Inverse Design of Nanophotonic Devices using SPINS-B

Aditya K. Rao
Final-Year UG: Department of Electrical Engineering

Instructor-in-Charge:
Dr. Shilpi Gupta
Department of Electrical Engineering

About SPINS

- The SPINS software is a gradient based optimisation software that supports 2D and 3D optimisation using Finite Difference Frequency Domain (FDFD) method.
- SPINS-B is an open source version of SPINS which has been used for this project.
- The library provides an effective method to create photonic architecture for various purpose using machine learning.

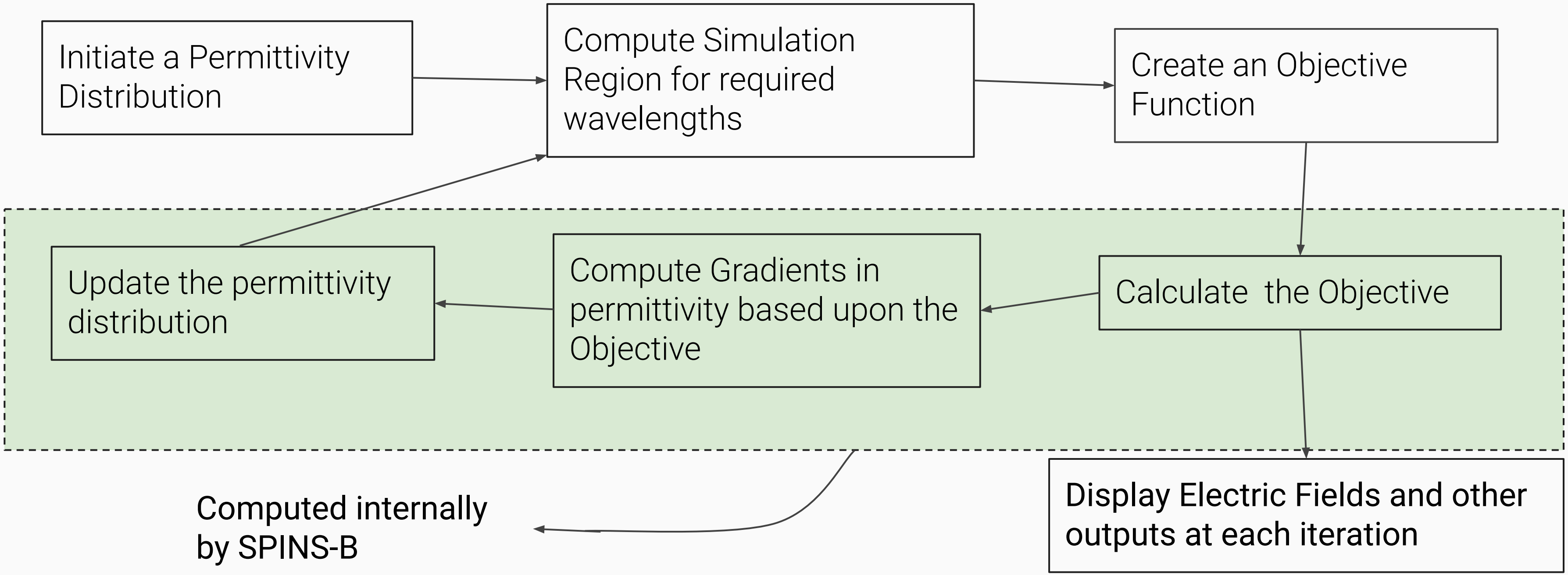
Some General Concepts

- **Optimisation Plan:** A sequential arrangement of Nodes and Action that help create a flowchart of iteratively solving a problem.
- **Nodes:** Variables that store data like refractive index, permittivity distribution, electric field distribution, gradients, etc.
- **Shape Nodes:** The library has inbuilt architecture commonly used in nanophotonics like waveguides, cylindrical/rectangular cavities, etc.

Some General Concepts

- **Source:** The library also has EM sources like gaussian pulse, plane waves, waveguide mode, waveguide mode impulse etc.
- **Simulation Node:** A well defined region where fields are calculated based upon FDFD Maxwell Equation Solvers. We can control the minimum feature shape and dimensions of the simulation region.
- **Output Specification.**
- The primary objective is to create a problem graph and optimisation plan for simulation to run.

Problem Graph



Mode Overlap Integral

We define mode overlap integral as following where E2 is reference complex valued electric field and E1 is the given electric field. The overlap matching gives a measure of how efficiently E1 is mimicking E2.

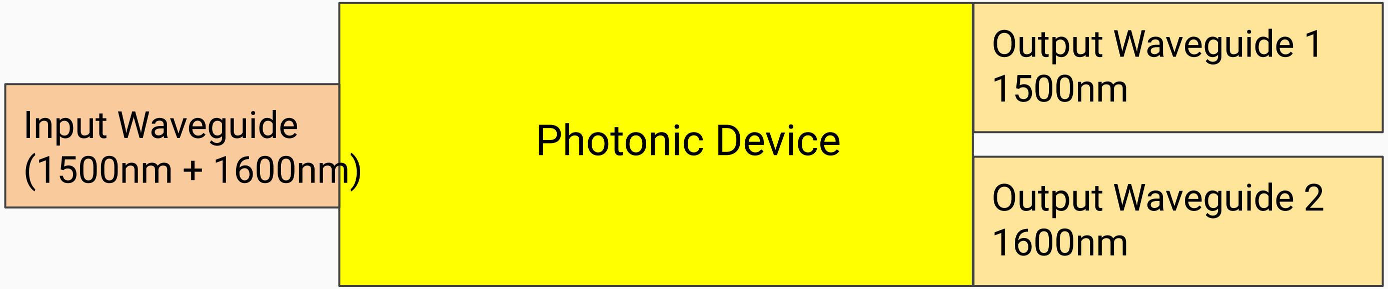
$$\eta = \frac{\int_S E_1^* E_2 dS}{\int_S |E_2|^2 dS}$$

$$E_1 = E_2 \Rightarrow |\eta| = 1$$

If E1 is phase shifted, the absolute overlap will still remain 1. In such a scenario, we can say that power emitted is same, without the field profiles being exactly similar.

Wavelength Demultiplexer

The aim was to create a Photonic Demultiplexer that uncouples two wavelengths (1500nm and 1600nm) present in a waveguide and gives out both wavelengths in two different waveguide as output.

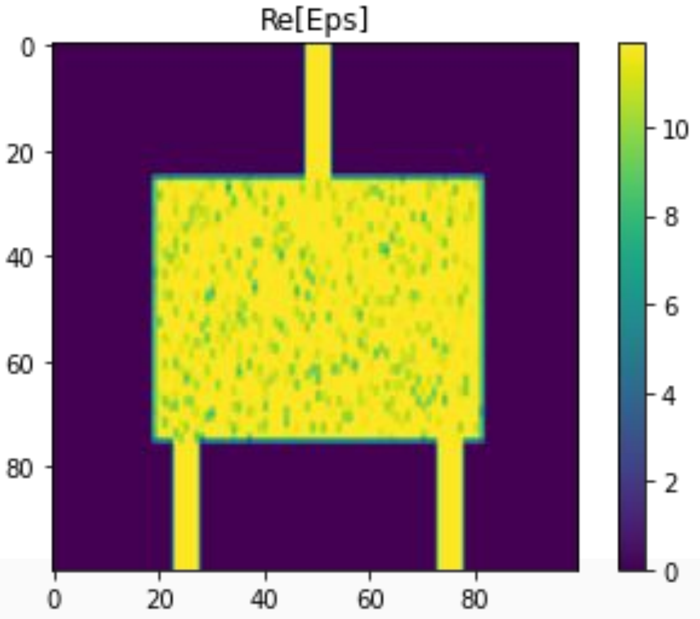


Wavelength Demultiplexer

We structured 3 Square Waveguides each of cross-section (200nmX200nm) placed as shown in the figure.

The permittivity distribution was initialised with majority of the device having n=3.45 and small pockets of n>1.0

Each pixel shape was 40nmX40nm as the minimum feature shape keeping in mind the physical manufacturing constraint.



```python
# Define simulation space.
my_simulation_space = maxwell.SimulationSpace(
          mesh=maxwell.UniformMesh(dx=40),
          sim_region=goos.Box3d(
              center=[0, 0, 0],
              extents=[4000, 4000, 40],
          ),
          pml_thickness=[400, 400, 400, 400, 0, 0])

# Define a waveguide mode source.
source = [maxwell.WaveguideModeSource(center=[-1400, 0, 0],
                                      extents=[0, 200, 200],
                                      normal=[1, 0, 0],
                                      mode_num=0,
                                      power=1)]


# Define simulation outputs.
outputs1=[ maxwell.Epsilon(name="eps"),
             maxwell.ElectricField(name="field"),
             maxwell.WaveguideModeOverlap(name="overlap1",
                                          center=[1400, -1000, 0],
                                          extents=[0, 200, 200],
                                          normal=[1, 0, 0],
                                          mode_num=0,
                                          power=1),
             maxwell.WaveguideModeOverlap(name="overlap2",
                                          center=[1400, 1000, 0],
                                          extents=[0, 200, 200],
                                          normal=[1, 0, 0],
                                          mode_num=0,
                                          power=1)
             ]
```

```python
# Define the design region as a pixelated continuous shape, w
# of voxels whose permittivities can take on any value betwee
# `material2`.
var, design = goos.pixelated_cont_shape(
              initializer=initializer,
              pos=goos.Constant([0, 0, 0]),
              extents=[2000, 2500, 200],
              material=goos.material.Material(index=1),
              material2=goos.material.Material(index=3.45),
              pixel_size=[40, 40, 200],
              var_name="var_cont")
```

DEVICE REGION

SOURCE AND OUTPUT

Wavelength Demultiplexer

Let the following function represent the magnitude of the overlap integral of the Electric field simulation of the wavelengths (1500nm, 1600nm) in Waveguide-i.

$$f(E_i, W_i)$$

The objective function to be minimised was structured as:

$$obj = (1 - f(E_1, W_1))^2 + (0 - f(E_1, W_2))^2 + (1 - f(E_2, W_2))^2 + (0 - f(E_2, W_1))^2$$

```python
# Setup the simulation object.
sim1 = maxwell.fdfd_simulation(
        name="sim1",
        wavelength= wlens[0],
        background=goos.material.Material(index=1.0),
        eps=eps,
        simulation_space = my_simulation_space,
        solver = my_solver,
        sources = source,
        outputs= outputs1
   )
```

```python
Setup the optimization strategy.
"""
with plan:
  goos.opt.scipy_minimize(
              obj,
              "L-BFGS-B",
              monitor_list=[sim1["eps"], sim2["eps"], sim3["eps"],sim1["field"],sim1["overlap"],sim2["fi
              max_iters=40,
              name="opt_cont")
```

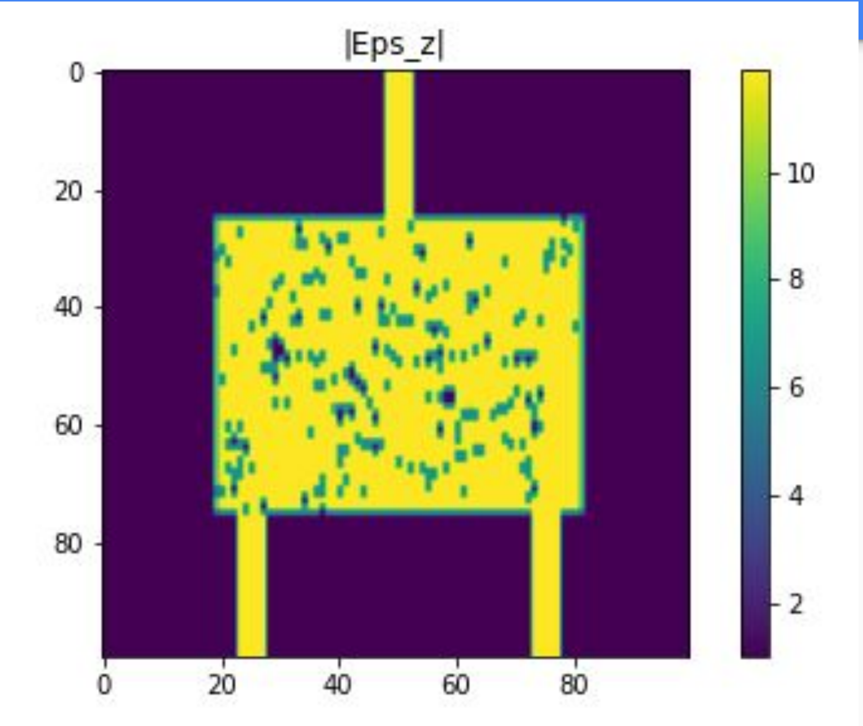
Discretisation

- SPINS optimises the objective function and gives out a continuous distribution of permittivity in the device.
- Necessity of discretisation to make the photonic device manufacturable.

To solve this constraint, we use the Sigmoid Discretisation.

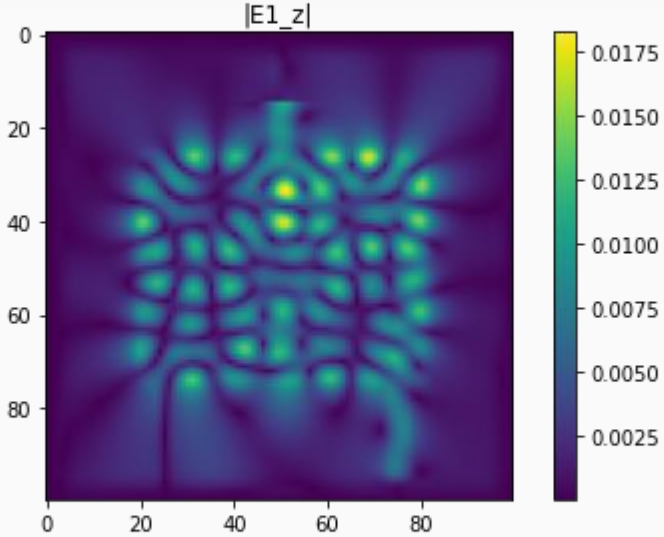
Waveguide Demultiplexer

Output permittivity distribution after optimising for 50 iteration and using sigmoid discretization of a factor of 50.



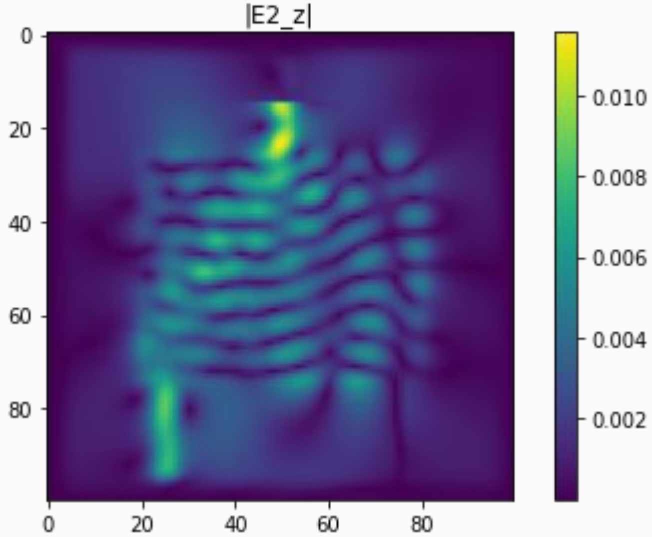
Waveguide Demultiplexer

WAVELENGTH=1500nm



$$f(E_1, W_1) = 0.0268$$
$$f(E_1, W_2) = 0.7561$$

WAVELENGTH=1600nm

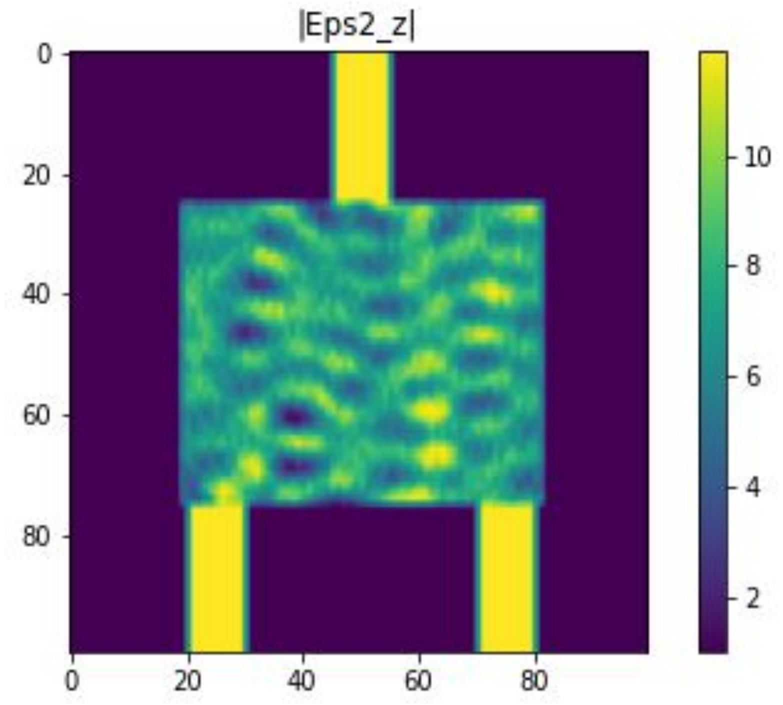


$$f(E_2, W_1) = 0.82292$$
$$f(E_2, W_2) = 0.03016$$

Wavelength Demultiplexer

Output permittivity distribution after optimising for 50 iteration and without using sigmoid discretisation.

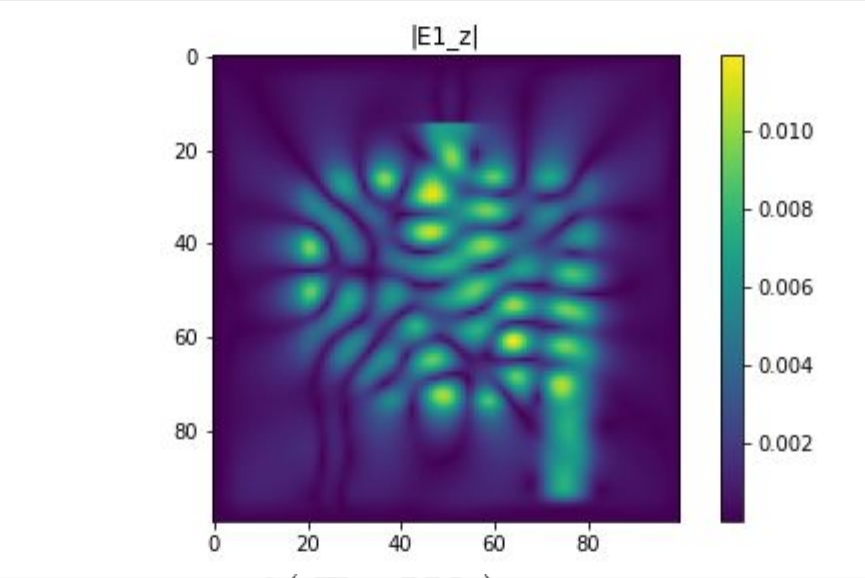
The pay-off is in the efficiency of transmission in the output waveguide.



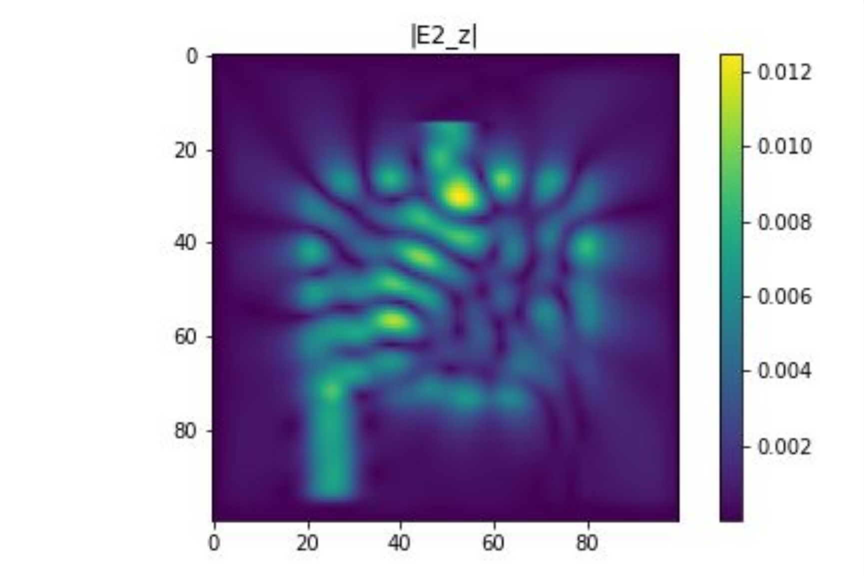
Waveguide Demultiplexer

WAVELENGTH=1500nm

WAVELENGTH=1600nm



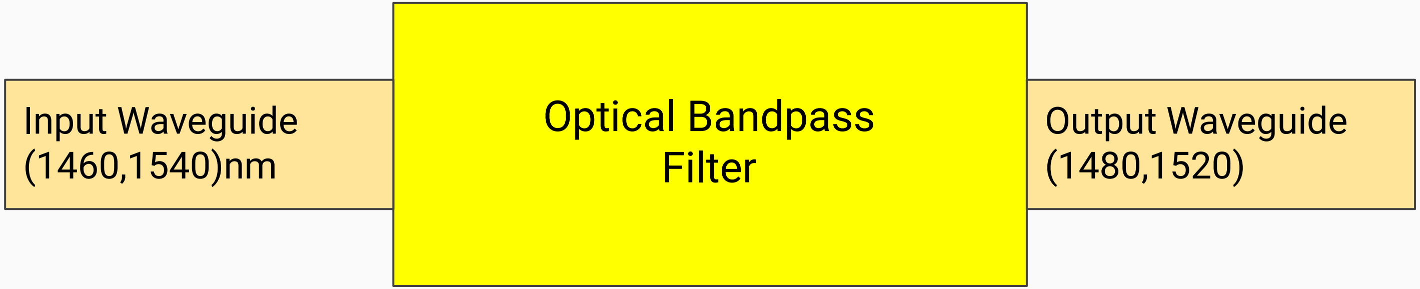
$$f(E_1, W_2) = 0.0028$$
$$f(E_1, W_2) = 0.9571$$



$$f(E_2, W_1) = 0.9702$$
$$f(E_2, W_2) = 0.00081$$

Optical Bandpass Filter

The aim was to create a Optical Bandpass filter that allows wavelengths in range (1480 nm-1520 nm) and suppressed the neighbouring wavelengths outside this band.



Optical Bandpass Filter

We define function f similar to the previous case and define objective corresponding to the i-th wavelength. Here t_i is the desired overlap.

$$obj_i = (t_i - f(E_i, W))^2$$

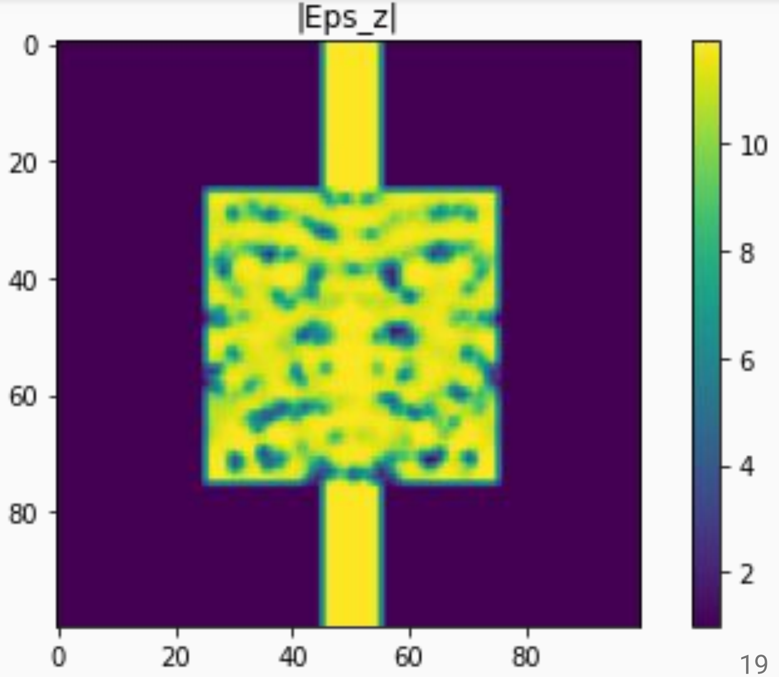
The objective function to be minimised was structured as:

$$obj = \sum_i obj_i$$

Optical Bandpass Filter

Output permittivity distribution after optimising for 40 iteration and using sigmoid discretization of a factor of 50.

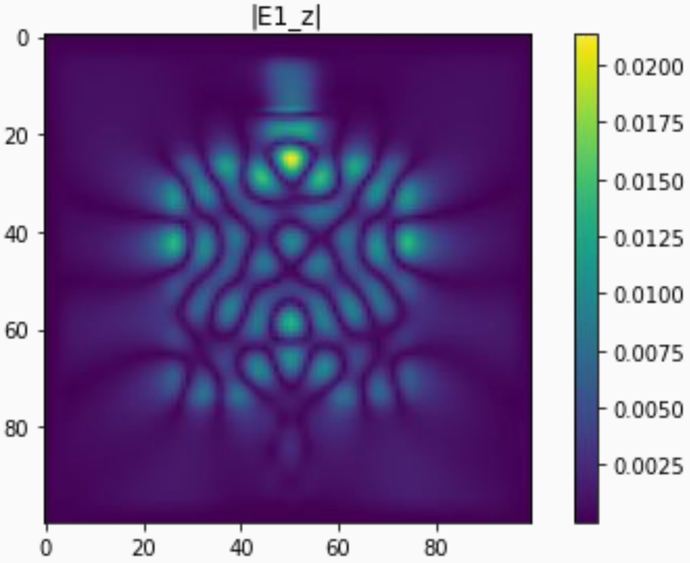
$$t_i = [0, 0.2, 1, 1, 1, 0.2, 0]$$
$$\lambda = [1470, 1475, 1480, 1500, 1520, 1525, 1530]$$



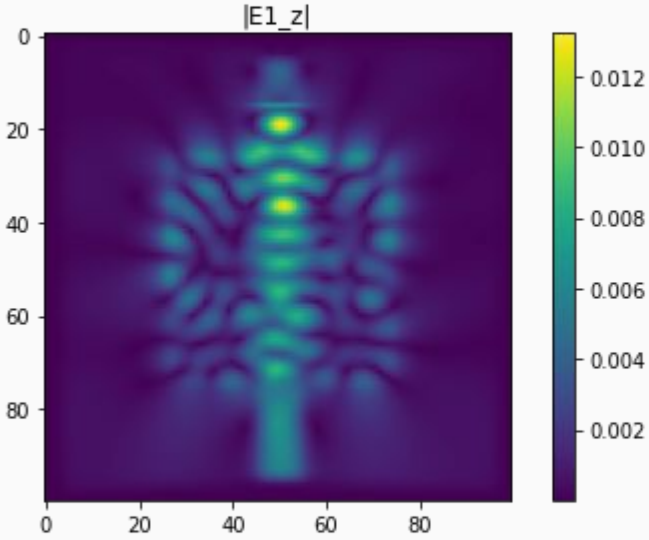
Optical Bandpass Filter

WAVELENGTH=1470 nm

WAVELENGTH=1500 nm



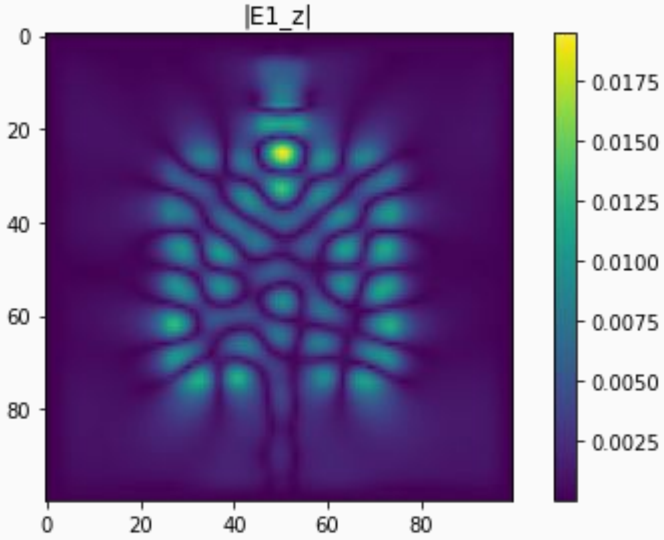
$$f(E_1, W) = 0.1515$$



$$f(E_4, W) = 0.8410$$

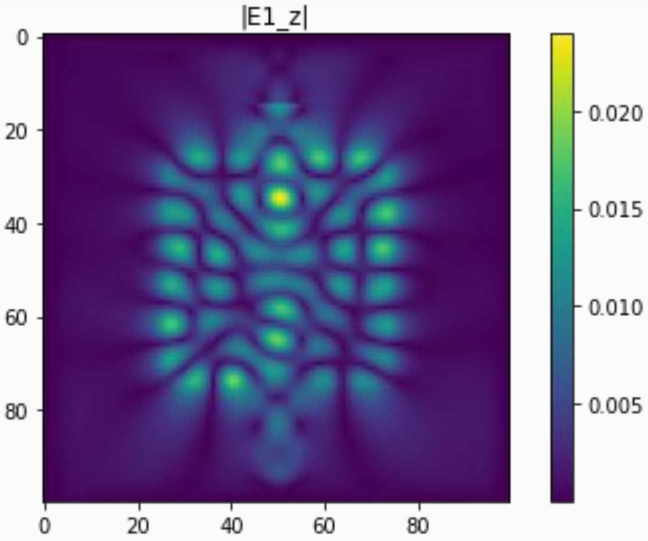
Optical Bandpass Filter

WAVELENGTH=1530 nm



$$f(E_7, W) = 0.0884$$

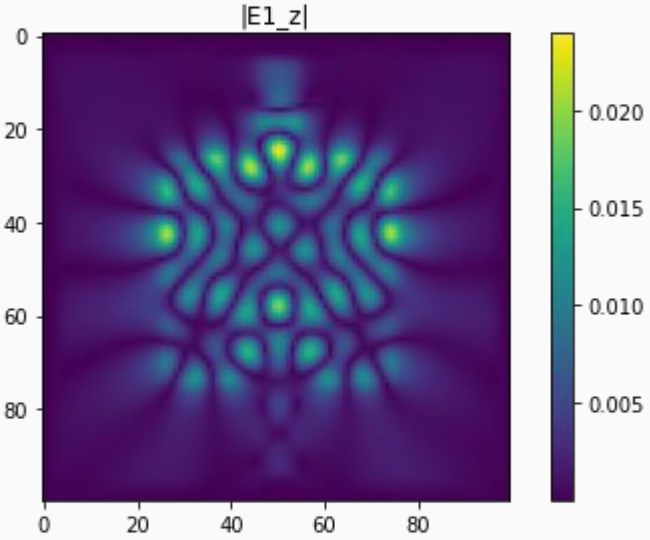
WAVELENGTH=1520 nm



$$f(E_5, W) = 0.8443$$

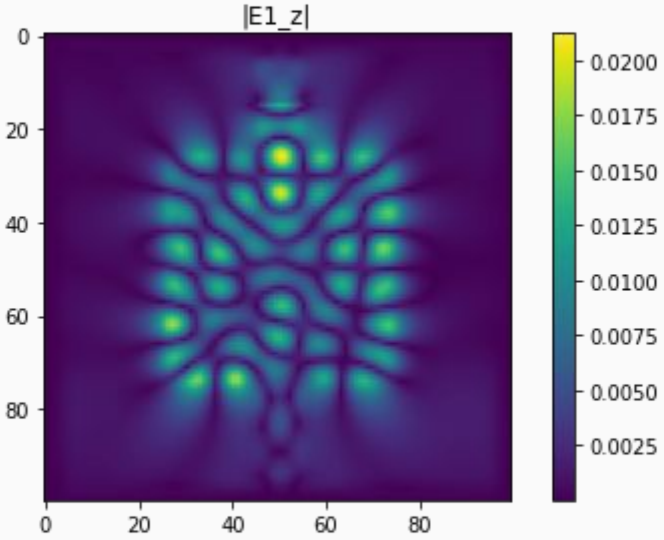
Optical Bandpass Filter

WAVELENGTH=1475 nm



$$f(E_2, W) = 0.2835$$

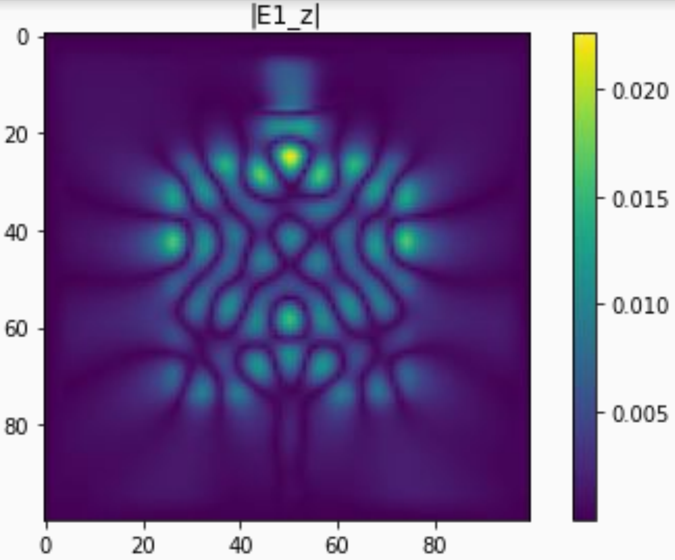
WAVELENGTH=1525 nm



$$f(E_6, W) = 0.3145$$

Optical Bandpass Filter

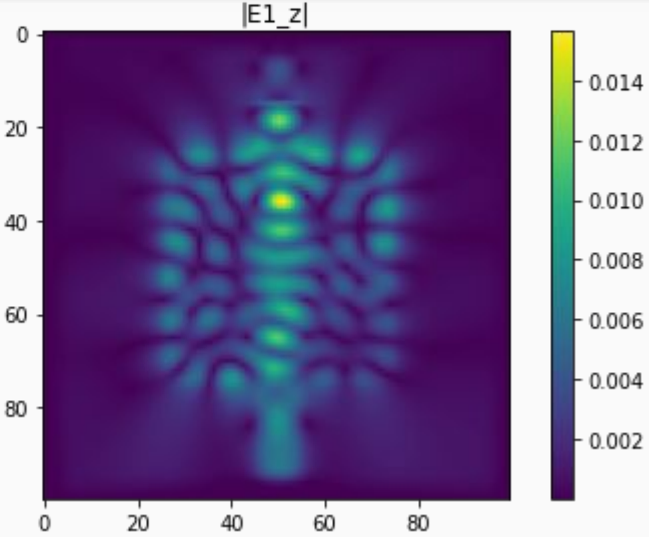
WAVELENGTH=1472 nm



$$f(E_8, W) = 0.04109$$

TEST CASES

WAVELENGTH=1510 nm

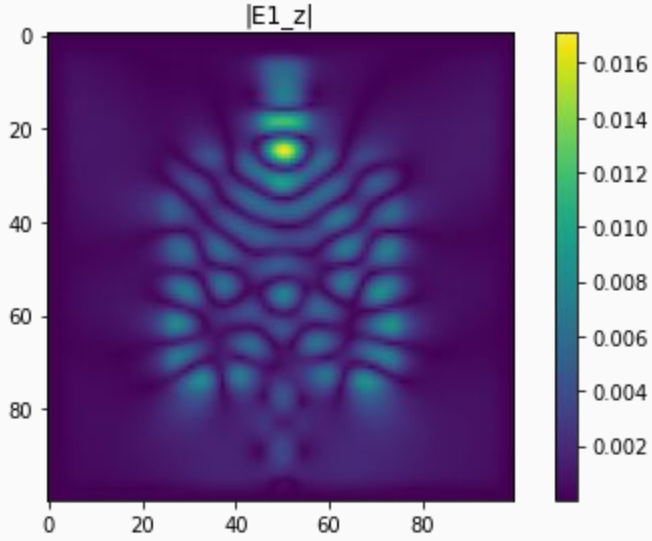


$$f(E_9, W) = 0.8828$$

Optical Bandpass Filter

WAVELENGTH=1540 nm

TEST CASES



$$f(E_{10}, W) = 0.338$$

Challenges in Design

- For multiple wavelength simulation, it is important choose correct spacing between wavelengths. Too many wavelengths will lead to long computation time. Sparse wavelengths will introduce intermediary variations.
- The initialisation of permittivity distribution is important. In one test case, despite using sigmoid discretisation, the output had only 500 out of 3000 pixels discretized.