Carlo Gigli *March 2020*

COMSOL with MATLAB interface for nanoantenna scattering

User Guide

Contents

The subject of this guide is an implementation of COMSOL with MATLAB interface for computation of linear scattering properties of $Al_{18}Ga_{82}As$ nanoantennas. The numerical tool explained in the following contains the following files:

- A COMSOL model 'NanodiskScattering.mph' for the solution of linear scattering problem for a fixed geometry and wavelength
- A Matlab object 'Model.m' linked to the COMSOL model
- A Matlab main script to run parametric simulations and extract results.

1. COMSOL model overview

COMSOL model is based on tutorial examples scattering_nanosphere and scatterer_on_substrate in COMSOL application library. Scattering problems at vacuum wavelength $\lambda_1 = \lambda_{FF}$ is modeled with the RF frequency domain in COMSOL. We could resume the entire process as follows: an external plane wave $[\mathbf{E}_{inc}, \mathbf{H}_{inc}]$ is impinging on a nanoresonator lying on a substrate. The infinite open space of the scattering problem is replaced by a finite space surrounded by perfect matching layers which absorbs all the outgoing waves. The total electric field $[\mathbf{E}_t, \mathbf{H}_t]$ is computed through the scattered field formulation of the electromagnetic wave frequency domain (emw) module in COMSOL. Refractive index for AlGaAs is interpolated from empirical model in [1]. In the following three section we will provide details on model geometry, background field at pump frequency and nonlinear current definitions respectively.

1.1 Model geometry

Resonators under study are nanocylinders with elliptical basis in the XY plane. We will call in the following r_a and r_b the major and minor semiaxis respectively and θ_{pil} the orientation of major semiaxis with respect to x axis. A general sketch is shown in Figure 1.

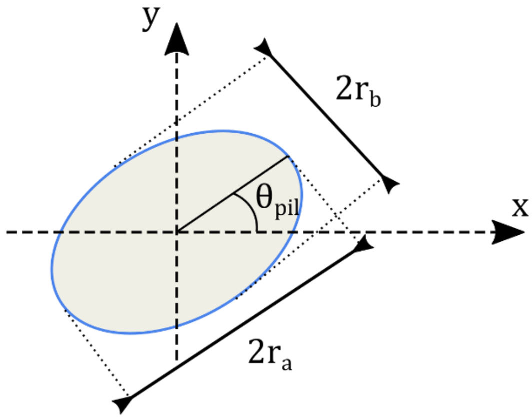

Figure 1 Nanocylinder base parameters

The cylinder with height h_0 is lying on a substrate with refractive index n_s, while the upper space has a refractive index n_c. The interface between the two material is placed at $z = z_0$. The PML-mapped space is a sphere with internal radius r_{pml}, while PML thickness is t_{pml} typically equal to $\lambda_{FF}/2$.

1.2 Definition background field

We can define the background field as the total electromagnetic field when the nanoresonator is removed, i.e. the solution of a Fresnel reflection problem at the interface (at $z = z_0$) between a cladding material, with refractive index n_c, and a substrate, with refractive index n_s. Referring to the coordinate system in Figure 2, the background field can be written as:

$$
\begin{aligned}
\mathbf{E}_b(x,y,z>z_0) = E_s & \left[e^{-in_c(k_x^i x - k_y^i y - k_z^i z)} + r_s e^{-in_c(k_x^i x - k_y^i y + k_z^i z)}\right][cos(\varphi)\hat{\boldsymbol{x}} + sin(\varphi)\hat{\boldsymbol{y}}] \\
& + E_p\left[e^{-in_c(k_x^i x - k_y^i y - k_z^i z)} + r_p e^{-in_c(k_x^i x - k_y^i y + k_z^i z)}\right][cos(\theta)cos(\varphi)\hat{\boldsymbol{y}} - cos(\theta)sin(\varphi)\hat{\boldsymbol{x}} \\
& - sin(\theta)\hat{\boldsymbol{z}}]
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{E}_b(x,y,z<z_0) = E_s & \left[t_s e^{-in_s(k_x^t x - k_y^t y - k_z^t z)}\right][cos(\varphi)\hat{\boldsymbol{x}} + sin(\varphi)\hat{\boldsymbol{y}}] \\
& + E_p\left[t_p e^{-in_s(k_x^t x - k_y^t y - k_z^t z)}\right][cos(\theta_t)cos(\varphi)\hat{\boldsymbol{y}} - cos(\theta_t)sin(\varphi)\hat{\boldsymbol{x}} - sin(\theta_t)\hat{\boldsymbol{z}}]
\end{aligned}
$$

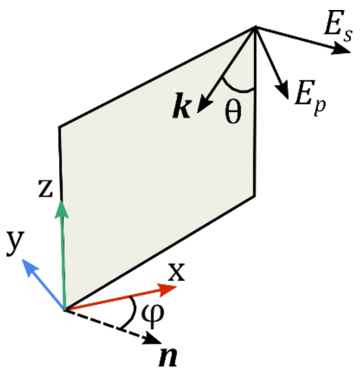

Figure 2 Coordinate reference system

In previous equation we defined:

- θ, θ_t incident and transmitted angle satisfying Snell's law: $\theta_t = \sin^{-1}(n_c sin(\theta)/n_s)$
- φ azimuthal angle between incident plane normal $\hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{x}}$ versor
- $n_c\boldsymbol{k}^i$, $n_s\boldsymbol{k}^t$ incident and transmitted wavevectors

- E_s, E_p s-(TE) and p-(TM) components of incident plane wave. We could define an additional angle γ and represent any polarization as a superposition of $E_s = E_0 \sin(\gamma)$ and $E_p = E_0 \cos(\gamma)$. This parameter is called pol_Eb in COMSOL model (see sec. 1.3)

- r_s, r_p, t_s, t_p Fresnel reflection and transmission coefficients for TE and TM polarizations.

These definitions can be found inside the three context menus 'ElectricField FF', 'BackgroundFieldFF_Air' and 'BackgroundFieldFF_Substrate'.

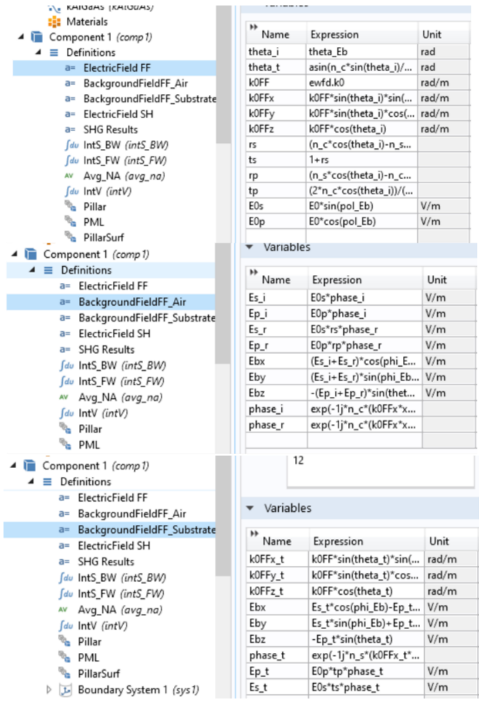

Figure 3

1.3 COMSOL parameters summary

We report in the following table all the parameters define in the COMSOL model with a brief description:

Parameter	Description
lambdaFF	Working pump wavelength
ra	Elliptical base major semiaxis (see r_a sec.1.1)
rb	Elliptical base minor semiaxis (see r_b sec.1.1)
h0	Nanocylinder height (see h_0 sec.1.1)
r_pml	Perfectly matched layer internal radius
t_pml	Perfectly matched layer thickness
I0	Pump wave intensity
E0	Pump wave electric field amplitude
theta_pil	Ellipse major semiaxis orientation with respect to x axis (see θ_{pil} sec.1.1)
theta_Eb	Incident angle with respect to z axis (see θ sec. 1.2)
phi_Eb	Orientation of incident plane normal with respect to x axis (see φ sec. 1.2)
pol_Eb	Angle defining incident plane wave polarization ($0° = p/TM$ - $90° = s/TE$).
n_c	Cladding material refractive index
n_s	Substrate material refractive index
z0	Cladding-Substrate interface z-coordinate
Lbox	Box edge size to extract near field for near to far field transformation in RETOP (see sec. 3.6)
zbox	Box center z-coordinate to extract near field for near to far field transformation in RETOP (see sec. 3.6)

2. Matlab Model class overview

The Matlab object is the core of the present tool. As any class in Matlab it contains properties and methods briefly described in the table here below:

Property name	Description
name	Comsol model name
mph	Comsol livelink
linear_study_name	Linear study tag in COMSOL model for computation of fundamental frequency scattering (default = 'std1')
geometry_name	COMSOL model geometry name (default : 'geom1')
linearmesh_name	Mesh name in COMSOL model for linear study (default: 'mesh1')
dsetFF	Dataset name for linear study solution in COMSOL model (default: 'dset1')
res_domain	Resonator domain index in COMSOL model (used in multipolar decomposition)
logfile	Log file name to store temporary solutions
params	Structure containing all the parameter of the scattering problem

Method name	Description
load	It loads the COMSOL with Matlab livelink and it initialize the property mph containing all the COMSOL functions
init_logfile	It initializes the log file with the current simulation parameters. It takes in input the ID of the logfile
reset_params	It sets all the parameters in COMSOL model. It takes in input a structure containing all the new parameters
solve	It solves the linear and nonlinear studies for the set parameters
sca_efficiency	It extracts from the COMSOL model the linear scattering efficiency
plot_nearfieldXZ	It plots the field component specified by the user in a XZ cutplane
plot_nearfieldYZ	It plots the field component specified by the user in a YZ cutplane
compute_farfieldRETOP	It implements RETOP open source code to extract the farfield
Extract_comsol_field	Subroutine called by the method compute_farfieldRETOP to extract the nearfield in a box with sizes (Lbox,Lbox,Lbox) from the COMSOL model to implement near to far field transformation
compute_multipoles	It implements the multipoles decomposition in vector spherical wave functions (valid just for antennas in homogeneous media)
extract_field	Subroutine called by the method compute_multipoles to extract the near field inside the resonator domain labelled with the property res_domain

3. Model class methods

In this section all the different methods listed before will be analyzed and explained in detail.

3.1 Load COMSOL model

The first step to use the COMSOL model in Matlab is to load COMSOL with Matlab LiveLink library and mph file. This can be done using the method model.load() as it follows:

```
model = Model; % It initialize the object model with Matlab class Model
model.name='NanodiskScattering.mph'; % COMSOL mph file name
model = model.load(); % It loads COMSOL with Matlab LiveLink and mph file
```

3.2 Initialize log file

It can be useful to open a log file to store the initial simulation parameters and progressive simulation results in order not to lose all the work if any error will occur during the simulation. The method model.init_logfile(logID) just write down in a previously opened logfile with handle logID. A typical syntax is:

```
selpath = uigetdir(path,'Choose folder destination for log file');
logID = fopen([selpath,'\',model.logfile],'a+');
model.init_logfile(logID);
```

3.3 Setting simulation parameters

Most of COMSOL model parameters summarized in sec. 1.3 are accessible from the Matlab Model class property Model.params. This property is a Matlab structure containing the following fields:

Field name	Unit	Description
lambdaFF	[m]	Working pump wavelength. (See lambdaFF sec. 1.3)
ra	[m]	Elliptical base major semiaxis (See ra sec. 1.3)
rb	[m]	Elliptical base minor semiaxis (See rb sec. 1.3)
h0	[m]	Nanocylinder height (See h0 sec. 1.3)
I0	[W/cm^2]	Pump wave intensity (See I0 sec. 1.3)
theta_pil	[deg]	Ellipse major semiaxis orientation with respect to x axis (See theta_pil sec. 1.3)
theta_Eb	[deg]	Incident angle with respect to z axis (See theta_Eb sec. 1.3)
phi_Eb	[deg]	Orientation of incident plane normal with respect to x axis (See phi_Eb sec. 1.3)
pol_Eb	[deg]	Angle defining incident plane wave polarization ($0° = p/TM$ - $90° = s/TE$). (See pol_Eb sec. 1.3)
n_c	[1]	Cladding material refractive index. (See n_c sec. 1.3)
n_s	[1]	Substrate material refractive index. (See n_s sec. 1.3)

The suggested procedure to reset simulations parameters is to define a structure with these same fields, we will call it params here below. Once the desired parameters have been set in this structure, the Matlab Model class and the COMSOL model can be updated through the method model.reset_params as it follows:

```
model = model.reset_params(params);
```

3.4 Solve a nonlinear scattering problem

Once the model parameters are correctly set, the numerical simulation can be solved with the method model.solve() as it follows:

```
model.solve();
```

This method runs COMSOL geometry and mesh, and after this it runs the study at pump frequency.

To extract the results on linear scattering cross sections, the method model.sca_efficiency() can be adopted. This function will give in output the linear scattering efficiency defined as the scattering cross section normalized by the geometrical cross section $\sigma_{sca}/\sigma_{geom}$, with:

$$\sigma_{sca} = \frac{1}{2I_0} \oiint_{\Sigma} Re(\mathbf{E}_S \times \mathbf{H}_S^*) \cdot \hat{\boldsymbol{n}} d\sigma$$

where Σ is the internal surface of PML mapped space.

Typical Matlab syntax to solve problems and extract results will be therefore:

```
model.solve();
[Csca] = model.sca_efficiency()
```

3.5 Near field plots

We will focus here on the function plot_nearfieldXZ but the explanation is valid for plot_nearfieldYZ too. The general syntax to call this mehod is:

```
[Xp,Zp,field_XZ,hAx] = model.plot_nearfieldXZ(field_component,dataset)
```

Input	Type	Description
model	Model object	Model object initialized in the main script as model = Model;
field_component	String	string array containing the COMSOL name of field to evaluate, e.g. ‘real(emw.Ex)’
dataset	String	dataset label inside the COMSOL model to extract result, this value can be contained inside the model properties model.dsetFF

Output	Type	Description
Xp	2D double array	Mesh evaluation point along X axis. This matrix is the result of a meshgrid.
Zp	2D double array	Mesh evaluation point along Z axis. This matrix is the result of a meshgrid.
field_Xz	2D double array	Values of required field component in [Xp,Zp] points. This matrix has the same
hAx	Plot handle	Plot handle of automatic generated plot

3.6 Near to far field transformation

The near to far field transformation is implemented with the open source tool RETOP. The package can be downloaded from https://www.lp2n.institutoptique.fr/equipes-de-recherche-du-lp2n/light-complex-nanostructures and it has to be included in the same folder as the Matlab codes. A bibliographic reference can be found in [2], please acknowledge this work if you use this code. The general syntax to call this mehod is:

```
[angles_up,angles_dn] =
model.compute_farfieldRETOP(wavelength,physics,theta_arr,phi_arr,dataset)
```

Input	Type	Description
model	Model object	Model object initialized in the main script as model = Model;
wavelength	Double	Working wavelength for the near to far field transformation. This value is generally related to the property model.params.lambdaFF
physics	String	Physics label for the extraction of near field. Typical values: ‘emw.’,’emw.rel’
theta_arr	1D double array [1x3]	Polar evaluation angle, this 3 elements vector contains the initial and final angles (in radians) and the number of evaluation points: $[\theta_i, \theta_f, N_t]$. $\theta_{i,f}$ are limited to the $[0, \pi/2]$ interval

phi_arr	1D double array [1x3]	Azimuthal evaluation angle, this 3 elements vector contains the initial and final angles (in radians) and the number of evaluation points: $[\varphi_i, \varphi_f, N_p]$. $\varphi_{i,f}$ are in the $[0,2\pi]$ interval
dataset	String	dataset label inside the COMSOL model to extract result, this value can be contained inside the model properties model.dsetFF and model.dsetSH

Output	Type	Description
angles_up	Struct	Structure containing the results of near to far field transformation in the upper direction (z>0). A detailed description of structure fields is given below.
angles_dn	Struct	Structure containing the results of near to far field transformation in the downward direction (z<0). A detailed description of structure fields is given below.

Structures angles_up and angles_dn contain the following fields:

- theta: 1D array with length $N_t \times N_p$ containing polar evaluation angles
- delta: 1D array with length $N_t \times N_p$ containing azimuthal evaluation angles
- k,u,v: 2D arrays with dimensions $[N_t \times N_p, 3]$ describing unitary vectors components in cartesian coordinates. k, u and v denote direction and polarization of outgoing plane waves.
- F: 1D array with length $N_t \times N_p$ containing Poynting vector intensity at evaluation angles theta,delta
- EH: 2D arrays with dimensions $[N_t \times N_p, 6]$ describing electric and magnetic field cartesian components at evaluation angles. The six columns denote respectively $E_x, E_y, E_z, H_x, H_y, H_z$
- ug, vg,wg: 2D arrays with dimensions $[N_t\ , N_p]$ containing x,y and z cartesian components of k vector reshaped with the same mesh defined by theta_arr and phi_arr. These three variables contain the same values of k field but they are already reshaped to build the far field plot.
- EE, HH: 2D arrays with dimensions $[N_t \times N_p, 2]$ containing electric and magnetic field components of outgoing plane waves along k directions. The two columns refer to polarization along u and v respectively.

Example:

This example show how to compute and plot far field radiation diagram of SH emission sampling the 4π solid angle with 80 polar angles from 0 to π (40 from 0 to $\pi/2$ for the upper and downward space respectively) times 50 azimuthal angles from $-\pi$ to π.

```
[angles_up,angles_dn] = model.compute_farfieldRETOP
(model.params.lambdaFF/2,'emw.',[0,pi/2,40],[-pi,pi,50],model.dsetSH);

figure()

Fh=reshape(angles_up.F,size(angles_up.ug));              % the flux at each
solid angle, upper space ("h"-->upper)

Fb=reshape(angles_dn.F,size(angles_dn.ug));              % the flux at each
solid angle, lower space ("b"-->lower)

surf(angles_up.ug.*Fh,angles_up.vg.*Fh,angles_up.wg.*Fh,Fh,'LineStyle','none
');

hold on
```

```
surf(angles_dn.ug.*Fb,angles_dn.vg.*Fb,-
angles_dn.wg.*Fb,Fb,'LineStyle','none');
set(gca,'projection','perspective');
axis tight;axis equal;xlabel('x');ylabel('y');
shading interp
colorbar
lighting PHONG;
light('position',[5,1,1],'color','w');
light('position',[5,1,1],'color','y');
view(60,20)
axis off
```

3.7 Multipolar Decomposition

Multipolar decomposition in vector spherical wave functions is performed with the method described in [3] knowing the near field distribution inside the resonator. Please acknowledge this work if you use this method. Once solved the scattering problem with the method model.solve(), the total near field is extracted from COMSOL model, centered with respect to the origin of coordinate system and projected onto spherical harmonic functions.

N.B. This decomposition is valid only for resonators in embedded uniform media. If you try to decompose the near field of a resonator in a layered medium the program will stop with an error.

A typical syntax relies on a double method call, the first one to extract the desired near field and the second to compute the expansion:

```
[field,mesh] = model.extract_field(physics,self.dsetFF);
[Csca,Cse,Csm] = model.compute_multipoles(field,mesh,lmd,lmax)
```

Input	Type	Description
model	Model object	Model object initialized in the main script as model = Model;
physics	string	Physics label for the extraction of near field. Typical values: ‘emw.’,’emw.rel’
lmd	Double	Working wavelength for the near to far field transformation. This value is generally related to the property model.params.lambdaFF
lmax	Int	Maximum orbital momentum to stop the decomposition (l=1 : dipoles, l=2 quadrupoles, etc…)

Output	Type	Description
Csca	Double	Total scattering efficiency
Cse	1D double array [1xlmax]	Electric multipoles contribution to the scattering. All the terms with same orbital index l and different azimuthal index m have been summed together.
Csm	1D double array [1xlmax]	Magnetic multipoles contribution to the scattering. All the terms with same orbital index l and different azimuthal index m have been summed together.

The built-in routine model.multipolesFF(lmax) implements the two previously presented subroutines in a whole function.

4. How to acknowledge

We kindly ask that you reference this toolbox from Université de Paris and its subroutines in any publication/report for which you used it. The preferred citation for this toolbox is the GitHub repository https://github.com/carlogigli/NanoantennasScattering

For the multipolar decomposition please cite reference [3] and for near-to-far field transformations cite reference [2].

5. References

1. S. Gehrsitz, F. K. Reinhart, C. Gourgon, N. Herres, A. Vonlanthen, and H. Sigg, "The refractive index of AlxGa1-xAs below the band gap: Accurate determination and empirical modeling," J. Appl. Phys. **87**(11), 7825–7837 (2000).
2. J. Yang, J. P. Hugonin, and P. Lalanne, "Near-to-Far Field Transformations for Radiative and Guided Waves," ACS Photonics **3**(3), 395–402 (2016).
3. P. Grahn, A. Shevchenko, and M. Kaivola, "Electromagnetic multipole theory for optical nanomaterials," New J. Phys. **14**(093033), 1–11 (2012).