PyMoosh functions

Denis Langevin, Antoine Moreau

This document serves as a reference to know what PyMoosh can do. **It lists all (user-facing) existing functions**, sorted by which file they are defined in.

It references what are their arguments and returned objects, as well as which Notebooks explain how they work (if there is one).

Throughout the manuscript, r and t refer to the amplitude reflection and transmission coefficients, while R and T refer to the intensity coefficients. TE (resp. TM) corresponds to the polarization where the electric field (resp. magnetic field) is perpendicular to the plain of incidence. As we define z to be the axis perpendicular to the interfaces, and xOz to be the plane of incidence, this means $E = E_y$ in TE polarization (resp. $H = Hy$ in TM).

Figure 1: The PyMoosh logo! Don't hesitate to use it, if you credit PyMoosh.

1 `classes.py`

Main classes are defined here: Material, Structure, Beam, Window.
They are explained in `Basic_tutorials/PyMoosh_Basics`.

Name	Necessary Arguments	Returns	Comments
`Material`	mat	Material object	Used to define structures
`material.get_permittivity`	wavelength	Permittivity (complex) at wavelength	
`material.get_permeability`	wavelength	Permeability (complex) at wavelength	
`Structure`	materials, layer_type, thickness	Structure object	Used for most functions
`structure.plot_stack`	wavelength	None	Creates and shows a schematic of the structure
`Beam`	wavelength, incidence, polarization, horizontal_waist	Beam object	Used to compute fields
`Window`	width, beam_relative_position, horizontal_pixel_size, vertical_pixel_size	Window object	Used to compute fields

2 `core.py`

Most common functions are defined here, computing fields; r, t, R, T coefficients, and absorption in a structure.

Most are explained in `Basic_tutorials/PyMoosh_Basics`.

Name	Necessary Arguments	Returns	Comments
`absorption`	structure, wavelength, incidence, polarization	absorption, r, t, R, T	
`field`	structure, beam, window	Ey (TE) or Hy (TM)	
`fields`	structure, beam, window	Ey, Hx, Hz (TE) or Hy, Ex, Ez (TM)	
`coefficient`	structure, wavelength, incidence, polarization	r, t, R, T	just a wrapper, calls `coefficient_S` by default
`coefficient_S`	structure, wavelength, incidence, polarization	r, t, R, T	S matrix formalism
`absorption_S`	structure, wavelength, incidence, polarization, layers	absorbtion, r, t, R, T	absorption is computed in given layers, all if layers is None

3 vectorized.py

Vectorized versions of the core functions, to compute coefficients along spectral or angular ranges. Two versions exist: either you give min and max values, as well as a number of points (spectrally or angularly), or you directly give the wavelength or angle list you want to compute on.

They are used in `Basic_tutorials/PyMoosh_Basics`.

Linear Range

Name	Necessary Arguments	Returns	Comments
`spectrum`	structure, incidence, polarization, wl_min, wl_max, len_wl	wavelengths, r, t, R, T	Just a wrapper, calls S matrix formalism by default
`spectrum_S (resp. spectrum_A)`	structure, incidence, polarization, wl_min, wl_max, len_wl	wavelengths, r, t, R, T	S (resp. Abélès) matrix formalism
`angular`	structure, wavelength, polarization, theta_min, theta_max, len_an	angles, r, t, R, T	Just a wrapper, calls S matrix formalism by default
`angular_S (resp. angular_A)`	structure, wavelength, polarization, theta_min, theta_max, len_an	angles, r, t, R, T	S (resp. Abélès) matrix formalism

Custom range

Name	Necessary Arguments	Returns	Comments
`spectrum_list`	structure, incidence, polarization, wavelengths	r, t, R, T	Just a wrapper, calls S matrix formalism by default
`spectrum_S_list (resp. spectrum_A_list)`	structure, incidence, polarization, wavelengths	r, t, R, T	S (resp. Abélès) matrix formalism
`angular_list`	structure, wavelength, polarization, angles	r, t, R, T	Just a wrapper, calls S matrix formalism by default
`angular_S_list (resp. angular_A_list)`	structure, wavelength, polarization, angles	r, t, R, T	S (resp. Abélès) matrix formalism

4 `models.py`

Most common functions to define Material models. Beware of units! **Wavelengths are always expected in nm**, but other parameter units may vary depending on common use (typically, eV).

Their general idea is presented in `In-depth_examples/How_materials_works`

Name	Necessary Arguments	Returns	Comments
`BrendelBormann`	wavelength, f0, omega_p, Gamma0, f, omega, gamma, sigma	permittivity	An improvement on the Drude-Lorentz model
`Drude`	wavelength, omega_p, Gamma0	permittivity	Free electrons
`Lorentz`	wavelength, f, omega, gamma, eps	permittivity	Bound electrons
`DrudeLorentz`	wavelength, omega_p, Gamma0, f, omega, gamma	permittivity	Both!
`ExpData`	wavelength, wavelength_list, permittivity_list	permittivity	Interpolation on external data

5 modes.py

Functions to play around with guided modes.

They are detailed in `In-depth_examples/Finding_representing_guided_modes`.

Name	Necessary Arguments	Returns	Comments
`dispersion`	alpha, structure, wavelength, polarization	`1/|r|`	Computes the reflection coefficient at given kx (α)
`complex_map`	structure, wavelength, polarization, real_bounds, imag_bounds, n_real, n_imag	re(n), im(n), `1/|r(n)|`	Computes dispersion on a 2D map of complex n values
`guided_modes`	structure, wavelength, polarization, neff_min, neff_max	n(λ)	Computes the index of modes at given wavelength
`follow_guided_modes`	structure, wavelength_list, polarization, neff_min, neff_max	n(λ)	Computes the evolution of the index of modes with varying wavelength
`profile`	structure, n_eff, wavelength, polarization	x, E(x)	Computes the field along the structure thickness, for the given index

6 `photo.py`

Functions dedicated to photovoltaic computations.

`photo` is detailed in `In-depth_examples/Photovoltaics`

Name	Necessary Arguments	Returns	Comments
`solar`	wavelength	I(λ)	Solar irradiance, converted in A/cm^2
`photo`	structure, incidence, polarization, wl_min, wl_max, active_layers, number_points	conversion_efficiency, total_current, total_current_max, wavelength_list, photon_density, total_absorbed	short circuit currents across several active layers are summed
`opti_photo`	structure, incidence, polarization, wl_min, wl_max, active_layers, number_points	id.	id. but vectorized
`gx`	structure, incidence, polarization, wl_min, wl_max, number_points	x, g(x)	Density of absorbed photons in photon/s/m^2/nm

7 `alt_methods.py`

Other matrix formalisms to compute the main coefficients.

They are explained in the PyMoosh tutorial paper, and tested in `In-depth_examples/Adapting_matrix_formalism`

Name	Necessary Arguments	Returns	Comments
`coefficient_A`	structure, wavelength, incidence, polarization	r, t, R, T	Abélès formalism
`coefficient_T`	structure, wavelength, incidence, polarization	r, t, R, T	Transfer Matrix formalism
`coefficient_DN`	structure, wavelength, incidence, polarization	r, t, R, T	Dirichlet-to-Neumann formalism
`coefficient_I`	structure, wavelength, incidence, polarization	r, t, R, T	Impedance formalism
`absorption_A`	structure, wavelength, incidence, polarization	absorption, r, t, R, T	Abélès formalism

8 `optim_algo.py`

Tried and tested optimization algorithms. We have a deep-dive tutorial on ArXiv (but cite the article paper) about how to optimize, and the use of these functions is shown (with differential evolution as an example) in `Basic_tutorials/Optimizing_with_PyMoosh`

Name	Necessary Arguments	Returns	Comments
`differential_evolution`	f_cout, budget, X_min, X_max	best, convergence	DE!
`bfgs`	f_cout, npas, start	best, f_cout(best)	Gradient descent
`QODE`	f_cout, budget, X_min, X_max	best, convergence	
`QNDE`	f_cout, budget, X_min, X_max	best, convergence	

9 `anisotropic.py`

Equivalent to the main functions but for structures including anisotropic materials.
They are presented in `In-depth_examples/Anisotropic_structure`

Name	Necessary Arguments	Returns	Comments
`coefficients_ani`	structure, wl, theta_inc	t_pp, t_ps, t_sp, t_ss, r_pp, r_ps, r_sp, r_ss	
`AniStructure`	materials, layer_type, thickness, ani_rot_angle, ani_rot_axis	AniStructure object	Used to compute the reflection and transmission coefficients
`AniMaterial`	mat	AniMaterial object	Used to define the structures
`AniMaterial.get_permittivity_ani`	wavelength	[epsilon1, epsilon2, epsilon2]	

10 `non_local.py`

Equivalent to the main functions but for structures including non-local (spatially dispersive) materials.
They are presented in `In-depth_examples/Spatial_dispersion_aka_nonlocality`

Name	Necessary Arguments	Returns	Comments
`NLcoefficient`	structure, wavelength, incidence, polarization	r, t, R, T	Computes the reflection and transmission coefficients for a non-local structure
`NLdispersion`	alpha, struct, wavelength, polarization	1/|r|	Computes the reflection coefficient at a given kx (α)
`NLcomplex_map`	struct, wavelength, polarization, real_bounds, imag_bounds, n_real, n_imag	re(n), im(n), 1/|r(n)|	Computes dispersion on a 2D map of complex n values
`NLguided_modes`	struct, wavelength, polarization, neff_min, neff_max	modes	Computes the index of modes at given wavelength
`NLfollow_guided_modes`	structure, wavelength_list, polarization, neff_min, neff_max	`n(`λ`)`	Computes the evolution of the index of modes with varying wavelength
`NLStructure`	materials, layer_type, thickness	NLStructure object	Used to compute the reflection and transmission coefficients
`NLMaterial`	mat	NLMaterial object	Used to define the structures
`NLMaterial.get_permittivity`	wavelength	epsilon	
`NLMaterial.get_values_nl`	wavelength	beta2, chi_b, chi_f, omega_p	

11 `green.py`

The function computing the green Dyadic function for a point source. Yes, it's all alone and sad but it didn't make sense to put it anywhere else. It is presented in `In-depth_examples/Computing_Green_functions`

Name	Necessary Arguments	Returns	Comments
`green`	structure, window, lam, source_interface	E	Returns the field resulting from the source