is defined;

(ii) a main Matlab pedagogical script, which firstly solves the COMSOL model (via Matlab-COMSOL livelink) and then retrieves radiation diagrams with **retop.m** and the electromagnetic field from COMSOL by using the auxiliary MATLAB subroutine “**extract_comsol_field.m**”, which interfaces **retop.m** and the COMSOL model (in order to send the field from COMSOL to **retop.m**).

Using RETOP with Maxwell’s solver other than COMSOL

For Maxwell’s solvers other than COMSOL, which may not offer the Matlab-COMSOL livelink feature, the user should build a user function, similar to **extract_comsol_field.m**, to provide **retop.m** with the field computed on the box by the user solver. It is a trivial operation, see Section 1.9. To help further the user that is not familiar with COMSOL, we provide two other examples,

Exemple_retop_2D_matlab_code_only.m and

Exemple_retop_3D_matlab_code_only.m,

which are fully-documented and rely on Matlab programs only. They are simple (they compute the radiation diagram of a dipole in free space) but are pedagogical.

All functions start by ret and the user should not use variables beginning by ret, to avoid potential conflict.

1.3 How to acknowledge and cite

We kindly ask that you reference the **RETOP** package from IOGS-CNRS and its authors in any publication/report for which you used it. The preferred citation for RETOP is the following paper:

Jianji Yang, Jean-Paul Hugonin and Philippe Lalanne, ACS Photonics **3**, 395–402 (2016)

“Near-to-far field transformations for radiative and guided waves”

A brief description of the algorithm might be:

“The near-to-far-field transformation was computed using a method based on reciprocity arguments using a freely available software package [ref].”

1.4 Provided files

RETOP with a COMSOL environment

File name	**Succinct description**	**Documented-didactical script? (Y/N)**
retop.m	Main function used for the retrieval	N
main_script_dipoles_in_waveguide.m	Matlab script, which first solves the COMSOL model double_dipole_in_waveguide_web.mph (via Matlab-COMSOL livelink) and then retrieves the free-space and guided radiation diagrams.	Y
main_script_sphere.m	Matlab script, which first solves the COMSOL model sphere_in_air_web.mph (via Matlab-COMSOL livelink) and then retrieves the free-space radiation diagram.	Y
extract_comsol_field.m	Matlab function to extract the (scattered or total) field from COMSOL multiphysics	N
double_dipole_in_waveguide_web.mph	COMSOL model for calculating the emission of two dipole source in a waveguide, corresponding to Example 2 in [1].	–
sphere_in_air_web.mph	COMSOL model for calculating the scattering of a dielectric sphere in free space, corresponding to Example 1 in [1].	–

RETOP with Maxwell’s equation solvers other than COMSOL

File name	**Succinct description**	**Documented-didactical script? (Y/N)**
retop.m	Main function used for the retrieval	N
Exemple_retop_2D_matlab_code_only.m	Simple example computing the radiation diagram of a line source (2D) in free space and illustrating how to use **RETOP** with a Maxwell solver other than COMSOL, in this case matlab only.	Y
Exemple_retop_3D_matlab_code_only.m	Simple example computing the radiation diagram of a point source (3D) in free space and illustrating how to use **RETOP** with a Maxwell solver other than COMSOL, in this case matlab only.	Y
extract_pointcoordinate_on_box.m	Very simple matlab function to extract the coordinates of the sampling points on the box. The point coordinates are saved in prv_coordonnees.mat.	N

1.5 Units and conventions of input/output data for RETOP

Unit. All the input information is required to be in the **SI unit** (e.g., volts per meter for electric field **E**, amperes per meter for magnetic field **H**, meter for stratum thickness and spatial coordinates …). Accordingly, the output information is given in **SI unit** as well.

Convention. The time dependent terms exp(*iωt*) or exp(-*iωt*) are allowed, but should be specified when initializing **RETOP**. The users should set "**option_i=1**" for exp(iωt) (e.g., COMSOL), and "**option_i=-1**" for exp(-iωt).

1.6 Coordinates

Three-dimensional (3D) case. For 3D cases, the amplitudes of plane-waves (for free-space radiation diagram) and the amplitudes of guided modes (for guided-mode radiation diagram) are respectively defined in spherical coordinates (r,θ,φ) with $\theta \in [0,\pi/2], \varphi \in [0,2\pi]$ (see Fig. 2a) and cylindrical coordinates (r,φ) with $\varphi \in [0,2\pi]$ (see Fig. 2b). Note that **the *z*-axis should be perpendicular to the thin-film layers**.

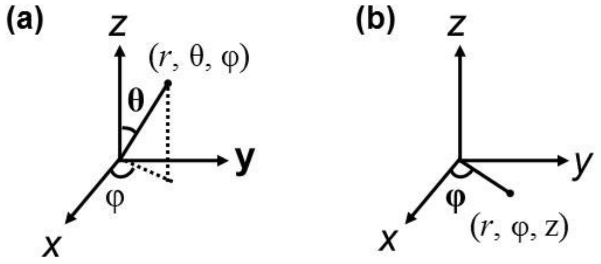


Figure 2. **Coordinates for 3D cases**. (**a**) Spherical coordinate is adopted for describing free-space radiation diagram. (**b**) Cylindrical coordinate is adopted for describing guided-mode radiation diagram. The space is divided into upper (z > +∞) and lower (lossless, e.g., air or glass) space (z < –∞) for the free-space radiation diagram (retrieved independently for upper and lower spaces).

Two-dimensional (2D) case. For 2D cases, the angular amplitude of plane-waves (for free-space radiation diagram) are defined in a polar coordinate (r,θ) shown in Fig. 3, with $\theta \in [-\pi/2, \pi/2]$. Note that, there are only two directions (i.e., +x and -x) that are needed for describing the "*angular*" distribution of guided mode amplitudes. Note that **the *y*-axis should be perpendicular to the thin-film layers**.

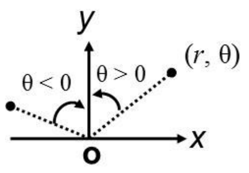


Figure 3. **Coordinates for 2D cases**. Polar coordinate is adopted for describing the free-space radiation diagram in 2D cases. The space is divided into upper (y > +∞) and lower space (y < –∞) for the free-space radiation diagram (retrieved independently for upper and lower space).

1.7 Polarization of free-space radiation diagrams

Figure 4a refers to 3D, and Fig. 4b to 2D. For a propagation direction defined by the unitary vector **k** specified by the angles (θ,φ), we define two unitary vectors **u** (in the plane (**k**, **z**)) and **v** (perpendicular to the plane (**k**, **z**)). **RETOP** provides the radiation diagrams for plane-waves polarized along both **u** and **v**.

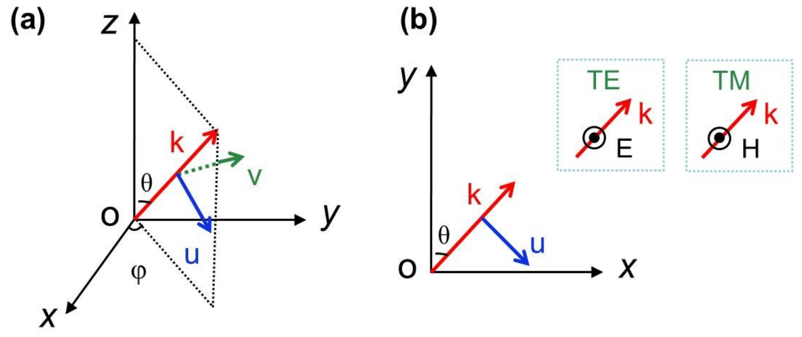


Figure 4. **Illustration of plane-wave polarizations for 3D and 2D free-space radiation diagrams**. (**a**) 3D case. The vector **k** labels the propagation direction of two collinear (but with orthogonal polarization) plane-waves, polarized along **u** or **v**. Vectors **u**, **v**, and **k** are unitary, orthogonal to each other and the pair (**k**,**u**,**v**) is direct. Note that the components of these vectors are given in the Retrieval output of **retop.m**. (**b**) 2D case. The vector **k** labels the propagation direction of a TE/TM plane-wave, and the vector **u** shows the direction of the **H**/**E** component. The polarization is defined by the control parameter '**polarization**' (polarization = 0/2 for TE/TM).

1.8 Outline of the theory and related key issues

As explained in [1], **RETOP** basically computes a surface integral (see Eqs. (2) and (11) in [1]) between the **near-zone field** (see **Point A** below) provided by any full-wave Maxwell's equation solver (not included in the present software) and the field induced by a backward-propagating **mode** (**Point B**) over a rectangular "**box**" (**Point C**) that fully surrounds the inhomogeneity (**Point D**).

● **Point A**: about the "**near-zone field**"

What is the sort of field that should be computed with the full-wave Maxwell's equation solver? It depends what the user wants to do, but in all cases, the field should correspond to outgoing waves (**RETOP** might handle incoming waves for which the radiation is theoretically diagram null; however it is recommended to substract incoming fields if possible). We give two examples. 1/ a scatterer is illuminated by a plane wave or a far-away source. One wishes to calculate the radiation diagram of the scatterer. The input field in **RETOP** is the difference between the total field and the incident field calculated without the scatterer in the stratified medium. The difference should be calculated on a box surrounding the scatterer. 2/ a scatterer is illuminated by a nearby local source. One wishes to calculate the radiation diagram of the source+scatterer. The input field in **RETOP** is simply the total field computed on a box surrounding the scatterer and the source.

● **Point B**: about the "**mode**"

For retrieval of free-space radiation diagram, the mode refers to radiative modes (free-space modes), *i.e.*, plane waves; however, for retrieval of guided mode radiation diagram, the mode refers to guided modes of the stratified structure.

● **Point C**: about the "**box**"

The "box" should completely surround the inhomogeneity [1].

● **Point D**: about the "**inhomogeneity**"

The inhomogeneity can be composed of a single scatterer, a collection of scatterers, a collection of scatterers and one or several local emitting sources.

1.9 An interface subroutine for interfacing retop.m with Maxwell's equation solvers

The main Matlab function of the **RETOP** package is **retop.m**. The latter relies on a few input parameters and returns the radiation diagrams. Details on input/output parameters are given in Tutorial Sections 2 and 3.

A key input for **retop.m** is the near-zone field (**E**,**H**) calculated on a closed "box" by a full-wave Maxwell's equation solver, and an uncomplicated auxiliary Matlab function, that may be named '**extract_field.m'**, which allows **retop.m** to read the near-zone field on points located on the "box" contour. Indeed the auxiliary Matlab subroutine depends on the Maxwell's equation solver at hand.

RETOP use with a COMSOL environment

For users that use COMSOL, like us, we provide a subroutine named as **extract_comsol_field.m** that has been developed for use with COMSOL Multiphysics and that has been documented in a pedagogical way. The key command in the subroutine is the COSMOL scripting command '**mphinterp**' used for sampling field at desired locations.

RETOP use with Maxwell's equation solvers other than COMSOL

For users that are not using COMSOL, we provide two pedagogical Matlab examples

- **Exemple_retop_2D_matlab_code_only.m**,
- **Exemple_retop_3D_matlab_code_only.m**,

which are documented scripts that may be helpful for the users to build their own function to extract the field computed with their own solver on the box contour. The examples can be tested autonomously in a Matlab environment for the sake of simplicity; they use the trivial Matlab function **extract_pointcoordinate_on_box.m** to compute the plane-wave decomposition of the emission of a dipole in free space, either in 2D (the dipole is a line source) or in 3D (the dipole is a Dirac source).

2. TUTORIAL: FREE-SPACE RADIATION DIAGRAM

The free-space radiation diagram describes the angular distribution of energy flux density of the field radiated by the inhomogeneity into the two upper and lower half-spaces. Hereafter we give a tutorial on how to implement retrievals of 3D and 2D radiation diagrams.

2.1 Free-space radiation diagram: 3D case

There are two key steps for retrieving 3D free-space radiation diagram: **initialization** and **retrieval**. Remember that the **z-axis is perpendicular to the stack layers**.

Step 1: < **Initialization** >

The following Matlab line initializes **RETOP** by setting the basic retrieval parameters; the output of the initialization step is '**init**', a cell array which is used for the retrieval step.

----

init=retop(extract_field, wavenumber, refractive_indices, z_layers, box_center, box_size, N_points, struct('option_i',option_i));

----

extract_field	: user-defined interface that links **RETOP** and the Maxwells' equation solver. For COMSOL users, we provide an interface subroutine **extract_comsol_field.m**. For users who do not use COMSOL, we provide a Matlab script **Exemple_retop_3D_matlab_code_only.m** that pedagogically show how the users may build their own extract_field function.
wavenumber	: vacuum wavenumber $k_0 = 2\pi/\lambda_0$.
refractive_indices[a]	: vector $[n_1,..n_m]$, refractive indices of each layer (from top to bottom).
z_layers[b]	: vector $[z_1,..z_{m-1}]$, z-positions of the layer interfaces.
box_center	: vector [x,y,z], coordinate of the center of the "box".
box_size	: vector $[L_x,L_y,L_z]$, box sizes in x, y and z directions.
N_points[c]	: real numbers that specifies the grid-finesse degree for discretizing the "box". We might recommend to set N_points=[60;60;60] (in 3D), if the box size is smaller than one wavelength. Otherwise, increase N_points proportionally.
option_i	: number that specifies the time-dependent term of the full-wave Maxwell's solver. The users should set "**option_i=1**" for exp(iωt) (e.g., COMSOL), and "**option_i=-1**" for exp(-iωt).

[**a**] and [**b**]: Refer to Fig. 5.

[**c**]: According to Ref. [1], **RETOP** calculates a surface integral on the closed 'box' to retrieve the modal amplitude. A Gauss-Legendre integration method is adopted to compute the surface integral, and the "box" is discretized according to this method. In 3D case, the box is a cuboid and 'N_points' has to be defined as "N_points=[N_x; N_y; N_z]" ($N_{x/y/z}$ are all positive integers), to discretize the box along the x/y/z-direction into $N_{x/y/z}$ points. Larger box sizes require larger values of $N_{x/y/z}$; empirically, we recommend to use N_x ~$10k_0L_x$). Similarly, in 2D case, N_points has to be defined as "N_points=[N_x; N_y]".

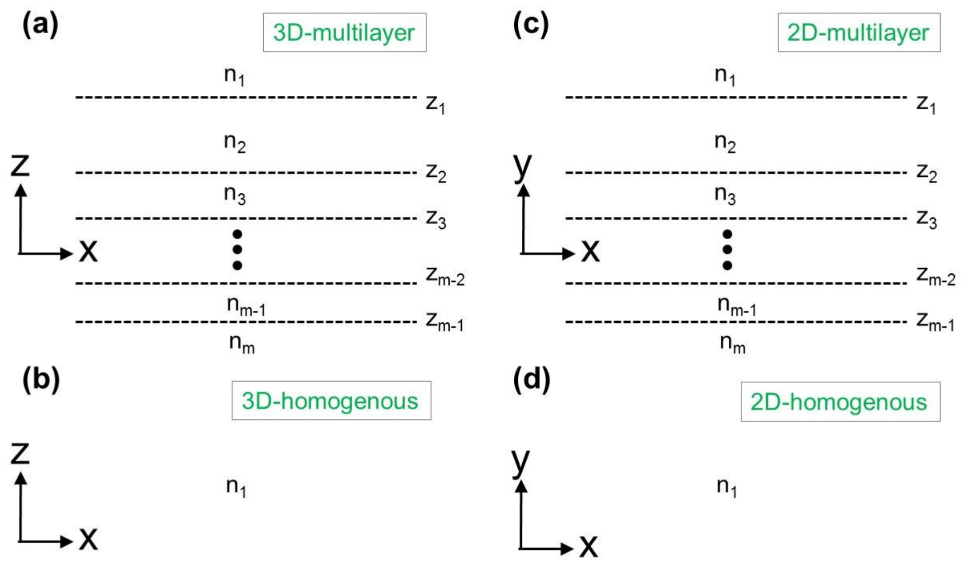


Figure 5. **Refractive indices and interface.**

(a) 3D stratified host medium: set refractive_indices=[n_1,..n_m] and z_layers=[z_1,…z_{m-1}].

(b) 3D uniform host medium: set refractive_indices=[n_1] and z_layers=[] (the (x,y,z) origin is the center of the box by default).

(c) 2D stratified host medium: set refractive_indices=[n_1,..n_m] and z_layers=[y_1,…y_{m-1}].

(d) 2D uniform host medium: set refractive_indices=[n_1] and z_layers=[] (the (x,y,z) origin is the center of the box by default).

Step 2: < **Retrieval** >

The following line performs retrieval, the output of this step is a Matlab structure '**angles**' described after.

----

angles=retop(init, u, v, direction, struct('test',1));

----

init	: the output of the initialization step.
u,v	: vectors containing the information on the polar and azimuthal angles defined by user.
direction	: number. 'direction=1/-1' retrieval for the upper/lower space.
struct('test',1)	: in 3D, if test equals 1, a figure showing the |**E**|^2 on the box is plotted.

See the following *example* on how to define u, v and direction (assuming *init* is known)

--- *example* (begin) ---

```
teta= linspace(0,pi/2,30);            % polar angle.
[phi,wphi]= linspace(0,2*pi,54);      % azimuthal angle.
% NOTE: the whole upper & lower spaces are discretized into (10x3)x(9x6)=1620 elementary directions.
[Teta,Phi]=ndgrid(teta,phi);          % generate the angular grid
u=sin(Teta).*cos(Phi);                % weight in x-coordinate
```

```
v=sin(Teta).*sin(Phi);                 % weight in y-coordinate
w=cos(Teta);                           % weight in z-coordinate
kx_up=k0*n1*u; ky_up=k0 *n1*v;         % kx, ky in the upper space with refractive index n1 (k0: wavenumber)
kx_dn=k0*nm*u; ky_dn =k0 *nm*v;        % kx, ky in the lower space with refractive index nm (k0: wavenumber)
direction_up =1;                       % for retrieving plane-waves propagating toward +z
direction_dn=-1;                       % for retrieving plane-waves propagating toward -z
angles_up=retop(init, kx_up, ky_up,direction_up); %  planewave decomposition for the upper space (+z)
angles_dn=retop(init, kx_dn, ky_dn,direction_dn); %  planewave decomposition for the lower space (-z)
```

-- *example* (end) ---

The Retrieval Output

The output of the **retop.m** function is a Matlab structure *angles* (e.g., '*angles_up*' or '*angles_dn*' in the example above), which contains a few important parameters illustrated for '*angle_up*':

1)	angles_up.theta:	[1620*x1 matrix]
2)	angles_up.delta:	[1620x1 matrix]
3)	angles_up.F:	[1620x1 matrix]
4)	angles_up.k:	[1620x3 matrix]
5)	angles_up.v:	[1620x3 matrix]
6)	angles_up.u:	[1620x3 matrix]
7)	angles_up.EH:	[1620x6 matrix]
8)	angles_up.EE:	[1620x2 matrix]
9)	angles_up.HH:	[1620x2 matrix]
10)	angles_up.origine:	[1x1 matrix]

* see the above example about the number '1620'

1)-2): 'theta' and 'delta' represent the polar and azimuthal angle of each direction **k**;

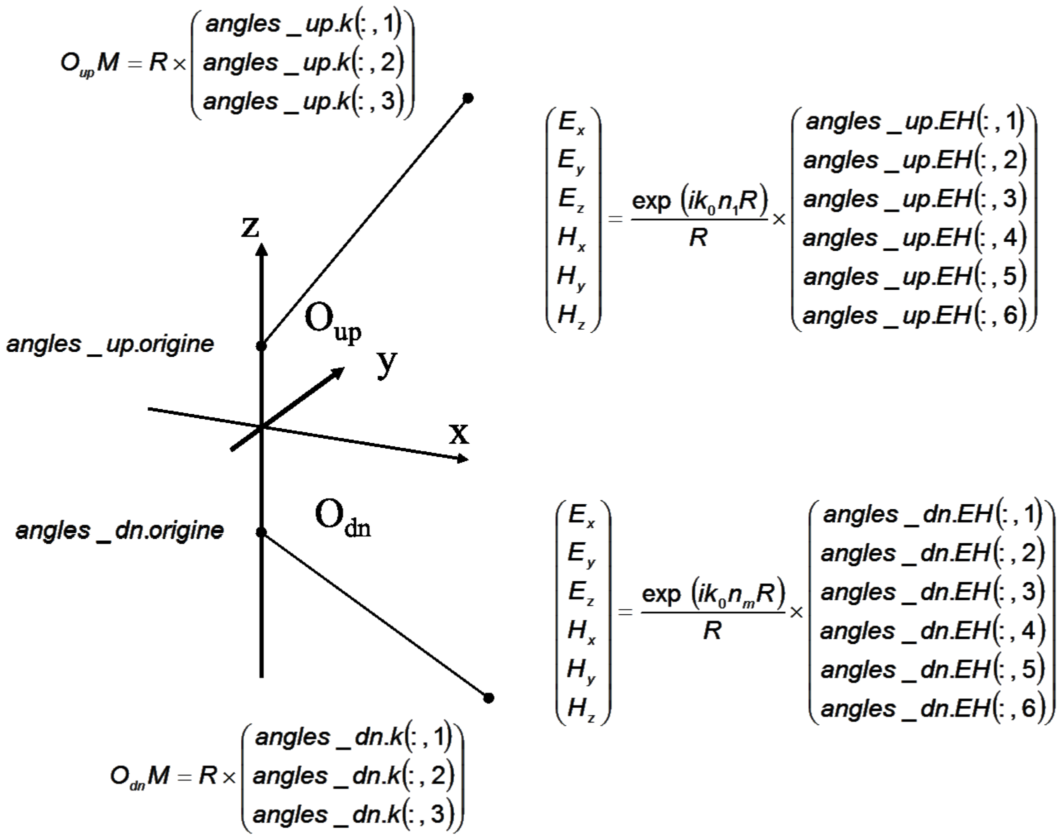
3): 'F' the total Poynting flux at each direction ('total' means for **u** and **v** plane waves, see Fig. 4);

4)-6): The unitary vectors **k**, **u**, and **v**, expressed by their projections in the Cartesian coordinates, i.e., (k_x,k_y,k_z), (u_x,u_y,u_z) and (v_x,v_y,v_z), define the direction and polarization of the planewaves.

7): angles_up.EH is a very important output that allows to know the complex amplitude of the far-field asymptotic field. Namely, angles_up.EH(:,1), angles_up.EH(:,2), angles_up.EH(:,3), angles_up.EH(:,4), angles_up.EH(:,5), and angles_up.EH(:,6) respectively represent the E_x, E_y, E_z, H_x, H_y, and H_z components of the asymptotic ($|z|\to\infty$) field. The latter is by definition of angles_up.EH: angles_up.EH × *exp*(ik_0R)/R, with $R = \|\mathbf{O_{up}M}\|$ in the upper half-space and $R = \|\mathbf{O_{dn}M}\|$ in the lower half-space (see Fig. 6).

8)-9): [angles_up.EE(:,1), angles_up.HH(:,2)] are vectors formed by the electric- and magnetic-field components of the plane-waves propagating along **k** and polarized along **u**; [angles_up.EE(:,2), angles_up.HH(:,1)] represents the electric and magnetic field components of the plane-waves propagating along **k** and polarized along **v**. Note that, angles_up.EE(:,1) and angles_up.HH(:,1) are vectors parallel to of **u**; similarly, angles_up.EE(:,2) and angles_up.HH(:,2) are vectors parallel to **v**.

10): 'origine' is a real number that indicates the z-coordinate of zero-phase of all the plane-waves. The phase origin is critical for further use, for instance for reconstructing the total field resulting from the interference between the driving and scattered fields. The x- and y-positions of zero-phase is set as x=0 and y=0 (a natural selection). Thus a plane wave scattered toward the upper half-space is written *exp*(i × *n* × **k** · [**r**–**r**$_0$]), with *n* denoting the refractive index of the upper space and **r**$_0$ = [0,0, angles_up.origine] in the convention *exp*(-iωt).



$O_{up}M = R \times \begin{pmatrix} angles\_up.k(:,1) \\ angles\_up.k(:,2) \\ angles\_up.k(:,3) \end{pmatrix}$
$\begin{pmatrix} E_x \\ E_y \\ E_z \\ H_x \\ H_y \\ H_z \end{pmatrix} = \frac{\exp(ik_0 n_1 R)}{R} \times \begin{pmatrix} angles\_up.EH(:,1) \\ angles\_up.EH(:,2) \\ angles\_up.EH(:,3) \\ angles\_up.EH(:,4) \\ angles\_up.EH(:,5) \\ angles\_up.EH(:,6) \end{pmatrix}$
z
O_{up}
angles_up.origine
y
x
angles_dn.origine
O_{dn}
$\begin{pmatrix} E_x \\ E_y \\ E_z \\ H_x \\ H_y \\ H_z \end{pmatrix} = \frac{\exp(ik_0 n_m R)}{R} \times \begin{pmatrix} angles\_dn.EH(:,1) \\ angles\_dn.EH(:,2) \\ angles\_dn.EH(:,3) \\ angles\_dn.EH(:,4) \\ angles\_dn.EH(:,5) \\ angles\_dn.EH(:,6) \end{pmatrix}$
$O_{dn}M = R \times \begin{pmatrix} angles\_dn.k(:,1) \\ angles\_dn.k(:,2) \\ angles\_dn.k(:,3) \end{pmatrix}$

Figure 6. Definition of some important output parameters of **retop.m** in 3D that allows us to define the complex amplitude (phase & modulus) of the far-field scattered wave. All elements of the Matlab structure *angles* are in the in the convention *exp*(-iωt).

2.2. Free-space radiation diagram: 2D case

The 2D case is similar to the 3D case, except for a new input parameter *polarization* that controls the polarization, TE (**E** parallel to the stack) or TM (**H** parallel to the stack), at the initialization step. Note that the normal to the stack is the y-direction in 2D (and not z as in 3D).

Step 1: < **Initialization** >

The following Matlab line initializes the **RETOP** by setting the basic retrieval parameters; the output of the initialization step is '**init**', a cell array which is used for the retrieval step.

----

init=retop(extract_field, polarization, wavenumber, refractive_indices, y_layers, box_center, box_size, N_points, struct('option_i',option_i));

---

extract_field	: user-defined interface that links **RETOP** and the Maxwells' equation solver. For COMSOL users, we provide an interface subroutine **extract_comsol_field.m**. For users who do not use COMSOL, we provide a Matlab script **Exemple_retop_2D_matlab_code_only.m** that pedagogically show how the users may build their own extract_field function.
polarization	: integer, polarization=0/2 TE/TM (**see Fig. 4**).
wavenumber	: vacuum wavenumber $k_0 = 2\pi/\lambda_0$.
refractive_indices[a]	: vector [n_1,..n_m], refractive indices of each layer (from top to bottom).
y_layers[b]	: vector [y_1,..y_{m-1}], y-positions of the layer interfaces.
box_center	: vector [x,y], coordinate of the center of the "box".
box_size	: vector [L_x,L_y], box sizes in x and y.
N_points[c]	: real numbers that specifies the grid-finesse degree for discretizing the "box". We might recommend to set N_points=[60;60] (in 2D), if the box size is smaller than one wavelength. Otherwise, increase N_points proportionally.
option_i	: number that specifies the time-dependent term of the full-wave Maxwell's solver. The users should set "**option_i=1**" for exp(iωt) (e.g., COMSOL), and "**option_i=-1**" for exp(-iωt).

[**a**] [**b**]: For the host stratified media made of multiple layers with different refractive indices, one has to specify the refractive index of every layer, by setting the parameter 'refractive_indices' as [n_1,..n_m] (from the top to the bottom, see **Fig. 5c**). Also, the y position of each interface between adjacent layers should be specified by setting the parameter 'y_layers' as [y_1,…y_{m-1}]. Note that if the host media is *homogeneous* (see **Fig. 5d**), one has to define the parameters 'refractive_indices' as [n_1] (with n_1 denoting the refractive index of the host media) and 'y_layers' as [] (with y_1).
[**c**]: see the notes on "N_points" in the 3D case.

Step 2: < **Retrieval** >
The following line performs retrieval, the output of this step is a Matlab structure '**angles**', which is explained in details later.

---

angles=retop(init,u,direction,struct('test',1));

---

init	: the output of the initialization step;
u	: vector containing the information of the polar angle defined by user;
direction	: 'direction=1/-1' retrieval for the upper/lower space (y=∞ or y=-∞).
struct('test',1)	: in 2D, if test equals 1, a figure showing the modulus squared of each field-components on the contour of the rectangular box is plotted.

See the following *example* on how to define 'u' and 'direction'

```
---------------------------------------------------- example (begin) ----------------------------------------------------------
k0=2*pi/1.5;                          % wavenumber
[teta]= linspace(-pi/2,pi/2,200);     % polar angle. One may use [teta,wteta]=retgauss(-pi/2,pi/2,10,20);
% NOTE: the whole upper/lower space is discretized into 200 elementary directions.
u=sin(teta);                          % weight in x-coordinate
v=cos(teta);                          % weight in y-coordinate
kx_up=k0*n1*u;                        % kx in the upper space with refractive index n1 (k0: wavenumber)
kx_dn=k0*nm*u;                        % kx in the lower space with refractive index nm (k0: wavenumber)
direction_up =1;                      % for retrieving plane-waves propagating toward +y
direction_dn=-1;                      % for retrieving plane-waves propagating toward -y
angles_up=retop(init, kx_up, direction_up); %  planewave decomposition for the upper half-space (+y)
angles_dn=retop(init, kx_dn, direction_dn); %  planewave decomposition for the lower half-space (-y)
---------------------------------------------------- example (end) ------------------------------------------------------------
```

The Retrieval Output

As seen in < **Retrieval** > section, the output of **retop.m** in the retrieval step is a Matlab structure *angles* (e.g., '*angles_up* or '*angles_dn*' in the example code above), which contains a few important parameters as listed below (we take '*angle_up'* in the next *example* but it is the same for '*angle_dn'*):

1)	angles_up.theta	[200x1 matrix] *
2)	angles_up.F	[200x1 matrix]
3)	angles_up.k	[200x2 matrix]
4)	angles_up.u	[200x2 matrix]
5)	angles_up.EH	[200x3 matrix]
6)	angles_up.EE	[200x1 matrix]
7)	angles_up.HH	[200x1 matrix]
8)	angles_up.origine	[1x1 matrix]

*: see the above example about the number '200'

and explained as following:

1): 'theta' represents the polar angle of each elementary solid angle;

2): 'F' the Pointing flux at each direction;

3)-4): The vectors **k** and **u** are defined for setting-up the plane-waves at in each elementary solid angle (see **Remarks on the polarization of plane-waves at every direction** in the beginning of **Section 1.2**), expressed by their projections in the Cartesian coordinates, i.e., (k_x,k_y) and (u_x,u_y).

5): angles_up.EH is a very important output that allows to know the complex amplitude of the far-field asymptotic field. Namely, angles_up.EH(:,1), angles_up.EH(:,2), angles_up.EH(:,3), respectively represent the H_z, E_x, E_y, (TM) or E_z, H_x, H_y, (TE) components of the asymptotic ($|z|\rightarrow\infty$) field. The latter is by definition of angles_up.EH: angles_up.EH × $exp(ik_0R)/R$, with $R = ||\mathbf{O_{up}M}||$ in the upper half-space and $R = ||\mathbf{O_{dn}M}||$ in the lower half-space (see Fig. 7).

6)-7): [angles_up.EE, angles_up.HH] represents the amplitudes of the electric and magnetic field of the plane-wave (with the polarization defined by user).

8): 'origine' indicates the y-position of zero-phase for all the plane-waves. The phase origin is critical for further applications, for reconstructing the total field with background field and scattered field. For the x-position of zero-phase, it is set as x=0 (natural selection).

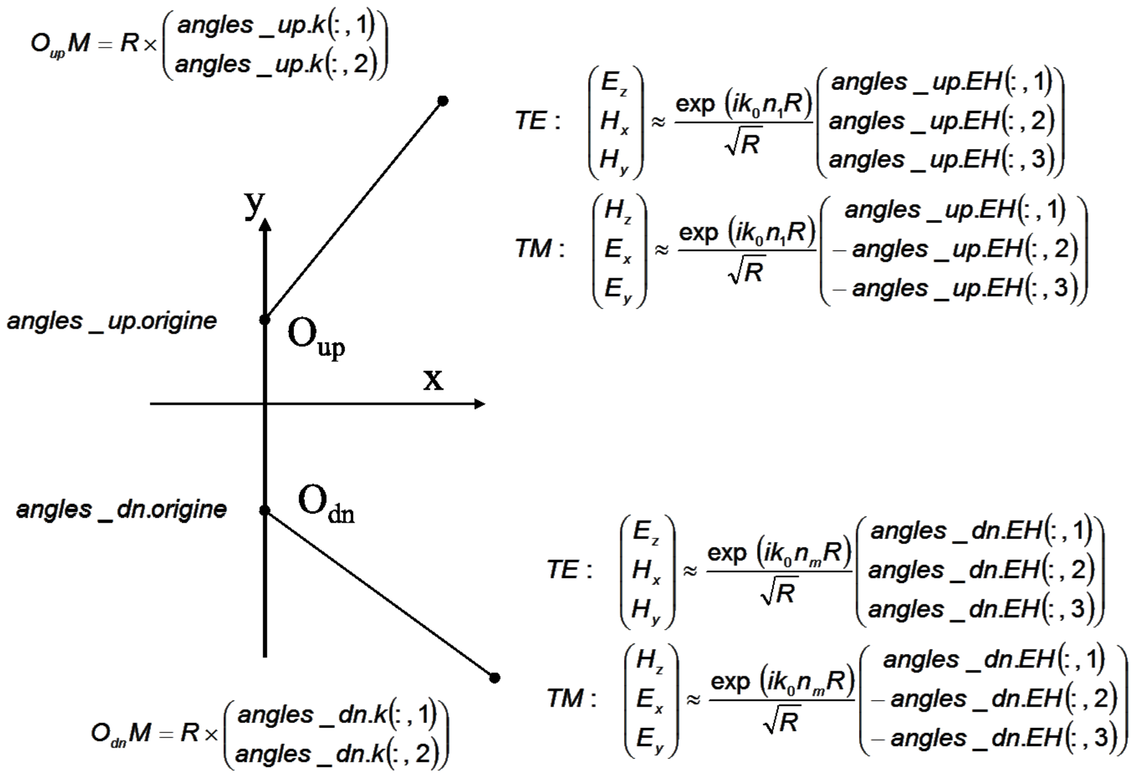

$O_{up}M = R \times \begin{pmatrix} angles\_up.k(:,1) \\ angles\_up.k(:,2) \end{pmatrix}$
$TE: \begin{pmatrix} E_z \\ H_x \\ H_y \end{pmatrix} \approx \frac{\exp(ik_0 n_1 R)}{\sqrt{R}} \begin{pmatrix} angles\_up.EH(:,1) \\ angles\_up.EH(:,2) \\ angles\_up.EH(:,3) \end{pmatrix}$
$TM: \begin{pmatrix} H_z \\ E_x \\ E_y \end{pmatrix} \approx \frac{\exp(ik_0 n_1 R)}{\sqrt{R}} \begin{pmatrix} angles\_up.EH(:,1) \\ -angles\_up.EH(:,2) \\ -angles\_up.EH(:,3) \end{pmatrix}$
y
angles_up.origine
O_{up}
x
angles_dn.origine
O_{dn}
$TE: \begin{pmatrix} E_z \\ H_x \\ H_y \end{pmatrix} \approx \frac{\exp(ik_0 n_m R)}{\sqrt{R}} \begin{pmatrix} angles\_dn.EH(:,1) \\ angles\_dn.EH(:,2) \\ angles\_dn.EH(:,3) \end{pmatrix}$
$TM: \begin{pmatrix} H_z \\ E_x \\ E_y \end{pmatrix} \approx \frac{\exp(ik_0 n_m R)}{\sqrt{R}} \begin{pmatrix} angles\_dn.EH(:,1) \\ -angles\_dn.EH(:,2) \\ -angles\_dn.EH(:,3) \end{pmatrix}$
$O_{dn}M = R \times \begin{pmatrix} angles\_dn.k(:,1) \\ angles\_dn.k(:,2) \end{pmatrix}$

Figure 7. Definition of some important output parameters of **retop.m** in 2D that allows us to define the complex amplitude (phase & modulus) of the far-field scattered wave. All elements of the Matlab structure *angles* are in the in the convention *exp*(-iωt).

3. TUTORIAL: GUIDED-MODE RADIATION DIAGRAM

The guided-mode radiation diagram describes the angular distribution of the guided-mode intensities in the stratified medium.

If the stack supports *N* guided modes, the retrieval should be performed *N* times, each time by specifying the polarization TE/TM and an approximate value of the mode effective index.

3.1. Guided-mode radiation diagram : 3D case

There are two key steps for retrieving guided-mode radiation diagram in 3D cases: **initialization** and **retrieval**. Note that if both free-space and guided-mode radiation diagrams are retrieved in the same code, **initialization** only needs to be made once.

Step 1: < **Initialization** >

The following Matlab line initializes **RETOP** by setting the basic retrieval parameters; the output of the initialization step is '**init**', which will be used for the retrieval step.

---

init=retop(extract_field, wavenumber, refractive_indices, z_layers, box_center, box_size, N_points, struct('option_i',option_i));

---

extract_field	: user-defined interface that links **RETOP** and the Maxwells' equation solver. For COMSOL users, we provide an interface subroutine **extract_comsol_field.m**. For users who do not use COMSOL, we provide a Matlab script **Exemple_retop_3D_matlab_code_only.m** that pedagogically show how the users may build their own extract_field function.
wavenumber	: vacuum wavenumber $k_0 = 2\pi/\lambda_0$.
refractive_indices[a]	: vector [n_1,..n_m], refractive indices of each layer (from top to bottom).
z_layers[b]	: vector [z_1,..z_{m-1}], z-positions of the layer interfaces.
box_center	: vector [x,y,z], coordinate of the center of the "box".
box_size	: vector [L_x,L_y,L_z], box sizes in x, y and z.
N_points[c]	: real numbers that specifies the grid-finesse degree for discretizing the "box". We might recommend to set N_points=[64;64;64] in 3D.
option_i	: number that specifies the time-dependent term of the full-wave Maxwell's solver. The users should set "**option_i=1**" for exp(iωt) (e.g., COMSOL), and "**option_i=-1**" for exp(-iωt).

[**a**], [**b**], and [**c**]: see the related notes in Section 2.1.

Step 2: < **Retrieval** >

The following line performs retrieval, the output of this step is a Matlab structure '**angles**'.

---

angles_mode=retop(init, neff_mode, polarization_mode, azimuthal_orders, phi);

---

init	: output of the initialization step.
neff_mode	: real number, mode effective index (complex-valued for lossy stacks, see [1]).
polarization_mode	: integer, "polarization_mode=0/2" corresponds to TE/TM guided modes.
azimuthal_orders	: vector defining the values of the azimuthal harmonic orders, *exp*(i*n*φ) with *n* (*n* = 0, ±1…), for the guided-wave expansion, see [1]. We recommend "azimuthal_orders=[-N:N]" with N ~ 5-200, depending on the

problem at hand. Large N values result in more resolved φ–expansions and longer computational times.

phi : vector containing the azimuthal angles φ.

See the following *example* on how to define neff_mode, polarization_mode, azimuthal_orders, and phi (assuming *init* is known)

-- *example* (begin) --

```
neff_TM=1.1+0.1i;            % approximate value of mode effective index (might be a complex number)
polarization_mode=2;         % TM mode, such as a surface plasmon polariton on dielectric/metal interface
azimuthal_orders =[-5:5];     % set the interval [-5:5] of azimuthal harmonic orders.
[phi_mode,wphi_mode]= linspace(0,2*pi,200); % define the sample points of the azimuthal angles to
compute the guided-mode radiation diagram.
angles =retop(init,neff_TM, polarization_mode, azimuthal_orders,phi_mode); % compute mode expansion
```

-- *example* (end) --

The Retrieval Output

As seen in < **Retrieval** > Section, the output of **retop.m** in the retrieval step is a Matlab structure *angles* (e.g., '*angles'* in the example code above), which contains a few important parameters:

1)	angles.f	[1x11 matrix]
2)	angles.amp	[1x400 matrix]
3)	angles.F	[1x400 matrix]
4)	angles.L	[-5 -4 -3 -2 -1 0 1 2 3 4 5]
5)	angles.neff	1.1013 + 0.0980i

1): 'f' is the complex-valued amplitude of the cylindrical guided mode (see [1]), it corresponds to the 'azimuthal_orders'.

2): 'amp' is the total complex-valued amplitude (summed over the azimuthal orders) scattered in each azimuthal direction phi_mode (F=abs(amp).^2). It corresponds to the $f_m(\varphi)$ in Eq. (4) in [1].

3): 'F' the Poynting flux (as a function of 'phi_mode') of the guided mode; the total energy carried by the guided modes is 2*pi*sum(abs(angles.f).^2)

4): 'L' represents the azimuthal harmonic orders considered in the retrieval;

5): 'neff' represents the effective index computed by the embedded mode solver (high precision calculation) that is the closest to the initial guess value 'neff_TM'.

3.2. Guided-mode radiation diagram: 2D case

The 2D case is similar to the 3D case, except for a new input parameter *polarization* that controls the polarization, TE or TM, at the initialization step. **Note that the normal to the stack is the y-direction in 2D (and not z as in 3D).**

Step 1: < **Initialization** >

The following Matlab line initializes the **RETOP** by setting the basic retrieval parameters; the output of the initialization step is '**init**', which will be used for the retrieval step.

```
init=retop(extract_field, polarization, wavenumber, refractive_indices, y_layers, box_center, box
_size,N_points,struct('option_i',option_i));
```

---

extract_field	: user-defined interface that links **RETOP** and the Maxwells' equation solver. For COMSOL users, we provide an interface subroutine **extract_comsol_field.m**. For users who do not use COMSOL, we provide a Matlab script **Exemple_retop_2D_matlab_code_only.m** that pedagogically show how the users may build their own extract_field function.
polarization	: integer, polarization=0/2 TE/TM (see Fig. 4).
wavenumber	: real number, vacuum wavenumber $k_0 = 2\pi/\lambda_0$.
refractive_indices[a]	: vector [n_1,..n_m], refractive indices of each layer (from top to bottom).
y_layers[b]	: vector [y_1,..y_{m-1}], y-positions of the layer interfaces.
box_center	: vector [x,y], coordinate of the center of the "box".
box_size	: vector [L_x,L_y], box sizes in x and y.
N_points[c]	: real number that specifies the grid-finesse degree for discretizing the "box".
option_i	: number that specifies the time-dependent term of the full-wave Maxwell's solver. The users should set "**option_i=1**" for exp(iωt) (e.g., COMSOL), and "**option_i=-1**" for exp(-iωt).

[**a**], [**b**], and [**c**]: see the related notes in Section 2.2.

Step 2: < **Retrieval** >

The following line performs retrieval; the output is a Matlab structure '**angles**'.

```
angles_mode=retop(init, neff_mode);
```

---

init	: the output of the initialization step;
neff_mode	: real number, mode effective index (complex-valued for lossy stacks, see [1]).

See the following *example* on how to define neff_mode (assuming *init* is known)

-- *example* (begin) --

```
neff_TM=1.1+0.1i;              % approximate value of mode effective index (might be a complex number)
angles_mode=retop(init,neff_TM);  % calculation of the mode expansion
```

-- *example* (end) --

The Retrieval Output

As seen in < **Retrieval** > Section, the output of the *retop* in the retrieval step is a Matlab structure *angles* (e.g., '*angles_mode*' in the example above), which contains a few important parameters:

1)	angles_mode.amp_p	0.0017 - 0.1368i
2)	angles_mode.Fp	0.0187
3)	angles_mode.amp_m	0.0017 - 0.1368i
4)	angles_mode.Fm	0.0187
5)	angles_mode.neff	1.1006 - 0.0981i

1): 'amp_p' represents the complex amplitude of the guided mode propagating to +x;

2): 'Fp' represents the energy flux carried by the guided mode propagating to +x;

3): 'amp_m' represents the complex amplitude of the guided mode propagating to -x;

4): 'Fm' represents the energy flux carried by the guided mode propagating to -x;

5): 'neff' represents the effective index computed by the embedded mode solver (high precision calculation) that is the closest to the initial guess value 'neff_TM'.

4. Example 1: Plane-wave scattering by a silicon nanosphere in a uniform medium

In this example, the COMSOL model sheet ***sphere_in_air_web.mph***, the Matlab main script ***main_script_sphere.m***, and the Matlab subroutine ***extract_comsol_field.m*** for transferring electromagnetic field of COMSOL to **RETOP**, are minutely explained in **subsections 4.1**, **4.2** and **4.3**, respectively. The example is sketched in Fig. 8, and more details can be found in Example-1 in Ref. [1].

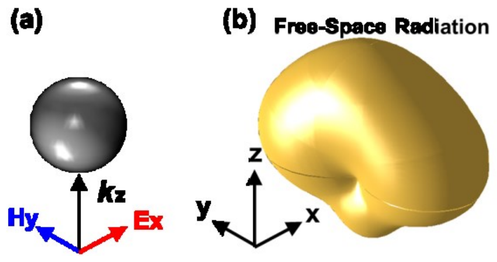


Figure 8. Free-space radiation diagram of a silicon nanosphere illuminated by an *x*-polarized planewave. (**a**) Sketch of the silicon nanosphere and the incident field. (**b**) Radiation diagram of the scattered field retrieved via RETOP.

4.1. Instructions on the COMSOL Model Sheet

The model sheet for the current example is ***sphere_in_air_web.mph***. To simulate light scattering at a nanoparticle in air, it is very convenient to use the scattered-field formulation in COMSOL. For this purpose, select "Scattered Field" in the "Settings" of the "Electromagnetic Waves, Frequency Domain Interface" and define the background electric field as the incident field (*x*-polarized plane wave here).

Definitions of the parameters of the model

In the model sheet ***sphere_in_air_web.mph***, the parameters are defined as below.

(1) **c**: the speed of light in vacuum

(2) **lambda** & **frequency**: operation wavelength & frequency

(3) **radius_sph**: radius of the silicon sphere

(4) **t_pml**: PML thickness (~1/3 wavelength)

(5) **radius_pml**: outer radius of the spherical PML

(6) **eps_sph** & **eps_air**: relative permittivity of the silicon sphere and air

(7) **box**: size of the cuboid, on which the scattered field is sampled and will be sent to **RETOP** for retrieval

(8) $\mathbf{E_0}$: the amplitude of electric field of the incident plane wave

The geometrie of the model

(1) **Sphere 1**: to define the silicon sphere

(2) **Sphere 2**: to define the host medium and the PML (see the option ‘layers’)

(3) **Block 1**: to define the box for sampling field

Definitions of the model materials

The model considers a silicon sphere embedded in a dielectric host medium, so there are two materials.

(1) **Material 1**: the relative permittivity and permeability of the air (host medium)

(2) **Material 2**: the relative permittivity and permeability of the sphere (scatterer)

The model mesh

Three different mesh sizes are used in different space regions and at the sampling ‘box’.

(1) **Size 1** is related to the mesh of the region of the nanosphere

(2) **Size 2** is related to the mesh of the regions of the host medium and the PML

(3) **Size 3** is related to the mesh of the box surfaces for sampling field

Solving the model in MATLAB-COMSOL environment

When the model is built in the graphic user interface, the model will be solved by the main MATLAB script that operates in the COMSOL-MATLAB environment (see Section 2.2).

4.2 Main MATLAB script — retrieving using RETOP

In the main MATLAB script “***main_script_sphere.m***”, two major tasks are executed: 1/solving the COMSOL model and 2/retrieving the angular distribution of the flux of the scattered field in free space. The electromagnetic field obtained from the COMSOL model is transferred to **RETOP** via a MATLAB subroutine. As the main script solves the COMSOL model, this script runs in the COMSOL-MATLAB environment, which requires the COMSOL feature *Livelink for MATLAB*.

Solving the COMSOL model

This part corresponds to ‘**Part 1’** in the main script. To solve the COMSOL model (i.e., Maxwell’s equation defined in the model) in COMSOL-MATLAB environment, a few script commands are written according to the script syntax documented in the COMSOL official user guides "COMSOL Livelink for MATLAB User Guide" and "COMSOL Java API Reference Guide".

Retrieval via RETOP

This part corresponds to Part 2 in the main script.

4.3 MATLAB subroutine — interfacing COMSOL and the main script

The subroutine '***extract_comsol_field.m***' interfaces the COMSOL model sheet and **RETOP** by transferring the electromagnetic field of COMSOL on the 'Box' to **RETOP**. The subroutine is called by the ***retop.m*** function in the main Matlab script. The user is allowed to specify the type of field (full field or scattered field) to be extracted from the model sheet.

5. EXAMPLE 2: EMISSION OF TWO DIPOLES IN A DIELECTRIC SLAB WAVEGUIDE

In this example, the construction of COMSOL model sheet ***double_dipole_in_waveguide_web.mph***, details of the Matlab main script ***main_script_dipoles_in_waveguide.m***, and the Matlab subroutine ***extract_comsol_field.m*** for transferring electromagnetic field of COMSOL to RETOP, are minutely explained in **subsections 5.1**, **5.2** and **5.3**, respectively. The example is sketched in Fig. 9, and more details can be found in Example-2 in Ref. [1].

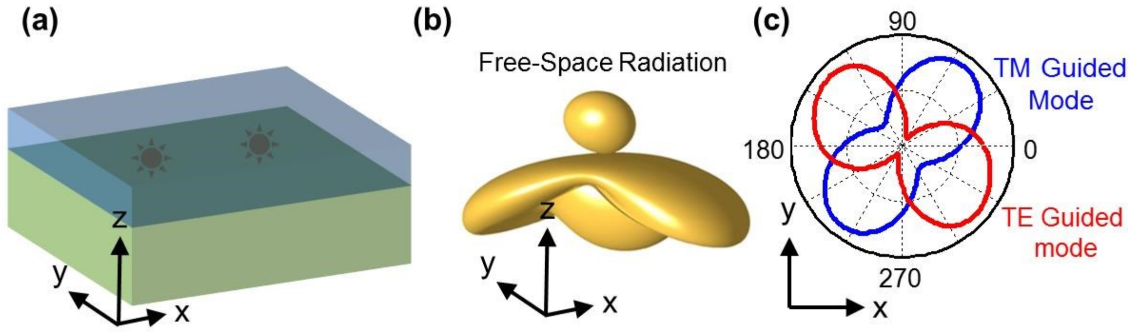


Figure 9. Radiation diagrams of the emitted field of two dipole sources embedeed in a dielectric slab waveguide. (**a**) Sketch of the configuration. (**b**) Free-space radiation diagram retrieved via RETOP. (**c**) Retrived guided-mode radiation diagrams for the fundamental TE and TM modes, respectively.

5.1 Instructions on the COMSOL Model Sheet

The model sheet for the current example is ***double_dipole_in_waveguide_web.mph***. To simulate dipole emission problems, we use the full-field formulation in COMSOL, in contrast to **Example 1**. For this purpose, select "Full Field" in the "Settings" of the "Electromagnetic Waves, Frequency Domain Interface".

Definitions of the parameters of the model

In the model sheet ***double_dipole_in_waveguide_web.mph***, the parameters are defined as below.

(1) **c**: the speed of light in vacuum

(2) **lambda** & **frequency**: operation wavelength & frequency

(3) **t_pml**: PML thickness (~1/3 wavelength)

(4) **radius_pml**: outer radius of the spherical PML

(5) **eps_top, eps_mid** & **eps_low**: relative permittivities of the slab waveguide. 'top', 'mid' and 'low' label the superstrate (upper cladding), core, and substrate (lower cladding).

(4) **thickness**: thickness of the dielectric core

(7) **Lx, Ly,** & **Lz**: size of the cuboid, on which the total field is sampled and will be sent to **RETOP** for retrieving the free-space and guide-mode radiation-patterns

(8) **current**: the amplitude of electric field of dipole source

(9) **x1** & **x2**: the *x*-coordinate of the two electric dipoles

(9) **block**: the size of a box that will be used to define waveguide geometry in the 'Geometry'

The geometry of the model

(1) **Sphere 2, Sphere 3,** & **Sphere 4**: the 3 spheres are used to define the 3 layers (semi-infinite upper-cladding, core, and semi-infinite low-cladding) of the waveguide.

(2) **Block 1**: to define the box for sampling field

(3) **Block 2** & **Block 3**: (**Block 2 + Sphere 2**) and (**Block 3 + Sphere 3**) are combined to create the upper & lower claddings using the Boolean operation "intersection".

(4) **Intersection 2** & **Intersection 3**: convert (**Block 2 + Sphere 2**) and (**Block 3 + Sphere 3**) into the upper & lower claddings.

(5) **Point 1** & **Point 2**: the locations of the dipole sources

Definitions of the model materials

The model considers light emission of two dipole sources embedded in a dielectric slab waveguide, so there are three materials (two semi-infinite claddings and the core layer).

(1) **Material 1**: the relative permittivity and permeability of the upper cladding

(2) **Material 2**: the relative permittivity and permeability of the dielectric core layer

(3) **Material 3**: the relative permittivity and permeability of the lower cladding

The model mesh

Three different mesh sizes are used in different space regions and at the sampling 'box'.

(1) **Size 1** is related to the mesh of the regions of the host medium and the PML

(2) **Size 2** is related to the mesh of the region where the dipoles are placed

(3) **Size 3** is related to the mesh of the box surfaces for sampling field

(4) **Size 4** is related to the mesh of the dipole sources

Solving the model in MATLAB-COMSOL environment

When the model is built in the graphic user interface, it is solved by the main MATLAB script that operates in the COMSOL-MATLAB environment (see Section 1.6).

5.2 Main MATLAB script —– retrieving using RETOP

In the main MATLAB script ‘***main_script_dipoles_in_waveguide.m***’, two major tasks are executed: 1/solving the COMSOL model and 2/retrieving the angular distribution of the flux of the total field in free space. The electromagnetic field obtained from the solved COMSOL model is transferred to **RETOP** via a MATLAB subroutine (see Section 1.6 for details). As the main script solves the COMSOL model, this script runs in the COMSOL-MATLAB environment, which requires the COMSOL feature *Livelink for MATLAB*.

Solving the COMSOL model

This part corresponds to **Part 1** in the main script. To solve the COMSOL model (i.e., Maxwell’s equation defined in the model) in COMSOL-MATLAB environment, a few script commands are written according to the script syntax documented in the COMSOL official user guides “COMSOL Livelink for MATLAB User Guide” and “COMSOL Java API Reference Guide”.

Retrieval via RETOP

This part corresponds to Part 2 in the main script.

MATLAB subroutine —– interfacing COMSOL and the main script

The subroutine, entitled ‘***extract_comsol_field.m***’ interfaces COMSOL model sheet and **RETOP** by transferring the electromagnetic field of COMSOL to **RETOP**. It is called by the ***retop.m*** function in the main Matlab script. Also one is allowed to specify the type of field (full field or scattered field) to extract from the model sheet.

6. Remarks for advanced users

6.1 Function ‘retgauss.m’

Legendre-Gauss quadrature is a numerical integration method also called the Gaussian quadrature or Legendre quadrature. The abscissas for quadrature order m are given by the roots of the Legendre polynomials $P_m(x)$, which occur symmetrically about 0.

The Matlab function ‘**retgauss.m’** generates the abscissas, eventually repeating identical zones.

```
-------------------------------------------------------------------------------------------
[xx,wx,x_disc]=retgauss(a,b,n,n_zone,xd);
-------------------------------------------------------------------------------------------
```

[a, b]	: interval a<x<b (we may have a>b)
n	: positive integer, number of sampling points per zone
n_zone	: positive integer, number of zones
xd	: abscissas of the discontinuities (the abscissas are periodic if the period **d** is specified)

xx	: vector, abscissas of sampling points
wx	: vector, weights
x_disc	: abscissas of the points separating the zones

```
------------------------------------------------------ example (begin) ------------------------------------------------------------
% computation of a surface integral ∬ cos(5πx/a) cos(7πy/b) dxdy
a=2;b=1;[x,wx]=retgauss(-a,a,10,5);[y,wy]=retgauss(-b,2*b,12,4);
result=(sum(wx.*cos(5*pi*x/a).^2)) *(sum(wy.*cos(7*pi*y/b).^2))
------------------------------------------------------ example (end) ------------------------------------------------------------
```

6.2 Stacks with magnetic layers

Under development.

6.3 Stacks with anisotropic layers

Under development.

7. Frequently asked questions

How to check that the radiation diagrams are safely calculated?

We do not have a good response to this good question. Perhaps try to compute the total power radiated, the absorption, the energy of the incoming field, and try to check energy balance.

Also try to increase N in azimuthal_orders to check that stable results are obtained.

Can we use RETOP with non-reciprocal materials?

The answer is no. Fundamentally the NFFT used in RETOP relies on reciprocity arguments [1], implying that the permittivity and permeability tensors of the stack should be symmetric.

For lossy claddings, can we calculate the far-field radiation diagram?

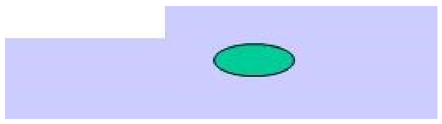
The answer is no. For instance if the substrate absorbs light, one cannot define a radiation diagram in the substrate. The field is zero in the far-field because of absorption. However one may calculate the radiation diagram in the upperstrate (if it is not lossy). Nothing prevents to calculate the guided-wave radiation diagram, even if the substrate and the upperstrate are both lossy. However care should be taken to give a proper meaning based on energy-consideration to the guided-mode expansion, see [1] and an ACS photonics paper published in 2016 by our group.

For lossy layers, can we calculate the far-field radiation diagram?

The answer is yes. For lossy layers (not including the upper and lower half space), both far-field and guided-wave radiation diagrams can be calculated with **RETOP**.

Can we calculate the far-field radiation diagram for a step index substrate?

Consider the following case, in which a green scatterer is embedded in a step like substrate.



We cannot use **RETOP** to calculate the far-field or guided radiation diagrams. The reason is simple: we cannot define a box that surrounds the scatterer, in such a way that outside the box, the refective-index distribution is that of a thin-film stack.

When considering a stack of layers, does the box surrounding the scatterers need to be embedded in one of the layers, or can the scatterers go through more than one?

The scatterers can be embedded in several layers including the claddings. All the scatterers need to be inside the box, implying that outside the box, the system is a pure thin film stack.

When the dimensions of the box are changed (while it still satisfies the conditions i.e. it fully surrounds the inhomogeneity), the results changes considerably. Is there particular box dimensions that leads to correct results?

The results of the plane wave expansion are theoretically independent of the box dimension (provided that all the scatterers are inside the box), and numerically almost independent of the box dimension (a three digits accuracy, at least, should be obtained as the box size is varied). The most frequent error is that the z-positions of the layer interfaces, vector z_layers=[z_1,..z_{m-1}], or the box center, vector box_center=[x,y,z] are simply wrong. Pay attention to their units. Pay also attention to the refractive-index vector, refractive_indices= [n_1,..n_m] ordered *from top to bottom* like the vector [z_1,..z_{m-1}]. If all these tests are correctly performed and, yet the results of the plane wave expansion depend on the box dimensions, then the probability that the field you computed is wrong becomes high.