modesolverpy Documentation

Release 0.4.4

Jean-Luc Tambasco

Feb 25, 2019

Contents

Contents:

CHAPTER 1

Introduction

This is the documentation for the modesolverpy API.

CHAPTER 2

API documentation

2.1 Mode Solvers

2.1.1 Functions

use_gnuplot()	Use gnuplot as the plotting tool for any mode related outputs.
use_matplotlib()	Use matplotlib as the plotting tool for any mode related outputs.
with_metaclass(meta, *bases)	Create a base class with a metaclass.

use_gnuplot

modesolverpy.mode_solver.**use_gnuplot**()
Use gnuplot as the plotting tool for any mode related outputs.

use_matplotlib

modesolverpy.mode_solver.**use_matplotlib**()
Use matplotlib as the plotting tool for any mode related outputs.

2.1.2 Classes

ModeSolverFullyVectorial(n_eigs[, tol, ...])	A fully-vectorial mode solver object used to setup and run a mode solving simulation.
ModeSolverSemiVectorial(n_eigs[, tol, ...])	A semi-vectorial mode solver object used to setup and run a mode solving simulation.

ModeSolverFullyVectorial

class modesolverpy.mode_solver.**ModeSolverFullyVectorial**(*n_eigs*, *tol=0.001*, *boundary='0000'*, *initial_mode_guess=None*, *n_eff_guess=None*)

Bases: modesolverpy.mode_solver._ModeSolver

A fully-vectorial mode solver object used to setup and run a mode solving simulation.

Parameters

- **n_eigs** (*int*) – The number of eigen-values to solve for.
- **tol** (*float*) – The precision of the eigen-value/eigen-vector solver. Default is 0.001.
- **boundary** (*str*) – The boundary conditions to use. This is a string that identifies the type of boundary conditions applied. The following options are available: 'A' - Hx is antisymmetric, Hy is symmetric, 'S' - Hx is symmetric and, Hy is antisymmetric, and '0' - Hx and Hy are zero immediately outside of the boundary. The string identifies all four boundary conditions, in the order: North, south, east, west. For example, boundary='000A'. Default is '0000'.
- **initial_mode_guess** (*list*) – An initial mode guess for the modesolver.
- **initial_n_eff_guess** (*list*) – An initial effective index guess for the modesolver.

Methods Summary

solve(structure)	Find the modes of a given structure.
solve_ng(structure[, wavelength_step, filename])	Solve for the group index, n_g, of a structure at a particular wavelength.
solve_sweep_structure(structures, ...[, ...])	Find the modes of many structures.
solve_sweep_wavelength(structure, wavelengths)	Solve for the effective indices of a fixed structure at different wavelengths.
write_modes_to_file([filename, plot, ...])	Writes the mode fields to a file and optionally plots them.

Methods Documentation

solve(*structure*)

Find the modes of a given structure.

Parameters **structure** (Structure) – The target structure to solve for modes.

Returns The 'n_effs' key gives the effective indices of the modes. The 'modes' key exists of mode profiles were solved for; in this case, it will return arrays of the mode profiles.

Return type dict

solve_ng(*structure*, *wavelength_step=0.01*, *filename='ng.dat'*)

Solve for the group index, n_g, of a structure at a particular wavelength.

Parameters

- **structure** (Structure) – The target structure to solve for modes.

- **wavelength_step** (*float*) – The step to take below and above the nominal wavelength. This is used for approximating the gradient of n_{eff} at the nominal wavelength. Default is 0.01.
- **filename** (*str*) – The nominal filename to use when saving the effective indices. Defaults to 'ng.dat'.

Returns A list of the group indices found for each mode.

Return type list

solve_sweep_structure(*structures*, *sweep_param_list*, *filename='structure_n_effs.dat'*, *plot=True*, *x_label='Structure number'*, *fraction_mode_list=[]*)

Find the modes of many structures.

Parameters

- **structures** (*list*) – A list of *Structures* to find the modes of.
- **sweep_param_list** (*list*) – A list of the parameter-sweep sweep that was used. This is for plotting purposes only.
- **filename** (*str*) – The nominal filename to use when saving the effective indices. Defaults to 'structure_n_effs.dat'.
- **plot** (*bool*) – *True* if plots should be generates, otherwise *False*. Default is *True*.
- **x_label** (*str*) – x-axis text to display in the plot.
- **fraction_mode_list** (*list*) – A list of mode indices of the modes that should be included in the TE/TM mode fraction plot. If the list is empty, all modes will be included. The list is empty by default.

Returns A list of the effective indices found for each structure.

Return type list

solve_sweep_wavelength(*structure*, *wavelengths*, *filename='wavelength_n_effs.dat'*, *plot=True*)

Solve for the effective indices of a fixed structure at different wavelengths.

Parameters

- **structure** (Slabs) – The target structure to solve for modes.
- **wavelengths** (*list*) – A list of wavelengths to sweep over.
- **filename** (*str*) – The nominal filename to use when saving the effective indices. Defaults to 'wavelength_n_effs.dat'.
- **plot** (*bool*) – *True* if plots should be generates, otherwise *False*. Default is *True*.

Returns A list of the effective indices found for each wavelength.

Return type list

write_modes_to_file(*filename='mode.dat'*, *plot=True*, *fields_to_write=('Ex'*, *'Ey'*, *'Ez'*, *'Hx'*, *'Hy'*, *'Hz')*)

Writes the mode fields to a file and optionally plots them.

Parameters

- **filename** (*str*) – The nominal filename to use for the saved data. The suffix will be automatically be changed to identifiy each field and mode number. Default is 'mode.dat'
- **plot** (*bool*) – *True* if plots should be generates, otherwise *False*. Default is *True*.

- **fields_to_write** (*tuple*) – A tuple of strings where the strings can be 'Ex', 'Ey', 'Ez', 'Hx', 'Hy' and 'Hz' defining what part of the mode should be saved and plotted. By default, all six components are written and plotted.

Returns A dictionary containing the effective indices and mode field profiles (if solved for).

Return type dict

ModeSolverSemiVectorial

class modesolverpy.mode_solver.**ModeSolverSemiVectorial**(*n_eigs*, *tol=0.001*, *boundary='0000'*, *mode_profiles=True*, *initial_mode_guess=None*, *semi_vectorial_method='Ex'*)

Bases: modesolverpy.mode_solver._ModeSolver

A semi-vectorial mode solver object used to setup and run a mode solving simulation.

Parameters

- **n_eigs** (*int*) – The number of eigen-values to solve for.
- **tol** (*float*) – The precision of the eigen-value/eigen-vector solver. Default is 0.001.
- **boundary** (*str*) – The boundary conditions to use. This is a string that identifies the type of boundary conditions applied. The following options are available: 'A' - Hx is antisymmetric, Hy is symmetric, 'S' - Hx is symmetric and, Hy is antisymmetric, and '0' - Hx and Hy are zero immediately outside of the boundary. The string identifies all four boundary conditions, in the order: North, south, east, west. For example, boundary='000A'. Default is '0000'.
- **mode_profiles** (*bool*) – *True if the the mode-profiles should be found, 'False* if only the effective indices should be found.
- **initial_mode_guess** (*list*) – An initial mode guess for the modesolver.
- **semi_vectorial_method** (*str*) – Either 'Ex' or 'Ey'. If 'Ex', the mode solver will only find TE modes (horizontally polarised to the simulation window), if 'Ey', the mode solver will find TM modes (vertically polarised to the simulation window).

Methods Summary

solve(structure)	Find the modes of a given structure.
solve_ng(structure[, wavelength_step, filename])	Solve for the group index, n_g, of a structure at a particular wavelength.
solve_sweep_structure(structures, ...[, ...])	Find the modes of many structures.
solve_sweep_wavelength(structure, wavelengths)	Solve for the effective indices of a fixed structure at different wavelengths.
write_modes_to_file([filename, plot, analyse])	Writes the mode fields to a file and optionally plots them.

Methods Documentation

solve(*structure*)
Find the modes of a given structure.

Parameters **structure** (Structure) – The target structure to solve for modes.

Returns The 'n_effs' key gives the effective indices of the modes. The 'modes' key exists of mode profiles were solved for; in this case, it will return arrays of the mode profiles.

Return type dict

solve_ng(*structure*, *wavelength_step=0.01*, *filename='ng.dat'*)
Solve for the group index, n_g, of a structure at a particular wavelength.

Parameters

- **structure** (Structure) – The target structure to solve for modes.
- **wavelength_step** (*float*) – The step to take below and above the nominal wavelength. This is used for approximating the gradient of n_{eff} at the nominal wavelength. Default is 0.01.
- **filename** (*str*) – The nominal filename to use when saving the effective indices. Defaults to 'ng.dat'.

Returns A list of the group indices found for each mode.

Return type list

solve_sweep_structure(*structures*, *sweep_param_list*, *filename='structure_n_effs.dat'*, *plot=True*, *x_label='Structure number'*, *fraction_mode_list=[]*)
Find the modes of many structures.

Parameters

- **structures** (*list*) – A list of *Structures* to find the modes of.
- **sweep_param_list** (*list*) – A list of the parameter-sweep sweep that was used. This is for plotting purposes only.
- **filename** (*str*) – The nominal filename to use when saving the effective indices. Defaults to 'structure_n_effs.dat'.
- **plot** (*bool*) – *True* if plots should be generates, otherwise *False*. Default is *True*.
- **x_label** (*str*) – x-axis text to display in the plot.
- **fraction_mode_list** (*list*) – A list of mode indices of the modes that should be included in the TE/TM mode fraction plot. If the list is empty, all modes will be included. The list is empty by default.

Returns A list of the effective indices found for each structure.

Return type list

solve_sweep_wavelength(*structure*, *wavelengths*, *filename='wavelength_n_effs.dat'*, *plot=True*)
Solve for the effective indices of a fixed structure at different wavelengths.

Parameters

- **structure** (Slabs) – The target structure to solve for modes.
- **wavelengths** (*list*) – A list of wavelengths to sweep over.

- **filename** (*str*) – The nominal filename to use when saving the effective indices. Defaults to 'wavelength_n_effs.dat'.
- **plot** (*bool*) – *True* if plots should be generates, otherwise *False*. Default is *True*.

Returns A list of the effective indices found for each wavelength.

Return type list

write_modes_to_file(*filename='mode.dat'*, *plot=True*, *analyse=True*)
Writes the mode fields to a file and optionally plots them.

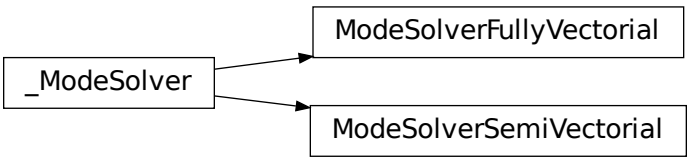
Parameters

- **filename** (*str*) – The nominal filename to use for the saved data. The suffix will be automatically be changed to identifiy each mode number. Default is 'mode.dat'
- **plot** (*bool*) – *True* if plots should be generates, otherwise *False*. Default is *True*.
- **analyse** (*bool*) – *True* if an analysis on the fundamental mode should be performed. The analysis adds to the plot of the fundamental mode the power mode-field diameter (MFD) and marks it on the output, and it marks with a cross the maximum E-field value. Default is *True*.

Returns A dictionary containing the effective indices and mode field profiles (if solved for).

Return type dict

2.1.3 Class Inheritance Diagram



2.2 Pre-defined Structures

2.2.1 Functions

use_gnuplot()	Use gnuplot as the plotting tool for any structure related outputs.
use_matplotlib()	Use matplotlib as the plotting tool for any structure related outputs.

use_gnuplot

modesolverpy.structure.**use_gnuplot**()
Use gnuplot as the plotting tool for any structure related outputs.

use_matplotlib

modesolverpy.structure.**use_matplotlib**()
Use matplotlib as the plotting tool for any structure related outputs.

2.2.2 Classes

RidgeWaveguide(wavelength, x_step, y_step, ...)	A general ridge waveguide structure.
WgArray(wavelength, x_step, y_step, ... [, ...])	

RidgeWaveguide

class modesolverpy.structure.**RidgeWaveguide**(*wavelength*, *x_step*, *y_step*, *wg_height*, *wg_width*, *sub_height*, *sub_width*, *clad_height*, *n_sub*, *n_wg*, *angle=0*, *n_clad=1.0*, *film_thickness='wg_height'*)

Bases: *modesolverpy.structure_base.Slabs*

A general ridge waveguide structure.

Parameters

- **wavelength** (*float*) – Wavelength the structure should operate at.
- **x_step** (*float*) – The grid step in x that the structure is created on.
- **y_step** (*float*) – The grid step in y that the structure is created on.
- **wg_height** (*float*) – The height of the ridge.
- **wg_width** (*float*) – The width of the ridge.
- **sub_height** (*float*) – The thickness of the substrate.
- **sub_width** (*float*) – The width of the substrate.
- **clad_height** (*float*) – The thickness of the cladding.
- **n_sub** (*float, function*) – Refractive index of the substrate. Either a constant (*float*), or a function that accepts one parameters, the wavelength, and returns a float of the refractive index. This is useful when doing wavelength sweeps and solving for the group velocity. The function provided could be a Sellmeier equation.
- **n_wg** (*float, function*) – Refractive index of the waveguide. Either a constant (*float*), or a function that accepts one parameters, the wavelength, and returns a float of the refractive index. This is useful when doing wavelength sweeps and solving for the group velocity. The function provided could be a Sellmeier equation.
- **angle** (*float*) – The angle of the sidewall [degrees] of the waveguide. Default is 0 degrees (vertical sidewalls).
- **n_clad** (*float, function*) – Refractive index of the cladding. Either a constant (*float*), or a function that accepts one parameters, the wavelength, and returns a float of the

refractive index. This is useful when doing wavelength sweeps and solving for the group velocity. The function provided could be a Sellmeier equation. Default is air.

- **film_thickness** (*float, str*) – The thickness of the film the waveguide is on. If the waveguide is a true ridge (fully etched), then the film thickness can be set to 'wg_height', otherwise the waveguide is a rib waveguide, and a float should be given specifying the thickness of the film.

Attributes Summary

eps	*np.array* – A grid of permittivies representing the permittivity profile of the structure.
eps_func	*function* – a function that when passed a *x* and *y* values, returns the permittivity profile of the structure, interpolating if necessary.
n	*np.array* – The refractive index profile matrix of the current slab.
n_func	*function* – a function that when passed a *x* and *y* values, returns the refractive index profile of the structure, interpolating if necessary.
x	*np.array* – The grid points in x.
x_ctr	*float* – The centre distance in x.
x_pts	*int* – The number of grid points in x.
xc	*np.array* – The centre points of the x points.
xc_max	*float* – The maximum value of *xc*.
xc_min	*float* – The minimum value of *xc*.
xc_pts	*int* – The number of points in *xc*.
y	*np.array* – The grid points in y.
y_ctr	*float* – The centre distance in y
y_pts	*int* – The number of grid points in y.
yc	*np.array* – The centre points of the y points.
yc_max	*float* – The maximum value of *yc*.
yc_min	*float* – The minimum value of *yc*.
yc_pts	*int* – The number of points in *yc*.

Methods Summary

add_slab(height[, n_background, position])	Creates and adds a `Slab` object.
change_wavelength(wavelength)	Changes the wavelength of the structure.
write_to_file([filename, plot])	Write the refractive index profile to file.

Attributes Documentation

eps

np.array – A grid of permittivies representing the permittivity profile of the structure.

eps_func

function – a function that when passed a *x* and *y* values, returns the permittivity profile of the structure, interpolating if necessary.

n

np.array – The refractive index profile matrix of the current slab.

n_func
function – a function that when passed a *x* and *y* values, returns the refractive index profile of the structure, interpolating if necessary.

x
np.array – The grid points in x.

x_ctr
float – The centre distance in x.

x_pts
int – The number of grid points in x.

xc
np.array – The centre points of the x points.

xc_max
float – The maximum value of *xc*.

xc_min
float – The minimum value of *xc*.

xc_pts
int – The number of points in *xc*.

y
np.array – The grid points in y.

y_ctr
float – The centre distance in y

y_pts
int – The number of grid points in y.

yc
np.array – The centre points of the y points.

yc_max
float – The maximum value of *yc*.

yc_min
float – The minimum value of *yc*.

yc_pts
int – The number of points in *yc*.

Methods Documentation

add_slab(*height*, *n_background=1.0*, *position='top'*)
Creates and adds a `Slab` object.

Parameters

- **height** (*float*) – Height of the slab.
- **n_background** (*float*) – The nominal refractive index of the slab. Default is 1 (air).

Returns The name of the slab.

Return type str

change_wavelength(*wavelength*)
Changes the wavelength of the structure.

This will affect the mode solver and potentially the refractive indices used (provided functions were provided as refractive indices).

Parameters **wavelength** (*float*) – The new wavelength.

write_to_file(*filename='material_index.dat'*, *plot=True*)
Write the refractive index profile to file.

Parameters

- **filename** (*str*) – The nominal filename the refractive index data should be saved to.
- **plot** (*bool*) – *True* if plots should be generates, otherwise *False*. Default is *True*.

WgArray

class modesolverpy.structure.**WgArray**(*wavelength*, *x_step*, *y_step*, *wg_height*, *wg_widths*, *wg_gaps*, *sub_height*, *sub_width*, *clad_height*, *n_sub*, *n_wg*, *angle=0*, *n_clad=1.0*)
Bases: *modesolverpy.structure_base.Slabs*

Attributes Summary

eps	*np.array* – A grid of permittivies representing the permittivity profile of the structure.
eps_func	*function* – a function that when passed a *x* and *y* values, returns the permittivity profile of the structure, interpolating if necessary.
n	*np.array* – The refractive index profile matrix of the current slab.
n_func	*function* – a function that when passed a *x* and *y* values, returns the refractive index profile of the structure, interpolating if necessary.
x	*np.array* – The grid points in x.
x_ctr	*float* – The centre distance in x.
x_pts	*int* – The number of grid points in x.
xc	*np.array* – The centre points of the x points.
xc_max	*float* – The maximum value of *xc*.
xc_min	*float* – The minimum value of *xc*.
xc_pts	*int* – The number of points in *xc*.
y	*np.array* – The grid points in y.
y_ctr	*float* – The centre distance in y
y_pts	*int* – The number of grid points in y.
yc	*np.array* – The centre points of the y points.
yc_max	*float* – The maximum value of *yc*.
yc_min	*float* – The minimum value of *yc*.
yc_pts	*int* – The number of points in *yc*.

Methods Summary

add_slab(height[, n_background, position])	Creates and adds a `Slab` object.
change_wavelength(wavelength)	Changes the wavelength of the structure.
write_to_file([filename, plot])	Write the refractive index profile to file.

Attributes Documentation

eps
np.array – A grid of permittivies representing the permittivity profile of the structure.

eps_func
function – a function that when passed a *x* and *y* values, returns the permittivity profile of the structure, interpolating if necessary.

n
np.array – The refractive index profile matrix of the current slab.

n_func
function – a function that when passed a *x* and *y* values, returns the refractive index profile of the structure, interpolating if necessary.

x
np.array – The grid points in x.

x_ctr
float – The centre distance in x.

x_pts
int – The number of grid points in x.

xc
np.array – The centre points of the x points.

xc_max
float – The maximum value of *xc*.

xc_min
float – The minimum value of *xc*.

xc_pts
int – The number of points in *xc*.

y
np.array – The grid points in y.

y_ctr
float – The centre distance in y

y_pts
int – The number of grid points in y.

yc
np.array – The centre points of the y points.

yc_max
float – The maximum value of *yc*.

yc_min
float – The minimum value of *yc*.

yc_pts
int – The number of points in *yc*.

Methods Documentation

add_slab(*height*, *n_background=1.0*, *position='top'*)
Creates and adds a `Slab` object.

Parameters

- **height** (*float*) – Height of the slab.
- **n_background** (*float*) – The nominal refractive index of the slab. Default is 1 (air).

Returns The name of the slab.

Return type str

change_wavelength(*wavelength*)
Changes the wavelength of the structure.

This will affect the mode solver and potentially the refractive indices used (provided functions were provided as refractive indices).

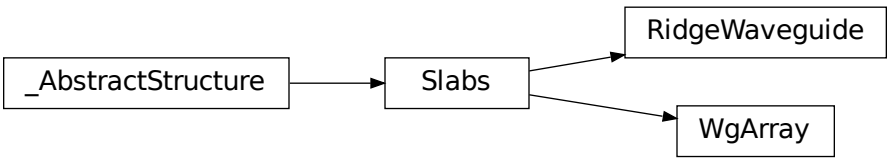
Parameters **wavelength** (*float*) – The new wavelength.

write_to_file(*filename='material_index.dat'*, *plot=True*)
Write the refractive index profile to file.

Parameters

- **filename** (*str*) – The nominal filename the refractive index data should be saved to.
- **plot** (*bool*) – *True* if plots should be generates, otherwise *False*. Default is *True*.

2.2.3 Class Inheritance Diagram



2.3 Structure Creation

2.3.1 Functions

use_gnuplot()	Use gnuplot as the plotting tool for any structure related outputs.
use_matplotlib()	Use matplotlib as the plotting tool for any structure related outputs.

Continued on next page

Table 11 – continued from previous page

with_metaclass(meta, *bases)	Create a base class with a metaclass.

use_gnuplot

modesolverpy.structure_base.**use_gnuplot**()
Use gnuplot as the plotting tool for any structure related outputs.

use_matplotlib

modesolverpy.structure_base.**use_matplotlib**()
Use matplotlib as the plotting tool for any structure related outputs.

2.3.2 Classes

Slab(name, x_step, y_step, x_max, y_max, ...)	A *Slab* represents a horizontal slice of the refractive index profile.
Slabs(wavelength, y_step, x_step, x_max[, x_min])	Class to implement device refractive index profile cross-section designs.
Structure(x_step, y_step, x_max, y_max[, ...])	
StructureAni(structure_xx, structure_yy, ...)	Anisottropic structure object.

Slab

class modesolverpy.structure_base.**Slab**(*name*, *x_step*, *y_step*, *x_max*, *y_max*, *x_min*, *y_min*, *n_background*, *wavelength*)
Bases: *modesolverpy.structure_base.Structure*

A *Slab* represents a horizontal slice of the refractive index profile.

A *Slabs* object composes many *Slab* objects. The more *Slab* are added, the more horizontal slices are added. A *Slab* has a chosen fixed height, and a background (nominal) refractive index. A slab can then be customised to include a desired design.

Parameters

- **name** (*str*) – The name of the slab.
- **x_step** (*float*) – The step in x.
- **y_step** (*float*) – The step in y.
- **x_max** (*float*) – The maximum x-value.
- **y_max** (*float*) – The maximum y-value.
- **x_min** (*float*) – The minimum x-value.
- **y_min** (*float*) – The minimum x-value.
- **n_background** (*float*) – The nominal refractive index.
- **wavelength** (*float*) – The wavelength the structure operates at.

name
str – The name of the *Slab* object.

position
int – A unique identifier for the

:class:`Slab` object.

Attributes Summary

eps	*np.array* – A grid of permittivies representing the permittivity profile of the structure.
eps_func	*function* – a function that when passed a *x* and *y* values, returns the permittivity profile of the structure, interpolating if necessary.
n	*np.array* – A grid of refractive indices representing the refractive index profile of the structure.
n_func	*function* – a function that when passed a *x* and *y* values, returns the refractive index profile of the structure, interpolating if necessary.
position	
x	*np.array* – The grid points in x.
x_ctr	*float* – The centre distance in x.
x_pts	*int* – The number of grid points in x.
xc	*np.array* – The centre points of the x points.
xc_max	*float* – The maximum value of *xc*.
xc_min	*float* – The minimum value of *xc*.
xc_pts	*int* – The number of points in *xc*.
y	*np.array* – The grid points in y.
y_ctr	*float* – The centre distance in y
y_pts	*int* – The number of grid points in y.
yc	*np.array* – The centre points of the y points.
yc_max	*float* – The maximum value of *yc*.
yc_min	*float* – The minimum value of *yc*.
yc_pts	*int* – The number of points in *yc*.

Methods Summary

add_material(x_min, x_max, n[, angle])	Add a refractive index between two x-points.
write_to_file([filename, plot])	Write the refractive index profile to file.

Attributes Documentation

eps
np.array – A grid of permittivies representing the permittivity profile of the structure.

eps_func
function – a function that when passed a *x* and *y* values, returns the permittivity profile of the structure, interpolating if necessary.

n
np.array – A grid of refractive indices representing the refractive index profile of the structure.

n_func

function – a function that when passed a *x* and *y* values, returns the refractive index profile of the structure, interpolating if necessary.

position = 0

x
np.array – The grid points in x.

x_ctr
float – The centre distance in x.

x_pts
int – The number of grid points in x.

xc
np.array – The centre points of the x points.

xc_max
float – The maximum value of *xc*.

xc_min
float – The minimum value of *xc*.

xc_pts
int – The number of points in *xc*.

y
np.array – The grid points in y.

y_ctr
float – The centre distance in y

y_pts
int – The number of grid points in y.

yc
np.array – The centre points of the y points.

yc_max
float – The maximum value of *yc*.

yc_min
float – The minimum value of *yc*.

yc_pts
int – The number of points in *yc*.

Methods Documentation

add_material(*x_min*, *x_max*, *n*, *angle=0*)
Add a refractive index between two x-points.

Parameters

- **x_min** (*float*) – The start x-point.
- **x_max** (*float*) – The stop x-point.
- **n** (*float, function*) – Refractive index between *x_min* and *x_max*. Either a constant (*float*), or a function that accepts one parameters, the wavelength, and returns a float of the refractive index. This is useful when doing wavelength sweeps and solving for the group velocity. The function provided could be a Sellmeier equation.

- **angle** (*float*) – Angle in degrees of the slope of the sidewalls at *x_min* and *x_max*. This is useful for defining a ridge with angled sidewalls.

write_to_file(*filename='material_index.dat'*, *plot=True*)
Write the refractive index profile to file.

Parameters

- **filename** (*str*) – The nominal filename the refractive index data should be saved to.
- **plot** (*bool*) – *True* if plots should be generates, otherwise *False*. Default is *True*.

Slabs

class modesolverpy.structure_base.**Slabs**(*wavelength*, *y_step*, *x_step*, *x_max*, *x_min=0.0*)
Bases: modesolverpy.structure_base._AbstractStructure

Class to implement device refractive index profile cross-section designs.

Slabs is a collection of *Slab* objects. Each slab has a fixed height (usually less than the maximum height of the desired simulation window), and is as wide as the simulation window.

Slabs objects can be index using *[name]* to return the various *Slab* objects. The bottom slab is returned first and so on up to the top slab.

Parameters

- **wavelength** (*float*) – The wavelength the structure operates at.
- **y_step** (*float*) – The step in y.
- **x_step** (*float*) – The step in x.
- **x_max** (*float*) – The maximum x-value.
- **x_min** (*float*) – The minimum x-value. Default is 0.

slabs
dict – The key is the name of the slab, and the value is the *Slab* object.

slab_count
int – The number of *Slab* objects added so far.

Attributes Summary

eps	*np.array* – A grid of permittivies representing the permittivity profile of the structure.
eps_func	*function* – a function that when passed a *x* and *y* values, returns the permittivity profile of the structure, interpolating if necessary.
n	*np.array* – The refractive index profile matrix of the current slab.
n_func	*function* – a function that when passed a *x* and *y* values, returns the refractive index profile of the structure, interpolating if necessary.
x	*np.array* – The grid points in x.
x_ctr	*float* – The centre distance in x.
x_pts	*int* – The number of grid points in x.

Continued on next page

Table 15 – continued from previous page

xc	*np.array* – The centre points of the x points.
xc_max	*float* – The maximum value of *xc*.
xc_min	*float* – The minimum value of *xc*.
xc_pts	*int* – The number of points in *xc*.
y	*np.array* – The grid points in y.
y_ctr	*float* – The centre distance in y
y_pts	*int* – The number of grid points in y.
yc	*np.array* – The centre points of the y points.
yc_max	*float* – The maximum value of *yc*.
yc_min	*float* – The minimum value of *yc*.
yc_pts	*int* – The number of points in *yc*.

Methods Summary

add_slab(height[, n_background, position])	Creates and adds a *Slab* object.
change_wavelength(wavelength)	Changes the wavelength of the structure.
write_to_file([filename, plot])	Write the refractive index profile to file.

Attributes Documentation

eps
np.array – A grid of permittivies representing the permittivity profile of the structure.

eps_func
function – a function that when passed a *x* and *y* values, returns the permittivity profile of the structure, interpolating if necessary.

n
np.array – The refractive index profile matrix of the current slab.

n_func
function – a function that when passed a *x* and *y* values, returns the refractive index profile of the structure, interpolating if necessary.

x
np.array – The grid points in x.

x_ctr
float – The centre distance in x.

x_pts
int – The number of grid points in x.

xc
np.array – The centre points of the x points.

xc_max
float – The maximum value of *xc*.

xc_min
float – The minimum value of *xc*.

xc_pts
int – The number of points in *xc*.

y
np.array – The grid points in y.

y_ctr
float – The centre distance in y

y_pts
int – The number of grid points in y.

yc
np.array – The centre points of the y points.

yc_max
float – The maximum value of *yc*.

yc_min
float – The minimum value of *yc*.

yc_pts
int – The number of points in *yc*.

Methods Documentation

add_slab(*height*, *n_background=1.0*, *position='top'*)
Creates and adds a `Slab` object.

Parameters

- **height** (`float`) – Height of the slab.
- **n_background** (`float`) – The nominal refractive index of the slab. Default is 1 (air).

Returns The name of the slab.

Return type str

change_wavelength(*wavelength*)
Changes the wavelength of the structure.

This will affect the mode solver and potentially the refractive indices used (provided functions were provided as refractive indices).

Parameters **wavelength** (`float`) – The new wavelength.

write_to_file(*filename='material_index.dat'*, *plot=True*)
Write the refractive index profile to file.

Parameters

- **filename** (`str`) – The nominal filename the refractive index data should be saved to.
- **plot** (`bool`) – *True* if plots should be generates, otherwise *False*. Default is *True*.

Structure

class modesolverpy.structure_base.**Structure**(*x_step*, *y_step*, *x_max*, *y_max*, *x_min=0.0*, *y_min=0.0*, *n_background=1.0*)
Bases: `modesolverpy.structure_base._AbstractStructure`

Attributes Summary

eps	*np.array* – A grid of permittivies representing the permittivity profile of the structure.
eps_func	*function* – a function that when passed a *x* and *y* values, returns the permittivity profile of the structure, interpolating if necessary.
n	*np.array* – A grid of refractive indices representing the refractive index profile of the structure.
n_func	*function* – a function that when passed a *x* and *y* values, returns the refractive index profile of the structure, interpolating if necessary.
x	*np.array* – The grid points in x.
x_ctr	*float* – The centre distance in x.
x_pts	*int* – The number of grid points in x.
xc	*np.array* – The centre points of the x points.
xc_max	*float* – The maximum value of *xc*.
xc_min	*float* – The minimum value of *xc*.
xc_pts	*int* – The number of points in *xc*.
y	*np.array* – The grid points in y.
y_ctr	*float* – The centre distance in y
y_pts	*int* – The number of grid points in y.
yc	*np.array* – The centre points of the y points.
yc_max	*float* – The maximum value of *yc*.
yc_min	*float* – The minimum value of *yc*.
yc_pts	*int* – The number of points in *yc*.

Methods Summary

write_to_file([filename, plot])	Write the refractive index profile to file.

Attributes Documentation

eps
np.array – A grid of permittivies representing the permittivity profile of the structure.

eps_func
function – a function that when passed a *x* and *y* values, returns the permittivity profile of the structure, interpolating if necessary.

n
np.array – A grid of refractive indices representing the refractive index profile of the structure.

n_func
function – a function that when passed a *x* and *y* values, returns the refractive index profile of the structure, interpolating if necessary.

x
np.array – The grid points in x.

x_ctr
float – The centre distance in x.

x_pts
int – The number of grid points in x.

xc
np.array – The centre points of the x points.

xc_max
float – The maximum value of *xc*.

xc_min
float – The minimum value of *xc*.

xc_pts
int – The number of points in *xc*.

y
np.array – The grid points in y.

y_ctr
float – The centre distance in y

y_pts
int – The number of grid points in y.

yc
np.array – The centre points of the y points.

yc_max
float – The maximum value of *yc*.

yc_min
float – The minimum value of *yc*.

yc_pts
int – The number of points in *yc*.

Methods Documentation

write_to_file(*filename='material_index.dat'*, *plot=True*)
Write the refractive index profile to file.

Parameters

- **filename** (*str*) – The nominal filename the refractive index data should be saved to.
- **plot** (*bool*) – *True* if plots should be generates, otherwise *False*. Default is *True*.

StructureAni

class modesolverpy.structure_base.**StructureAni**(*structure_xx*, *structure_yy*, *structure_zz*, *structure_xy=None*, *structure_yx=None*)

Bases: object

Anisottropic structure object.

This is used with the fully-vectorial simulation when an anisotropic material is being used.

The form of the refractive index is

$$n = \begin{bmatrix} n_{xx} & n_{xy} & 0 \\ n_{yx} & n_{yy} & 0 \\ 0 & 0 & n_{zz} \end{bmatrix}.$$

Parameters

- **structure_xx** (Structure) – The structure with refractive index, n_{xx}.
- **structure_yy** (Structure) – The structure with refractive index, n_{yy}. Presumably the same structure as *structure_xx*, but with different refractive index parameters.
- **structure_zz** (Structure) – The structure with refractive index, n_{zz}. Presumably the same structure as *structure_xx*, but with different refractive index parameters.
- **structure_xy** (*None*, Structure) – The structure with refractive index, n_{yx}. Presumably the same structure as *structure_xx*, but with different refractive index parameters. Default is *None*.
- **structure_yx** (*None*, Structure) – The structure with refractive index, n_{yx}. Presumably the same structure as *structure_xx*, but with different refractive index parameters. Default is *None*.

Attributes Summary

eps
eps_func
n
n_func
x
x_ctr
x_pts
x_step
xc
xc_max
xc_min
xc_pts
y
y_ctr
y_pts
y_step
yc
yc_max
yc_min
yc_pts

Methods Summary

change_wavelength(wavelength)	Changes the wavelength of the structure.
write_to_file([filename, plot])	Write the refractive index profile to file.

Attributes Documentation

eps

eps_func

n

n_func

x

x_ctr

x_pts

x_step

xc

xc_max

xc_min

xc_pts

y

y_ctr

y_pts

y_step

yc

yc_max

yc_min

yc_pts

Methods Documentation

change_wavelength(*wavelength*)
Changes the wavelength of the structure.

This will affect the mode solver and potentially the refractive indices used (provided functions were provided as refractive indices).

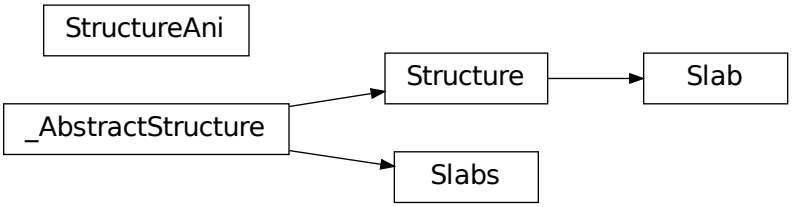
Parameters **wavelength** (*float*) – The new wavelength.

write_to_file(*filename='material_index.dat'*, *plot=True*)
Write the refractive index profile to file.

Parameters

- **filename** (*str*) – The nominal filename the refractive index data should be saved to.
- **plot** (*bool*) – *True* if plots should be generates, otherwise *False*. Default is *True*.

2.3.3 Class Inheritance Diagram



2.4 Design Tools

2.4.1 Functions

directional_coupler_lc(wavelength_nm, ...)	Calculates the coherence length (100% power transfer) of a directional coupler.
grating_coupler_period(wavelength, n_eff, ...)	Calculate the period needed for a grating coupler.
loss(n, wavelength)	
qpm_period(pmp_n, pmp_l, sig_n, sig_l, ...)	
qpm_wavenumber(pmp_n, pmp_l, sig_n, sig_l, ...)	

directional_coupler_lc

modesolverpy.design.**directional_coupler_lc**(*wavelength_nm*, *n_eff_1*, *n_eff_2*)

Calculates the coherence length (100% power transfer) of a directional coupler.

Parameters

- **wavelength_nm** (*float*) – The wavelength in [nm] the directional coupler should operate at.
- **n_eff_1** (*float*) – n_eff of the fundamental (even) supermode of the directional coupler.
- **n_eff_2** (*float*) – n_eff of the first-order (odd) supermode of the directional coupler.

Returns The length [um] the directional coupler needs to be to achieve 100% power transfer.

Return type float

grating_coupler_period

modesolverpy.design.**grating_coupler_period**(*wavelength*, *n_eff*, *n_clad*, *incidence_angle_deg*, *diffration_order=1*)

Calculate the period needed for a grating coupler.

Parameters

- **wavelength** (*float*) – The target wavelength for the grating coupler.
- **n_eff** (*float*) – The effective index of the mode of a waveguide with the width of the grating coupler.
- **n_clad** (*float*) – The refractive index of the cladding.
- **incidence_angle_deg** (*float*) – The incidence angle the grating coupler should operate at [degrees].
- **diffration_order** (*int*) – The grating order the coupler should work at. Default is 1st order (1).

Returns The period needed for the grating coupler in the same units as the wavelength was given at.

Return type float

loss

modesolverpy.design.**loss**(*n*, *wavelength*)

qpm_period

modesolverpy.design.**qpm_period**(*pmp_n*, *pmp_l*, *sig_n*, *sig_l*, *idl_n*, *idl_l*, *type='forward'*)

qpm_wavenumber

modesolverpy.design.**qpm_wavenumber**(*pmp_n*, *pmp_l*, *sig_n*, *sig_l*, *idl_n*, *idl_l*, *period_qpm*, *type='forward'*)

2.5 Coupling Efficiency

2.5.1 Functions

coupling_efficiency(mode_solver, fibre_mfd)	Finds the coupling efficiency between a solved fundamental mode and a fibre of given MFD.
reflection(n1, n2)	Calculate the power reflection at the interface of two refractive index materials.
transmission(n1, n2)	Calculate the power transmission at the interface of two refractive index materials.

coupling_efficiency

modesolverpy.coupling_efficiency.**coupling_efficiency**(*mode_solver*, *fibre_mfd*, *fibre_offset_x=0*, *fibre_offset_y=0*, *n_eff_fibre=1.441*)

Finds the coupling efficiency between a solved fundamental mode and a fibre of given MFD.

Parameters

- **mode_solver** (*_ModeSolver*) – Mode solver that has found a fundamental mode.
- **fibre_mfd** (*float*) – The mode-field diameter (MFD) of the fibre.

- **fibre_offset_x** (*float*) – Offset the fibre from the centre position of the window in x. Default is 0 (no offset).
- **fibre_offset_y** (*float*) – Offset the fibre from the centre position of the window in y. Default is 0 (no offset).
- **n_eff_fibre** (*float*) – The effective index of the fibre mode. Default is 1.441.

Returns The power coupling efficiency.

Return type float

reflection

modesolverpy.coupling_efficiency.**reflection**(*n1*, *n2*)

Calculate the power reflection at the interface of two refractive index materials.

Parameters

- **n1** (*float*) – Refractive index of material 1.
- **n2** (*float*) – Refractive index of material 2.

Returns The percentage of reflected power.

Return type float

transmission

modesolverpy.coupling_efficiency.**transmission**(*n1*, *n2*)

Calculate the power transmission at the interface of two refractive index materials.

Parameters

- **n1** (*float*) – Refractive index of material 1.
- **n2** (*float*) – Refractive index of material 2.

Returns The percentage of transmitted power.

Return type float

Python Module Index

m

Index

P

Q

R

S

T

U

W

X

Y