A-FMM User Guide

Marco Passoni

Chapter 1

Reference Guide

In this chapter a detailed explanation of the module is provided. For each submodule a summary of the classes and function involved can be found, together with details on the most common functions.

1.1 Module composition

The A-FMM module in turn composed by different submodules:

- creator module: Contain the creator class, used to define the dielectric costant of the single layer.
- layer module: Contain the layer class, used to solve the Maxwell equation inside each single layer. It requires a creator instance to be initialized.
- scattering module: Contain the S_matrix class. All method for scattering matrix creation and manipulation are here implemented.
- satck module: Contain the stack class, used to calculate the Scattering Matrix and related quantities of the hole structure. It is initialize from a collection of layer instances.
- sub_sm module: Contains auxiliary function that are called in more than one module. It is non loaded by default when loading A_FMM.

1.2 Scattering module

1.2.1 S_matrix class

Definition ad initialization

This class contains the definition of the scattering matrix object and all method for scattering matrix manipulation. The scattering matrix is an object relating the amplitudes of the incoming end outgoing modes of a structure. It presets itself as following:

$$\begin{bmatrix} u' \\ d \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} u \\ d' \end{bmatrix} \tag{1.1}$$

following the convention of figure 1.1. An instance is initialized as:

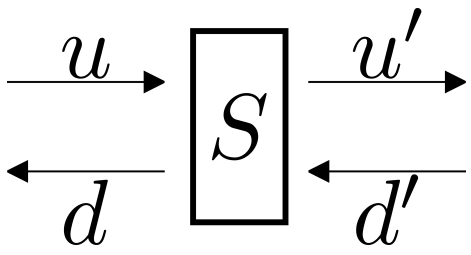

Figure 1.1: Convention adopted for the scattering matrix formalism.

S=S_matrix(N)

where `N` is the dimension of the four $N \times N$ matrices composing the scattering matrix. Each of this matrix is saved as an `numpy.ndarray` as an attribute of the S_matrix object. For example to the $(1, 2)$ element of the S_{21} matrix is accessed as `S.S21[1,2]`.

Method

`add(s)`

Takes as input an other instance of the S_matrix class s, which is then joined to the right to the scattering matrix that calls the method.

`add_left(s)`

Same as `add` but s matrix is joined to the left.

`add_uniform(lay,d)`

Takes as input:

- `lay`: An instance of the layer class;
- `d`: A float representing the thickness of the layer.

This method creates the scattering matrix for propagation trough a layer `lay` of thickness `d`, and then joins it to the right of the scattering matrix that calls the method.

`add_uniform_left(lay,d)`

Same as `add_uniform` but joins to the left.

1.3 Layer Module

1.3.1 Layer class

Initialization

This module contains several implementation of the layer class, which is responsable for definition of the single layer inside the scattering matrix approach. It also contains the methods for adding the coordinate transform an plotting of the mode fields. The different layer are:

`lay=layer(Nx,Ny,creator,Nyx=1.0)`

which is the standard implementation of the class. A `creator` object is needed for the creation, and all the calculation for the Fourier transform are done analytically. The parameter for the initialization are:

- `Nx,Ny`: Trucation order respectively in x and y direction. Actual number of plane waves is $(2N_x+1)(2N_y+1)$.
- `creator`: The creator instance describing the dielectric function.
- `Nyx`: Ratio between the cell period in y and x direction (default is 1.0).

`lay=layer_num(Nx,Ny,func,args=(),Nyx=1.0,NX=1024,NY=1024)`

Implementation in which the Fourier transforms for the layer are handled numerically. It is more versatile than the standard implementation, albeit should be less precise. The profile of the dielectric is obtained from the function `func`. The objects needed are:

- `Nx,Ny`: Integers, trucation order respectively in x and y direction. Actual number of plane waves is $(2N_x+1)(2N_y+1)$.
- `func`: Function which defines the dielectric constant. It has to be in the form `func=func(x,y,..)`. After x and y additional parameters can be defined in the function and passed in `args`. Domain of the function in -0.5:0.5 in x and -0.5*Nyx:0.5*Nyx in y.
- `Nyx`: Ratio between the cell period in y and x direction (default is 1.0).
- `args`: Tuple containing eventual additional parameters for `func`.
- `NX,NY`: Integers, number of point in each direction used for numerical integration in the Fourier transforms.

`lay=layer_uniform(Nx,Ny,eps,Nyx=1.0)`

Implementation of the uniform layer. It is implemented on is own because in this way its definition and solution are much faster. The parameters are:

- `Nx,Ny`: Trucation order respectively in x and y direction. Actual number of plane waves is $(2N_x+1)(2N_y+1)$.
- `eps`: The dielectric constant of the uniform layer. Can be complex.
- `Nyx`: Ratio between the cell period in y and x direction (default is 1.0).

When initializing the layer class all the matrices involve in the eigenvalue problem for the layer are created.

Methods

`trasform(ex=0,ey=0)`

Add the real coordinate transform.

- `ex,ey`: Respectively the width of the untransformed region in x and y direction. If not specified transformation in not applied in that direction.

`trasform_complex(ex=0,ey=0)`

Add the complex coordinate transform simulating PML boundary conditions.

- `ex,ey`: Respectively the width of the untransformed region in x and y direction. If not specified transformation in not applied in that direction.

`mode(k0,kx=0.0,ky=0.0,v=1)`

Solve the eigenvalue problem of the layer.

- `k0`: Energy of the mode
- `kx,ky`: Respectively the lateral wavevector in the x and y direction (in unit of inverse of the period).
- `v`: If equal to 0 only eigenvalue are calculated. Useful when interested only in propagation constant of the mode of the single layer. Default is 1.

create three new attributes of the class layer:

- `W`: Vector of the eigenvalues (effective indexes of modes squared).
- `V`: Matrix of electric eigenvectors of the modes. `V[:,i]` is the vector to the i^{th} mode.
- `VH`: Matrix of magnetic eigenvectors of the modes. Same convention as `V`.

`eps_plot(pdf=None,N=200,s=1.0)`

Plot dielectric function profile reconstructed from fourier transform.

- `pdf`: Name of the pdf file used to save the figure (string, without the .pdf).
- `N`: Width in pixel of the cell. Default 200.
- `s`: Number of fundamental cells plotted. Default 1.

`plot_E(pdf,i,N=100,s=1,func=np.abs)`

Plot electric field profile of selected mode.

- `pdf` Instances of the PdfPages class. Specify pdf file where to save field.
- `i` Number of mode to plot. Mode are ordered in decreasing effective index. Numeration start at 1.
- `N` Width in pixel of the cell. Default 100.
- `s` Number of fundamental cells plotted. Default 1.
- `func` Since field is complex, function to apply to field before plotting. Default is abs. Useful can be real, imag, angle.

`plot_H(pdf,i,N=100,s=1,func=np.abs)`

Plot magnetic field profile of selected mode. Same inputs as `plot_E`.

`plot_field(pdf,i,N=100,s=1,func=np.abs)`

Plot both electric and magnetic field profile of selected mode. Same inputs as `plot_E`.

`get_field(x,y,i,func=np.abs)`

Return field of selected mode at selected point.

- `x,y` Coordinate of the point in which to calculate fields.
- `i` Number of mode to plot. Mode are ordered in decreasing effective index. Numeration start at 1.
- `func` Since field is complex, function to apply to field before plotting. Default is abs. Useful can be real, imag, angle.

Return 1-dim array of length 4 containing in order: Ex,Ey,Hx,Hy

`get_P_norm()`

Calculate z component of Poynting vector, used in the scattering matrix algorithm to normalize correctly reflection ad transmission. The values of Poynting vector are then stored in the new attribute `P_norm`.

`T_interface(lay)`

Takes in input a different instance of the layer class. Return the transfer matrix representing the interface between the layer that calls the method and the layer given ad input. The obtained matrix is an instance of `numpy.ndarray`.

`T_prop(d)`

Takes in input a float value `d` representing the thickness of the layer. Return the transfer matrix of the propagation trough a thickness d of the layer that calls the method. Could generate numerical instabilities. The obtained matrix is an instance of `numpy.ndarray`.

`interface(lay)`

Takes in input a different instance of the layer class. Return the scattering matrix representing the interface between the layer that calls the method and the layer given ad input. The obtained matrix is an instance of `S_matrix`.