Transmission of Huygens' Surface

Yankun (Alex) Meng

Important Questions to answer:

- Why do we need a normalizing run? (FDTD 101)
- Why is there a mismatch between `td.FreqRange.wvl0` and `td.C_0 / td.FreqRange.freq0`

Huygens' Metasurface Paper Link

```python
import matplotlib.pyplot as plt
import numpy as np
import tidy3d as td
import tidy3d.web as web
```

Preconditions

```
# 0 Define a FreqRange object with desired wavelengths
fr = td.FreqRange.from_wvl_interval(wvl_min=1.1, wvl_max=1.6)
N = 301 # num_points
```

```
# 1 Computational domain size
h = 0.220  # Height of cylinder
spc = 2
Lz = spc + h + h + spc

Px = Py = P = 0.666 # periodicity
sim_size = [Px, Py, Lz]
```

```
# 2 Grid Specifications
dl = P / 32
horizontal_grid = td.UniformGrid(dl=dl)
vertical_grid = td.AutoGrid(min_steps_per_wvl=32)
grid_spec=td.GridSpec(
    grid_x=horizontal_grid,
    grid_y=horizontal_grid,
    grid_z=vertical_grid,
)
```

```
# 3 Structures
r = 0.242  # radius of the cylinder
n_Si = 3.5
Si = td.Medium(permittivity=n_Si**2, name='Si')
cylinder = td.Structure(
```

Simulation

```
sim_empty=td.Simulation(
    size=sim_size,
    grid_spec=grid_spec,
    structures=[substrate, glass],
    sources=[source],
    monitors=[monitor],
    run_time=run_time,
    boundary_spec=bc
)
```

```
sim_actual = td.Simulation(
    size=sim_size,
    grid_spec=grid_spec,
    structures=[substrate, glass, cylinder],
    sources=[source],
    monitors=[monitor],
    run_time=run_time,
    boundary_spec=bc
)
```

```
fig, (ax1,ax2,ax3) = plt.subplots(1, 3, figsize=(10, 6))
sim_actual.plot_eps(x=0, ax=ax1)
sim_actual.plot_eps(y=0, ax=ax2)
sim_actual.plot_eps(z=0, ax=ax3)
```

Postprocess

- 35 seconds to run both simulations
- results are stored inside `batch_data`

```python
T = batch_data["actual"]["flux"].flux / batch_data["norm"]["flux"].flux
```

```python
# plot transmission, compare to paper results, look similar
fig, ax = plt.subplots(1, 1, figsize=(6, 4.5))
plt.plot(fr.freqs(N), T, "r", label="T")
plt.xlabel(r"Frequency ($Hz$)")
plt.ylabel("Magnitude")
plt.ylim(0, 1)
plt.legend()
plt.show()
```