Silicon Photonics Toolkit

Getting Started

Silicon Photonics Toolkit is a toolkit providing fundamental waveguide and material properties to aid in the design of silicon photonic components on SOI platforms with high accuracy and extremely fast runtime. See the documentation for tutorials and API reference.

Installation

Pip

The package can be installed via pip:

```
pip install sipkit
```

You can install siphotonics with additional packages for developers:

```
pip install sipkit[dev]
```

Build from source

Alternatively, the package can be built from source by cloning the repository and running the setup script:

```
git clone https://github.com/Photonic-Architecture-Laboratories/si-photonics-toolkit.git
cd si-photonics-toolkit
pip install -e .
```

Dependencies

The package requires the following packages to be installed:

- NumPy
- Jax

Citing SiPhotonics Toolkit

```
@software{silicon-photonics-toolkit2022github,
```

Effective Index

Effective Index of the Fundamental Mode

Effective index of the fundamental mode of a waveguide can be accessed by using `neff` function of `effective_index` module. Note that for the default parameters below, this mode happens to be a TE mode. The function takes the following arguments:

- `wavelength`: Wavelength of the light in the waveguide, in micrometers. A list of wavelengths can also be passed. Wavelength range is from 1.2 µm to 1.7 µm; it will return 0 for wavelengths out of range.
- `width`: Width of the waveguide, in micrometers. A list of widths can also be passed. Width range is from 0.240 µm to 0.700 µm; it will return 0 for widths out of range.

In [1]:
```
from sipkit.effective_index import neff
import jax.numpy as jnp

neff(width=0.5, wavelength=1.55)
```

```
No GPU/TPU found, falling back to CPU. (Set TF_CPP_MIN_LOG_LEVEL=0 and rerun
for more info.)
```

Out[1]:
```
Array(2.44523381, dtype=float64)
```

Use Cases

- Scalar width, scalar wavelength

In [2]:
```
neff(width=0.5, wavelength=1.55)
```

Out[2]:
```
Array(2.44523381, dtype=float64)
```

- Scalar width, 1D Array wavelength

In [3]:
```
neff(width=0.5, wavelength=jnp.linspace(1.4, 1.5, 5))
```

Out[3…
```
Array([2.6138077 , 2.58583721, 2.55782222, 2.52975709, 2.50163816],
      dtype=float64)
```

Group Index

Getting Started

This function returns group index of the fundamental mode for given waveguide width and wavelength.

Group Index is calculated using the following equation where n_eff is the effective index, n_g is the group index, w is the waveguide width, and λ is the wavelength.

$$n_g(w, \lambda) = n_\mathrm{eff}(w, \lambda) - \lambda \cdot \frac{\partial n_\mathrm{eff}(w, \lambda)}{\partial \lambda}$$
This is what is implemented in the built-in function `ng`.

Note: `ng` function implementation is based on finite difference method. Therefore, the result is not exact. However, the result is accurate enough for most of the cases. If you need exact result, please use `grad_neff` function to calculate the derivative of `neff` function.

In [1]:
```
import sipkit
import matplotlib.pyplot as plt
from jax import numpy as jnp

%matplotlib inline

wavelengths = jnp.linspace(1.5, 1.6, 1000)
group_index_400 = jnp.array([sipkit.ng(0.4, wl) for wl in wavelengths])
group_index_450 = jnp.array([sipkit.ng(0.45, wl) for wl in wavelengths])
group_index_500 = jnp.array([sipkit.ng(0.5, wl) for wl in wavelengths])
group_index_550 = jnp.array([sipkit.ng(0.55, wl) for wl in wavelengths])
group_index_600 = jnp.array([sipkit.ng(0.6, wl) for wl in wavelengths])
```

```
No GPU/TPU found, falling back to CPU. (Set TF_CPP_MIN_LOG_LEVEL=0 and rerun
for more info.)
```

In [2]:
```
plt.plot(wavelengths, group_index_400, label='400 nm')
plt.plot(wavelengths, group_index_450, label='450 nm')
plt.plot(wavelengths, group_index_500, label='500 nm')
plt.plot(wavelengths, group_index_550, label='550 nm')
plt.plot(wavelengths, group_index_600, label='600 nm')
plt.legend()
plt.xlabel('Wavelength (µm)')
plt.ylabel('Group index')
plt.title('Group index of a SOI strip waveguide (SiO2/Si)')
plt.ylim(4.0, 4.4)
```

Dielectric Constant of Silicon and Silicon Dioxide

Getting Started

Dielectric constant (relative permittivity) of Si or SiO2.

- It takes **1 argument** which is **wavelength** in micrometers.
- Similar to effective index function, wavelength range is between 1.2 µm and 1.7 µm.

Arguments:

- **wavelength** (float): wavelength in micrometers, range 1.2 - 1.7

Returns:

- **permittivity** (float): permittivity of specified material

Dielectric constant of silicon at a wavelength of 1.55 µm:

In [1]:
```
import sipkit

print(sipkit.perm_si(1.55))
```

```
No GPU/TPU found, falling back to CPU. (Set TF_CPP_MIN_LOG_LEVEL=0 and rerun
for more info.)
12.085653911764705
```

Dielectric constant of silicon dioxide at a wavelength of 1.55 µm:

In [2]:
```
print(sipkit.perm_oxide(1.55))
```

```
2.085216543284305
```

Exceptions

Unless the first argument is between 1.2 µm and 1.7 µm, it will raise an exception:

API Reference

`sipkit.effective_index`

`sipkit.effective_index.neff(width, wavelength)`

Gets Effective Index value by using corresponding parameters. This is a JAX compatible function. JIT is enabled.

Parameters:

Name	Type	Description	Default
`width`	`float`	Waveguide width in microns. (0.24 - 0.7) Scalar or list	*required*
`wavelength`	`float`	Wavelength in microns. (1.2 - 1.7) Scalar or list	*required*

Returns:

Type	Description
`float | list[float]`	Effective Index value(s).

Examples:

```
>>> neff(0.5, 1.5)
array(2.)
```

```
>>> neff(0.5, [1.5, 1.6])
array([2. , 2.1])
```

```
>>> neff([0.5, 0.6], 1.5)
array([2. , 1.9])
```

```
>>> neff([0.5, 0.6], [1.5, 1.6])
array([[2. , 2.1],
```

```
>>> neff(0.5, [[1.5, 1.6], [1.7, 1.8]])
array([[2. , 2.1],
       [1.9, 2. ]])
```