Sphinx format for Latex and HTML

Meher Krishna Patel

Created on : Octorber, 2017

Last updated : December, 2022

Table of contents

Table of contents	**i**
List of figures	**iii**
List of tables	**v**
1 Waveguide	**1**
2 Pump	**5**
3 Experiment	**7**
Python Module Index	**11**
Python Module Index	**11**

List of figures

List of tables

Chapter 1

Waveguide

The Waveguide package, containing tools for manipulating and and constructing waveguide geometry and physical data.

`class pyjsa.waveguide.Waveguide(`*film_thickness_um: float, width_idx: int, height_idx: int, length: float, wg_angle=60, cladding='air', profile=0, custom_poling_profile=None, custom_nonlinearity_profile=None*`)`

Implements a Waveguide object, which stores information relevant for the LNOI waveguide construction and material properties. The properties of the waveguide like the dispersion relation are stored externally in .npz files in the folder "..data".

`GAUSSIAN_POLED`

Attribute for specifying that the waveguide has a Gaussian poling profile.

Type
int

`GAUSSIAN`

Attribute for specifying that the waveguide has a Gaussian nonlinearity profile.

Type
int

`REGULAR_POLED`

Attribute for specifying that the waveguide has a regular poling profile.

Type
int

`CUSTOM_POLED`

Attribute for specifying that the waveguide has a custom poling profile.

Type
int

`CUSTOM`

Attribute for specifying that the waveguide has a custom nonlinearity profile.

Type
int

Parameters

- `film_thickness_um` (*float*) – Thickness of the LN film used in the fabrication of the waveguide. For the default dataset, values can only be elements of the array [0.3:0.1:0.8] (micrometers).
- `width_idx` (*int*) – The width of the waveguide. It specifies the index of the element in the (default) array [0.5:0.1:2.0] (micrometers).

- `height_idx` (*int*) – The height of the waveguide. It specifies the index of the element in the (default) array [0.2:0.1:film_thickness] (micrometers).
- `length` (*float*) – The length of the waveguide (meters).
- `wg_angle` (*float, Optional*) – The angle of the waveguide ridge (degrees). For the default dataset, only the default value of 60 is possible.
- `cladding` (*string, Optional*) – The cladding of the waveguide. For the default dataset, only the default value of "air" is possible.
- `profile` (*int, Optional*) – An integer specifying the poling of the waveguide taken from the attributes of the class mentioned above, by default 0.
- `custom_poling_profile` (*numpy.ndaray, Optional*) – A custom poling profile to be used if the waveguide is specified to be CUSTOM_POLED, by default None.
- `custom_nonlinearity_profile` (*callable, Optional*) – A custom nonlinearity profile to be used if the waveguide is specified to be CUSTOM, by default None.

`property custom__poling_profile`

A custom poling profile for the waveguide.

Type
np.ndarray

`property custom_nonlinearity_profile`

A custom nonlinearity profile for the waveguide.

Type
callable

`property custom_poling_profile`

A custom poling profile for the waveguide.

Type
np.ndarray

`g(`*poling_period=None*`)`

A function that returns the poling profile or nonlinearity profile of the waveguide, depending on the profile attribute of the class.

Parameters
`poling_period` (*float, optional*) – The poling period of the waveguide (meters), by default None.

Returns
The poling profile or nonlinearity profile of the waveguide.

Return type
np.ndarray or callable

`property length`

Length of the waveguide.

Type
float

`property neff_TE`

Dispersion relation for TE polarization, of the form neff_TE(wavelength in micrometers).

Type
callable

`property neff_TM`

Dispersion relation for TM polarization, of the form neff_TM(wavelength in micrometers).

Type
callable

`property profile`

Profile of the waveguide.

Type
int

`pyjsa.waveguide.find_poling_profile(`*length*, *poling_period*, *target_pmf*, **args*`)`

An implementation of the deleted domain algorithm described in Chapter 2.

Parameters
- `length` (*float*) – The length of the waveguide (meters).
- `poling_period` (*float*) – The poling period of the waveguide (meters).
- `target_pmf` (*callable*) – A function that returns the target PMF at a position along the waveguide.
- `args` (*iterable*) – Arguments to be passed to target_pmf

Returns
The poling profile that minimizes the error between the target PMF and the resulting PMF.

Return type
np.ndarray

`pyjsa.waveguide.pmf_gaussian(`z, L`)`

A function to compute the PMF for a waveguide with a Gaussian nonlinearity profile at a fixed phase mismatch

Parameters
- `z` (*float*) – position along the waveguide
- `L` (*float*) – length of the waveguide

Returns
PMF at the specified position

Return type
float

Chapter 2

Pump

The Pump package contains only the class Pump, used to compute and store the properties of the pump laser.

`class pyjsa.pump.Pump(`*width*, *points=1000*`)`

A class for storing and computing the properties of the pump laser.

Parameters

- `width` (*float*) – Width of the pump spectrum at half maximum (meters).
- `points` (*int, Optional*) – The number of points into which to divide the λ_i and λ_s axis in order to compute the PEF, by default 1000.

`property center`

The center of the pump spectrum, inferred from the signal and the idler by energy conservation (meters). Cannot be set.

Type

float

`hermite_mode(`*x: float*`)`

A normalised Hermite-Gaussian function in temporal mode 0:

$$\frac{e^{-(x_0-x)^2/(2w^2)}}{\sqrt{\sqrt{\pi}w}} H_0(x_0 - x)$$

where x_0 is the center and w is the width.

Parameters

x (*float*) – The wavelength (meters).

Return type

float

`property idler`

An array with two elements that defines the idler. The first element is the central wavelength and the second is the width of the window (meters).

Type

numpy.ndarray

`property points`

The number of points into which to divide the λ_i and λ_s axis.

Type

int

`pump_envelope_function()`

A function that computes the PEF.

Returns
The PEF as a matrix array with dimesnions (points, points).

Return type
np.ndarray

`property signal`

An array with two elements that defines the signal. The first element is the central wavelength and the second is the width of the window (meters).

Type
numpy.ndarray

`signal_idler_ranges()`

A function that returns the signal and idler ranges over which to compute the PEF.

Returns
A tuple where the first elements is the signal range and the second is idler range.

Return type
tuple

`property width`

The width of the pump spectrum at half maximum (meters).

Type
float

Chapter 3

Experiment

This module is the core of the package. It contains classes and functions that aid in the computation of the PMF, JSA, Schmidt decompostion and in the optimization of the output photon spectral purity.

`class pyjsa.experiment.Experiment(`*waveguide: Waveguide*, *pump: Pump*, *signal*, *idler*, *SPDC_type=0*`)`

The experiment class incorporates the tools for computing the PMF, JSA, Schmidt decomposition and heralding efficiencies.

Parameters

- `waveguide` (`pyjsa.waveguide.Waveguide`) – A waveguide object for the experiment.
- `pump` (`pyjsa.pump.Pump`) – A pump object for the experiment.
- `signal` (*tuple*) – A tuple where the first element is the wavelength of the signal and the second is the width of the window (meters).
- `idler` (*tuple*) – A tuple where the first element is the wavelength of the idler and the second is the width of the window (meters).
- `SPDC_type` (*int, Optional*) – The type of the SPDC process. Can only be 0, 1 or 2, by default 0.

`property SPDC_type`

The SPDC type.

Type

int

`property delta_k`

The phase mismatch along the $\lambda_i - \lambda_s$ plane. Cannot be set.

Type

numpy.ndarray

`joint_spectral_amplitude(`*filter_signal_width*, *filter_idler_width*, *filter_signal_center*, *filter_idler_center*`)`

A function that computes the JSA for the specified waveguide, pump and signal and idler ranges combination, incorporating filtering, For no filtering, you should set the center and width of the filters to those of the signal and idler

Parameters

- `filter_signal_width` (*float*) – The width of the filter on the signal.
- `filter_idler_width` (*float*) – The width of the filter on the idelr
- `filter_signal_center` (*float*) – The center of the filter on the signal.
- `filter_idler_center` (*float*) – The center of the filter on the idler.

Returns

- *numpy.ndarray* – The JSA.
- *float* – Probability that the signal passes its filter.
- *float* – Probability that the idler passes its filter.
- *float* – Probability that both the signal and the idler pass their filters.

`phase_matching_function`(*points=10000*)

A function that computed the PMF for the specified signal, idler ranges and waveguide poling profile.

Parameters

`points` (*int , optional*) – The number of points into which to discretize the length of the waveguide if its profile is set to CUSTOM, by default 10000

Returns

The PMF.

Return type

numpy.ndarray

`property pmf`

The PMF. If it is None, it will be computed.

Type

numpy.ndarray

`property poling_period`

The poling period for the specified process, signal and idler combination. Cannot be set.

Type

float

`property pump`

The pump in the experiment.

Type

pyjsa.pump.Pump

`schmidt_decomposition`(*filter_signal_width*, *filter_idler_width*, *filter_signal_center*, *filter_idler_center*)

A function that computes the Schmidt decomposition of the JSA.

Parameters

- `filter_signal_width` (*float*) – The width of the filter on the signal.
- `filter_idler_width` (*float*) – The width of the filter on the idelr
- `filter_signal_center` (*float*) – The center of the filter on the signal.
- `filter_idler_center` (*float*) – The center of the filter on the idler.

Returns

- *numpy.ndarray* – An array of the normalized singular values of the JSA.
- *float* – The spectral purity of the output photons.

`property waveguide`

The waveguide in the experiment.

Type

pyjsa.waveguide.Waveguide

`pyjsa.experiment.find_optimal_pump_width`(*exp*: Experiment, *width_bounds*, *filter_signal_width*, *filter_idler_width*, *filter_signal_center*, *filter_idler_center*)

A function that optimizes the width of the pump of an experiment such that it yields the highest spectral purity. It uses the default optimizer of scipy.optimize.minimize_scalar.

Parameters

- `exp` (`Experiment`) – The experiment whose pump width is to be optimized.
- `width_bounds` (*numpy.array , tuple*) – An array with two elements that specifies the bounds of the optimization, ex. [0.1, 10] (nanometers).
- `filter_signal_width` (*float*) – The width of the filter on the signal.
- `filter_idler_width` (*float*) – The width of the filter on the idelr
- `filter_signal_center` (*float*) – The center of the filter on the signal.
- `filter_idler_center` (*float*) – The center of the filter on the idler.

Returns

The result of the optimization routine.

Return type
scipy.optimize.OptimizeResult

`pyjsa.experiment.rect`(*x*, *center*, *width*)

A rectangular window function.

Parameters

- `x` (*float*) – The postion at which to evaluate the window function.
- `center` (*float*) – The center of the window function.
- `width` (*float*) – The Width of the window function.

Returns
The value of the window function at position x.

Return type
float

Python Module Index

p

pyjsa.experiment, 7
pyjsa.pump, 5
pyjsa.waveguide, 1