

Simulating Ultra narrowband Metasurface

Yankun (Alex) Meng

In this simulation of [Paper Link](#), I did 1200–1600nm parameter sweep, and linear polarization. Finding the transmittance of this structure.

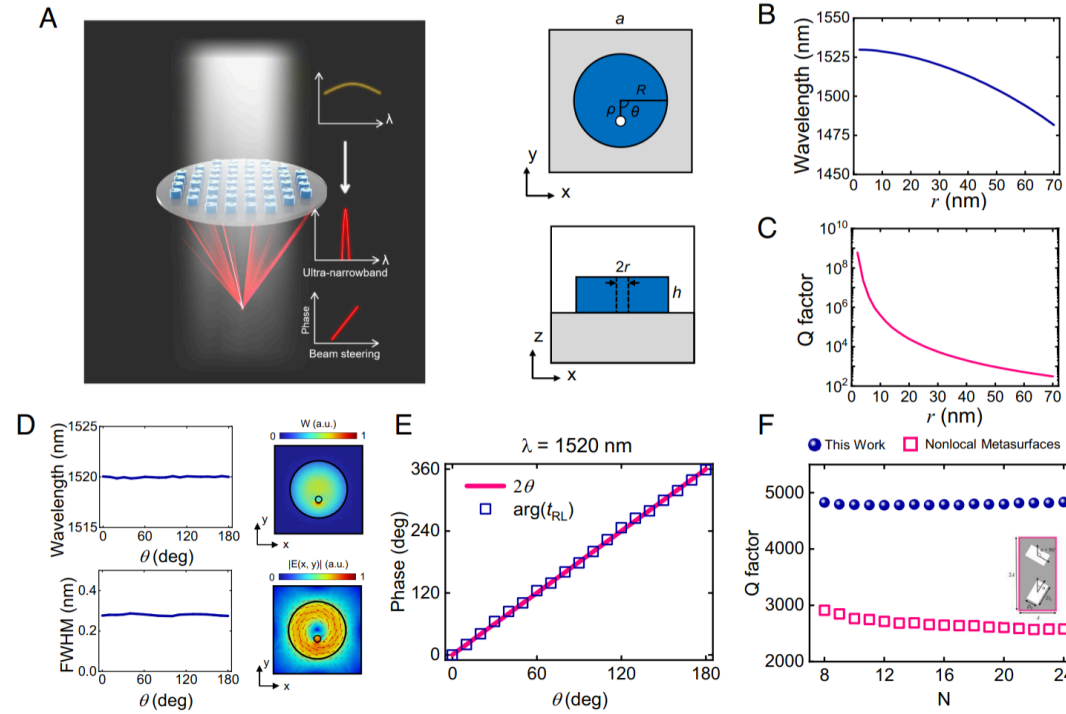
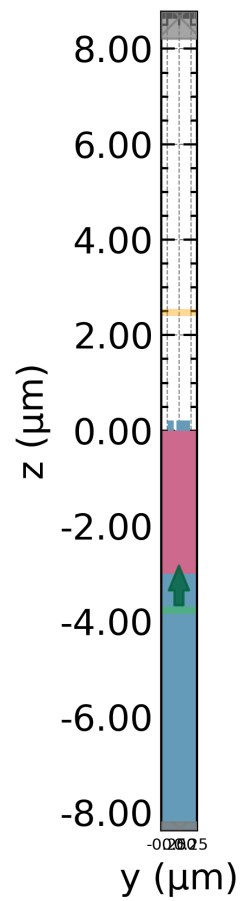


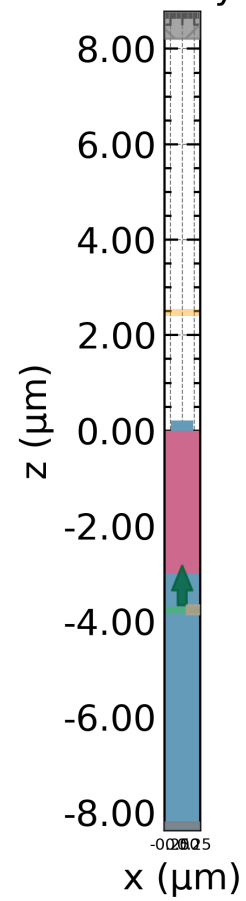
Fig. 1. Concept of high-Q phase-gradient metasurfaces. (A) Schematic picture of the Si metasurface. *Insets* show a single cell with detailed structural parameters. (B) and (C) show the dependence of resonant wavelength and Q factor of BIC mode on the radius of an air hole. (D) Resonant wavelength (*Top*) and FWHM (*Bottom*) of quasi-BIC mode as a function of rotational angle θ . Here, the size of the air hole is fixed at $r = 30$ nm. *Insets* show the numerically calculated distributions of power (*Top*) and electric field (*Bottom*). (E) Acquired geometric phase ϕ_{PB} (dots) as a function of rotational angle θ . The solid line represents the curve of $\phi_{PB} = 2\theta$. (F) Dependence of Q factors of supercells containing N phase steps of our metasurface (dots) and conventional nonlocal metasurface (open squares). Here, the position parameter ρ is fixed at $\rho = 80$ nm. The *Inset* shows the schematic of a conventional nonlocal metasurface structure.

Figure from paper

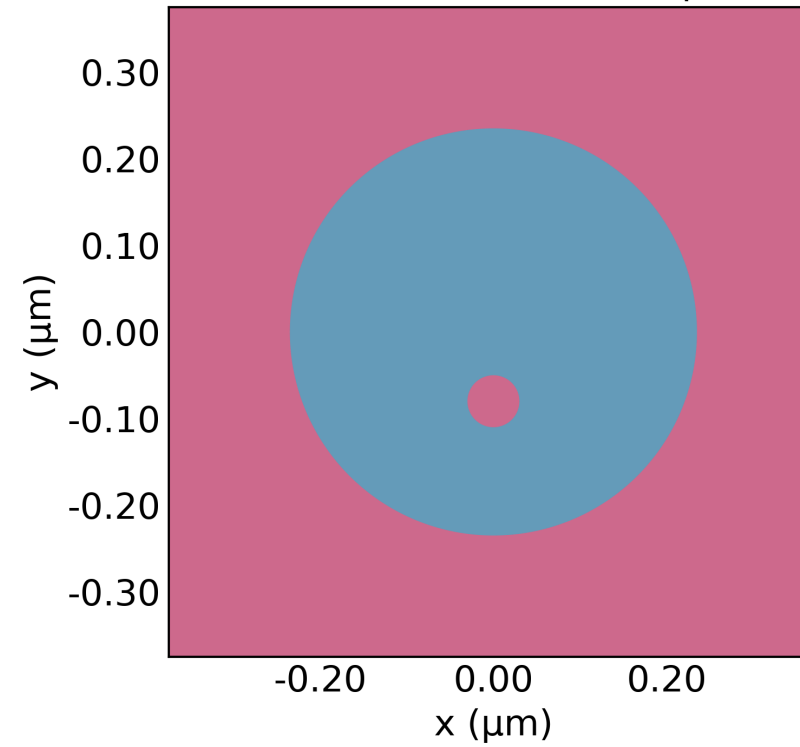
cross section at $x=0.00$ (μm)

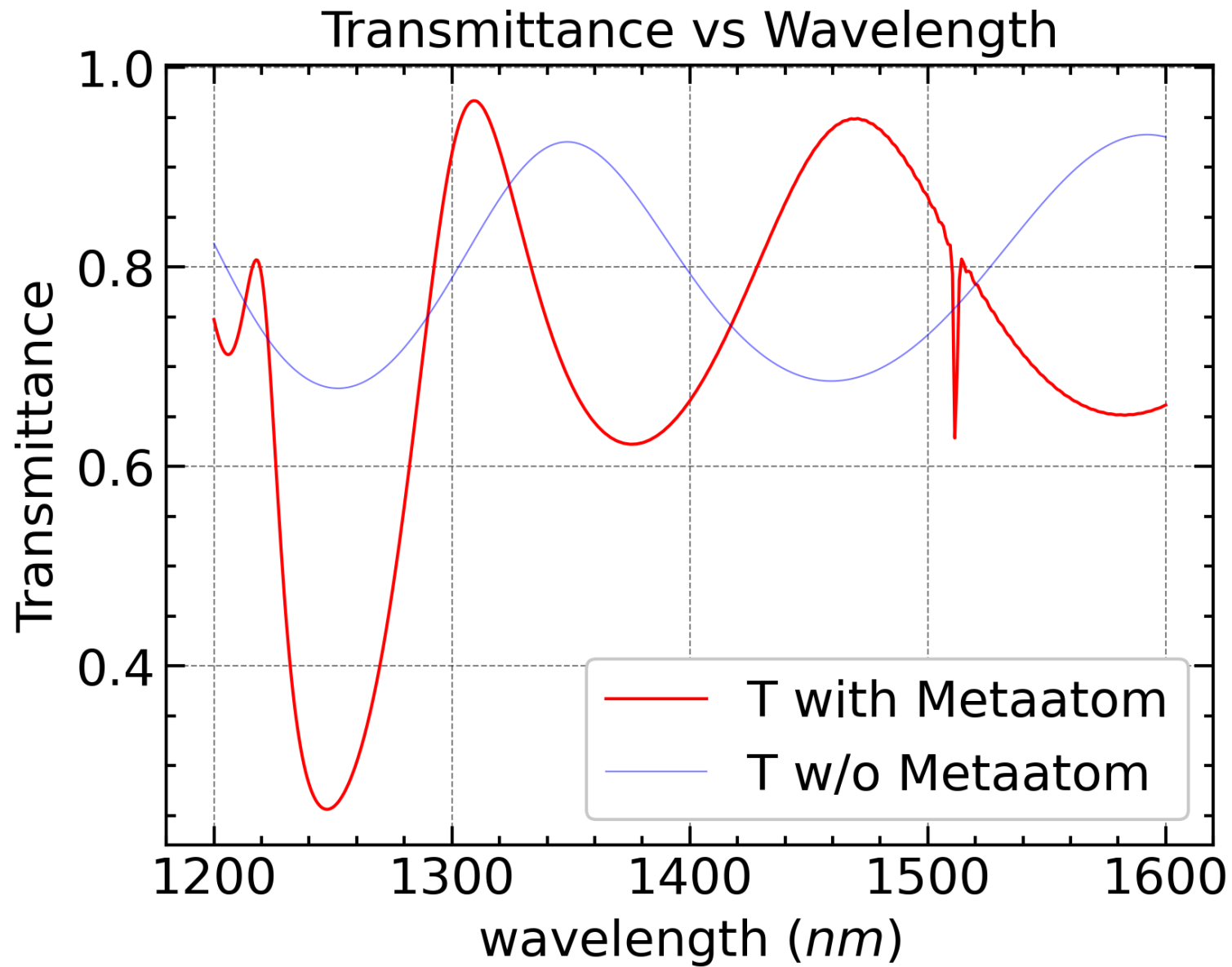


cross section at $y=0.00$ (μm)

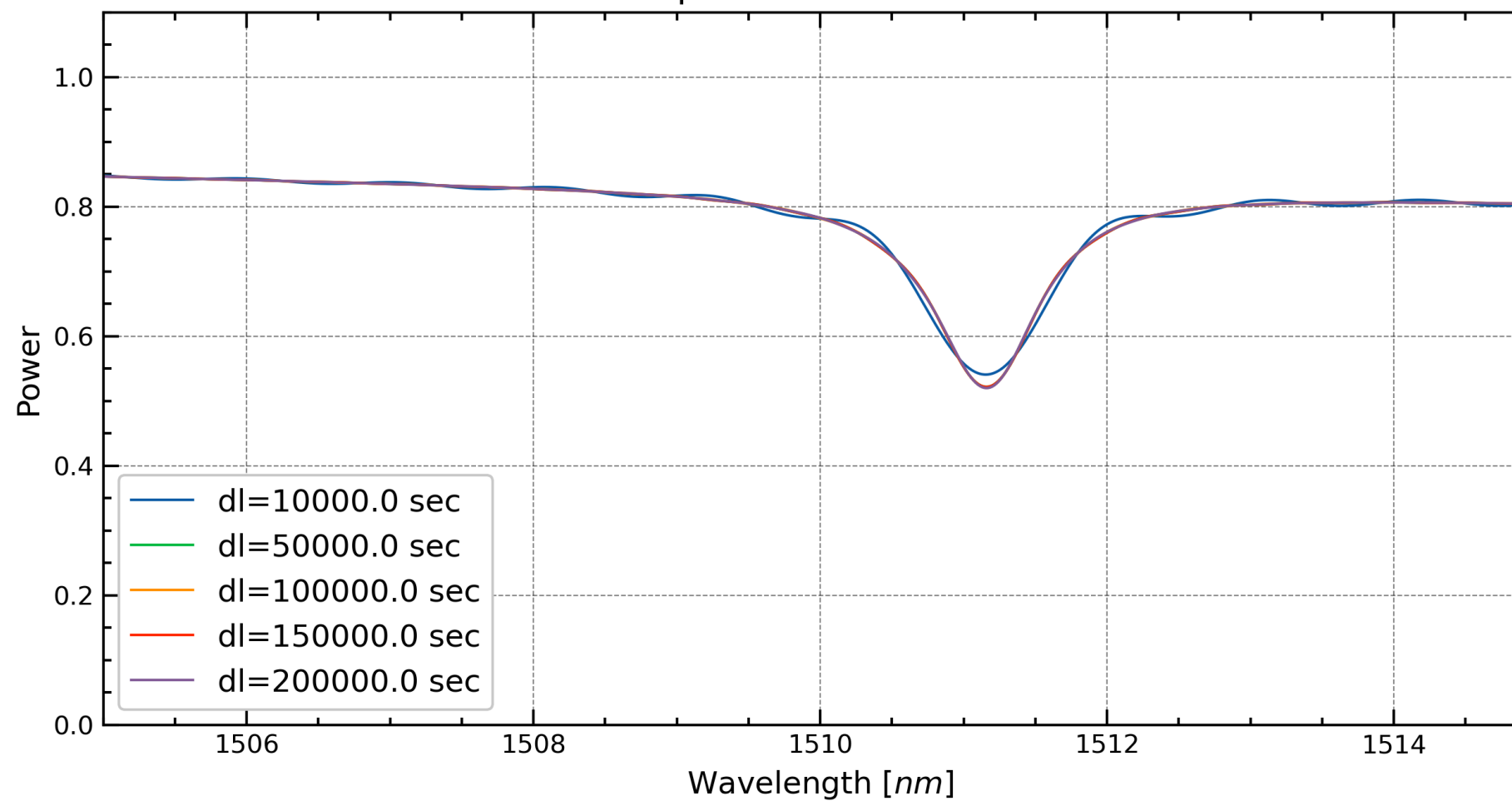


cross section at $z=0.00$ (μm)

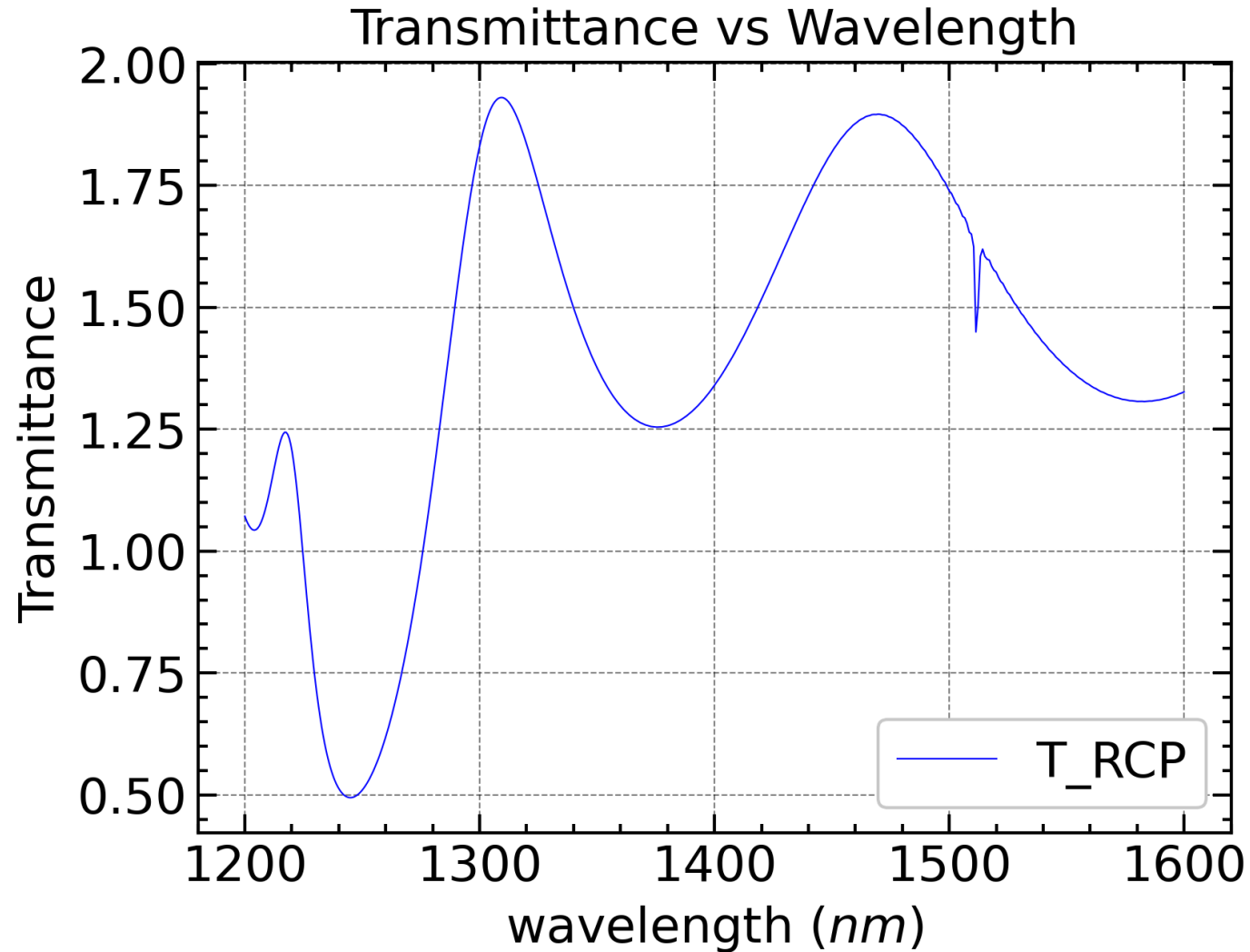




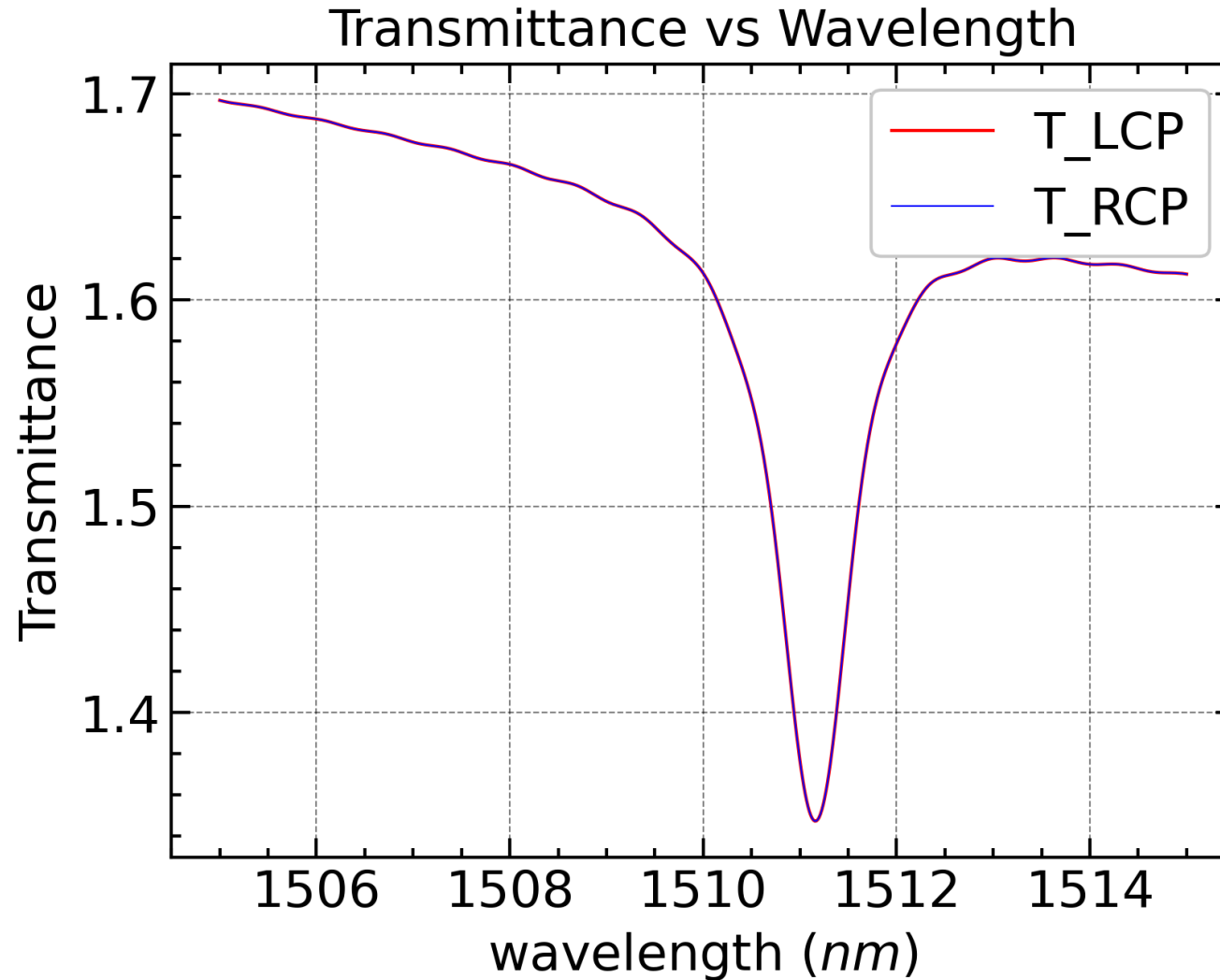
Transmission Spectra with Different Runtimes



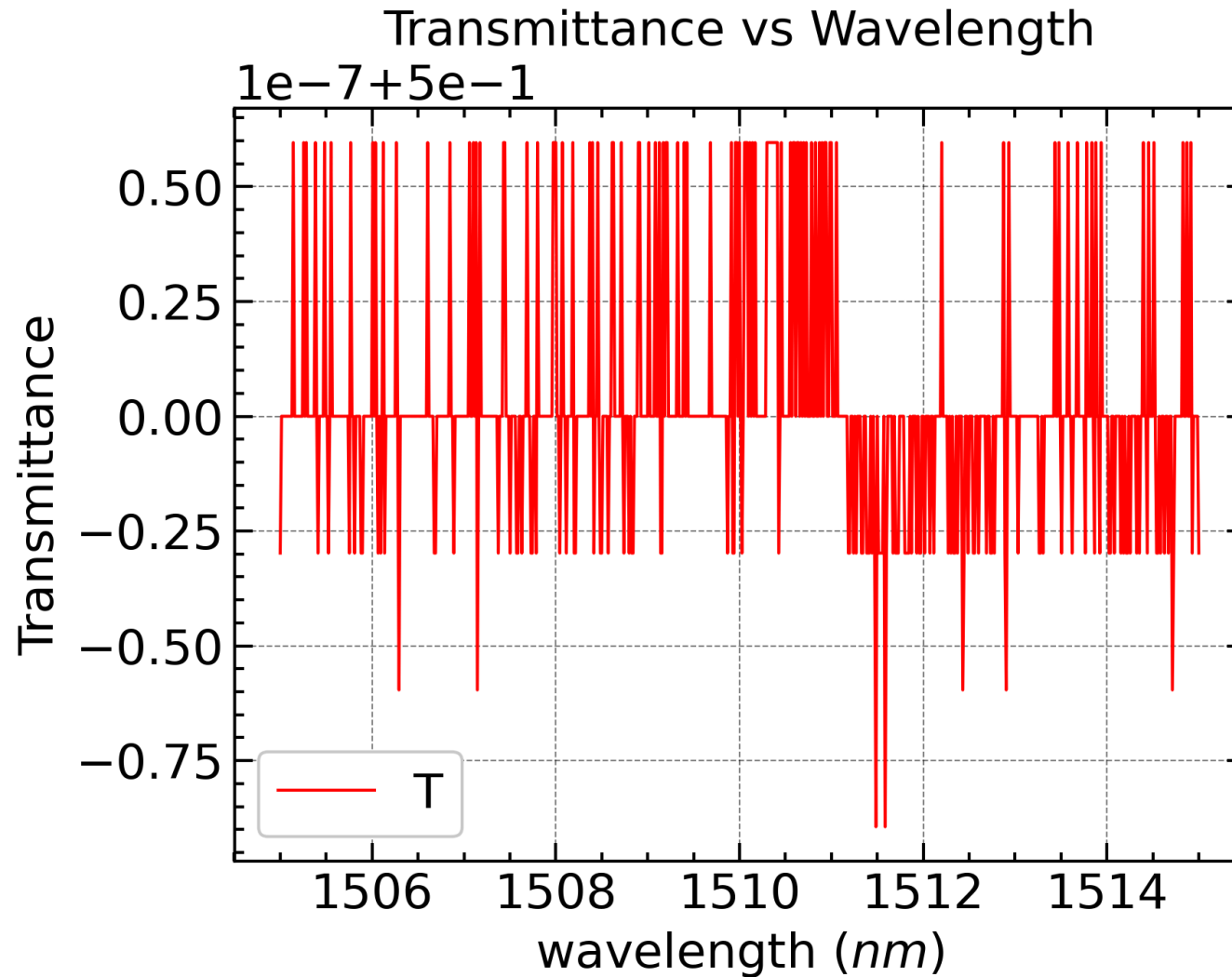
Circular Polarization



Circular Zoomed in



$$\text{LCP} / (\text{LCP} + \text{RCP})$$



Preconditions

```
1 # Import the necessary packages
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import tidy3d as td
5 import tidy3d.web as web
6 from tidy3d import material_library
7 import scienceplots
8
9 td.config.logging_level = "ERROR"
```

```
1 # 0 Define a FreqRange object with desired wavelengths
2 fr = td.FreqRange.from_wvl_interval(wvl_min=1.2, wvl_max=1.6)
3 N = 501 # num_points
4 fwidth = fr.fmax - fr.fmin
5 freq0 = fr.freq0
6 lda0 = td.C_0 / fr.freq0
```

```
1 # 1 Computational Domain Size
2 h = 0.210 # Height of cylinder
3 spc = 8
4 sh = 3 # height of the SiO2
5 Lz = spc + h + spc + h
6
7 Px = Py = P = 0.750 # periodicity
8 sim_size = [Px, Py, Lz]
```

```

1 # 2 Grid Resolution
2 dl = P / 32
3 horizontal_grid = td.UniformGrid(dl=dl)
4 vertical_grid = td.AutoGrid(min_steps_per_wvl=32)
5 grid_spec=td.GridSpec(
6     grid_x=horizontal_grid,
7     grid_y=horizontal_grid,
8     grid_z=vertical_grid,
9 )

```

```

1 # 3 Structures and Materials
2 R = 0.235 # radius of the cylinder
3 r = 0.030 # radius of the inner hole
4 p = 0.080 # distance between hole to center of circle
5 theta = np.deg2rad(90) # angle between x-axis and p vector
6
7 Si = material_library['cSi']['Green2008']
8 SiO2 = material_library['SiO2']['Horiba']
9
10 outer_cylinder = td.Cylinder(
11     center=[0, 0, h / 2],
12     radius=R,
13     length=h,
14     axis=2
15 )
16
17 inner_cylinder = td.Cylinder(
18     center=[p*np.cos(theta), -p*np.sin(theta), h / 2],
19     radius=r,
20     length=h,

```

```
1 source = td.PlaneWave(  
2     source_time=td.GaussianPulse(freq0=fr.freq0, fwidth=fwidth),  
3     size=(td.inf, td.inf, 0),  
4     center=(0, 0, -Lz/2 + spc - (sh - h) - 0.5 * lda0),  
5     direction="+",  
6     pol_angle=0,  
7 )
```

```
1 monitor = td.FluxMonitor(  
2     center=(0, 0, Lz/2 - spc + 1.5 * lda0),  
3     size=(td.inf, td.inf, 0),  
4     freqs=fr.freqs(N),  
5     name="flux_monitor"  
6 )
```

```
1 bandwidth = fr.fmax - fr.fmin
2 run_time = 500 / bandwidth # run_time for the transmittance simulation
```

```
1 bc = td.BoundarySpec(
2     x=td.Boundary.periodic(),
3     y=td.Boundary.periodic(),
4     z=td.Boundary.pml()
5 )
```

Simulation

```
1 sim_empty = td.Simulation(  
2     size=sim_size,  
3     grid_spec=grid_spec,  
4     structures=[substrate, dioxide],  
5     sources=[source],  
6     monitors=[monitor],  
7     run_time=run_time,  
8     boundary_spec=bc  
9 )  
10  
11 sim_actual = td.Simulation(  
12     size=sim_size,  
13     grid_spec=grid_spec,  
14     structures=[substrate, dioxide, atom],  
15     sources=[source],  
16     monitors=[monitor],  
17     run_time=run_time,  
18     boundary_spec=bc  
19 )
```

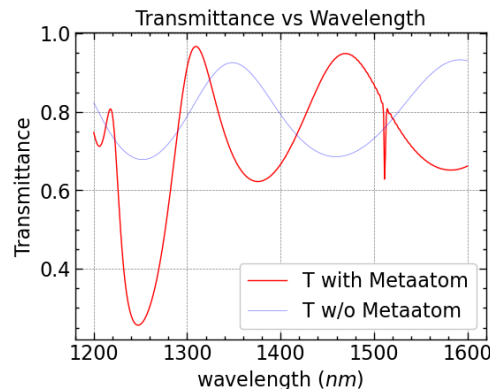
```
1 sims = {  
2     "actual": sim_actual,  
3     "norm": sim_empty  
4 }
```

```
1 sims["actual"].plot_3d()
```

Postprocess

```
1 T_actual = batch_data["actual"]["flux_monitor"].flux
2 T_norm = batch_data["norm"]["flux_monitor"].flux
```

```
1 # this uses scienceplots to make plots look better
2 plt.style.use(['science', 'notebook', 'grid'])
3 # plot transmission, compare to paper results, look similar
4 fig, ax = plt.subplots(1, 1, figsize=(6, 4.5))
5 plt.plot(td.C_0 / fr.freqs(N) * 1000, T_actual, "r", lw=1, label="T with Metaatom")
6 plt.plot(td.C_0 / fr.freqs(N) * 1000, T_norm, "b", lw=0.5, alpha=0.5, label="T w/o Metaa")
7 plt.xlabel(r"wavelength ($nm$)")
8 plt.ylabel("Transmittance")
9 plt.legend()
10 plt.title("Transmittance vs Wavelength")
11 plt.savefig("power_geom", dpi=300)
12 plt.show()
```



Circular Polarization

```
1 def circular_polarized_plane_wave(pol):
2     # define a plane wave polarized in the x direction
3     plane_wave_x = td.PlaneWave(
4         source_time=td.GaussianPulse(freq0=fr.freq0, fwidth=fwidth),
5         size=(td.inf, td.inf, 0),
6         center=(0, 0, -Lz/2 + spc - (sh - h) - 0.5 * lda0),
7         direction="+",
8         pol_angle=0,
9     )
10
11     # determine the phase difference given the polarization
12     if pol == "left":
13         phase = -np.pi / 2
14     elif pol == "right":
15         phase = np.pi / 2
16     else:
17         raise ValueError("pol must be `left` or `right`")
18
19     # define a plane wave polarized in the y direction with a phase difference
20     plane_wave_y = td.PlaneWave(
21         source_time=td.GaussianPulse(freq0=freq0, fwidth=fwidth, phase=phase),
22         size=(td.inf, td.inf, 0),
23         center=(0, 0, -Lz/2 + spc - (sh - h) - 0.5 * lda0),
24         direction="+",
25         pol_angle=np.pi / 2,
26     )
27
```