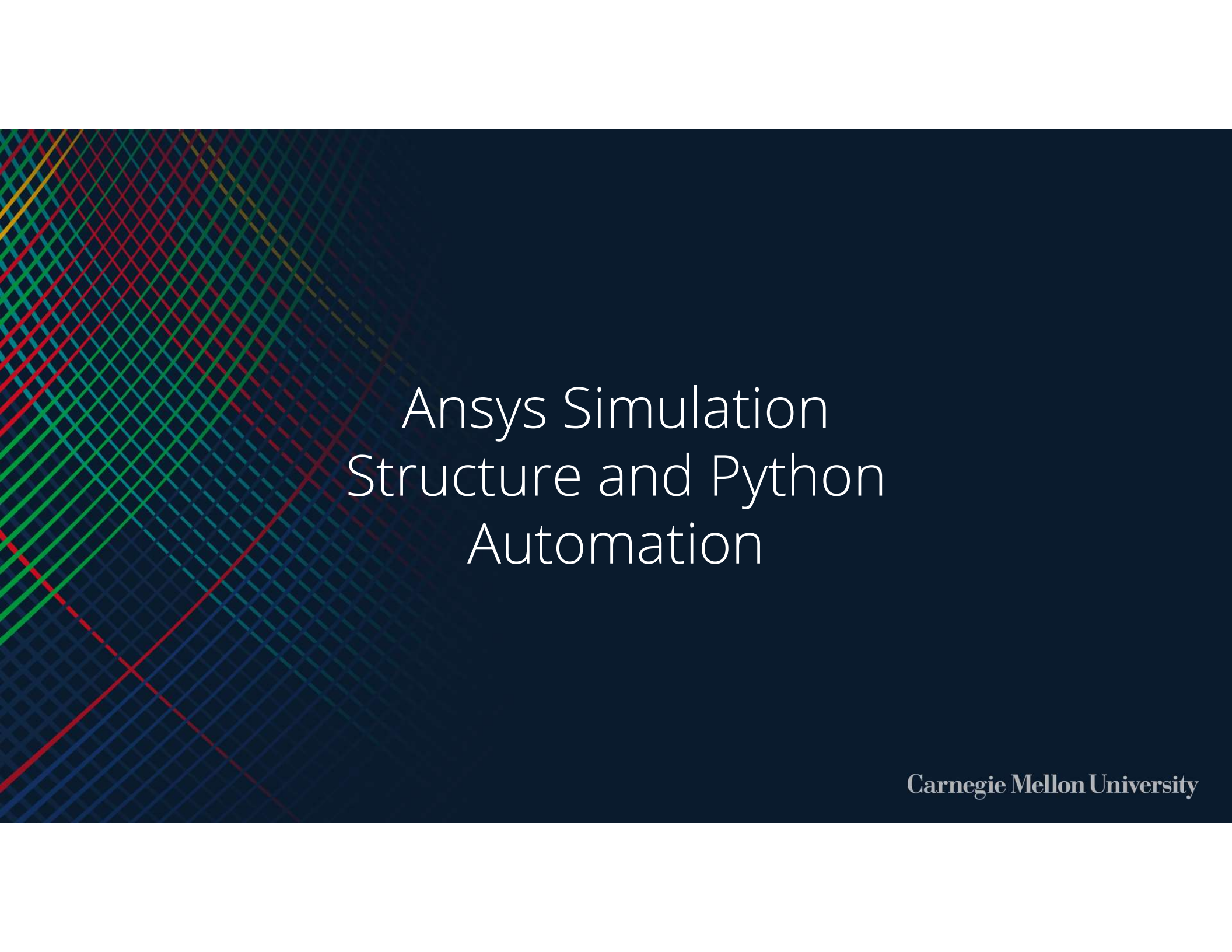




Final Report

MAY 4TH, 2024

Sunghwan Baek



Ansys Simulation Structure and Python Automation

Carnegie Mellon University



Automated Process of Making Structures

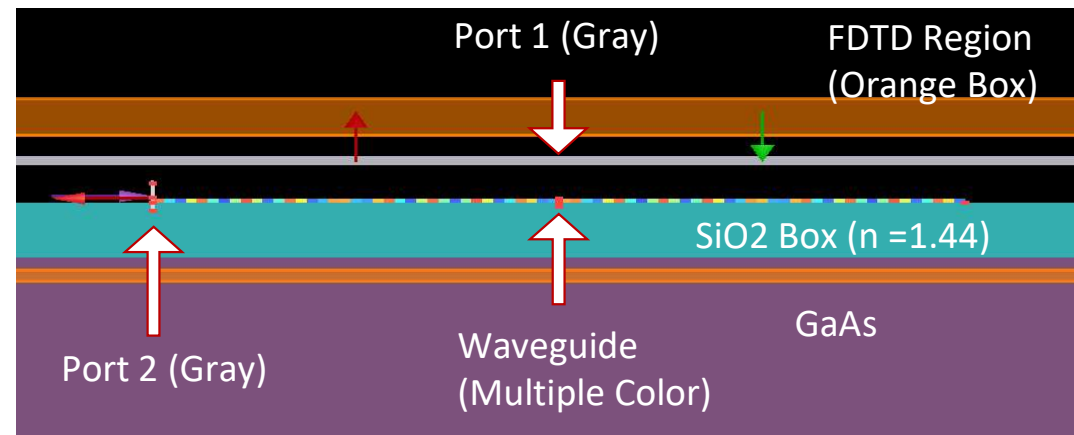
- Utilizing the Ansys Python API enables meticulous modification of each structural dimension, ensuring heightened accuracy and alignment with project specifications during simulations.
- This approach fosters precision and enhances adaptability, owing to the integration of these parameters within the Python codebase, allowing for seamless adjustments to structural sizes as necessary.

```
basedir = os.getcwd()
# hide should be False, in order to make all process automated
fdtd = lumapi.FDTD(hide=False)
fdtd.load(basedir + '/Ansys_Project_Fall2023_part1.fsp')
fdtd.switchtolayout()
fdtd.select('wave_guide')
fdtd.delete()
make_nk(fdtd, x_span, a)
fdtd.select('::model::FDTD')
fdtd.set('x', 0)
fdtd.set("x span", 3*x_span_end+4*10**(-6))
fdtd.set('y', 0.0)
fdtd.set("y span", y_span)
fdtd.select('::model::FDTD::ports::port 1')
fdtd.set('x', 0)
fdtd.set('x span', 3*x_span_end)
fdtd.set('y', y_span/2 - 1*10**(-6))
fdtd.set('z', 0)
fdtd.set("z span", 0.3*10**(-6))
fdtd.select('::model::FDTD::ports::port 2')
fdtd.set('x', (10**(-10))-x_span/2)
fdtd.set('y', 0.2*10**(-6))
fdtd.set("y span", 1*10**(-6))
fdtd.set('z', 0)
fdtd.set("z span", 0.3*10**(-6))
fdtd.run()
```

Waveguide Core

The rod with multiple colors in the image is the waveguide with sinusoidal modulation

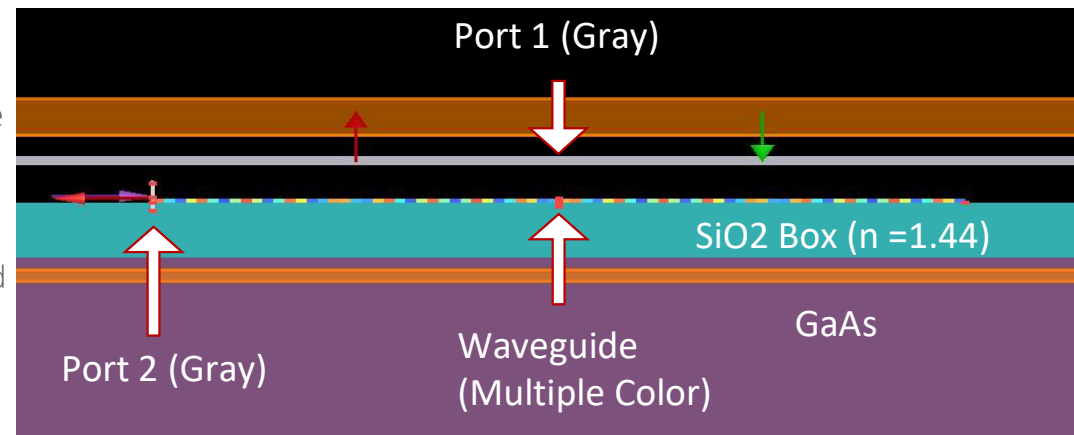
- Material of choice of the Waveguide: A substance with a refractive index of 2.0.
- Serves as the primary medium for light propagation in the experiments.
- The waveguide is placed atop a SiO₂ box for support and isolation.
- Refractive index of the SiO₂ box: 1.44.

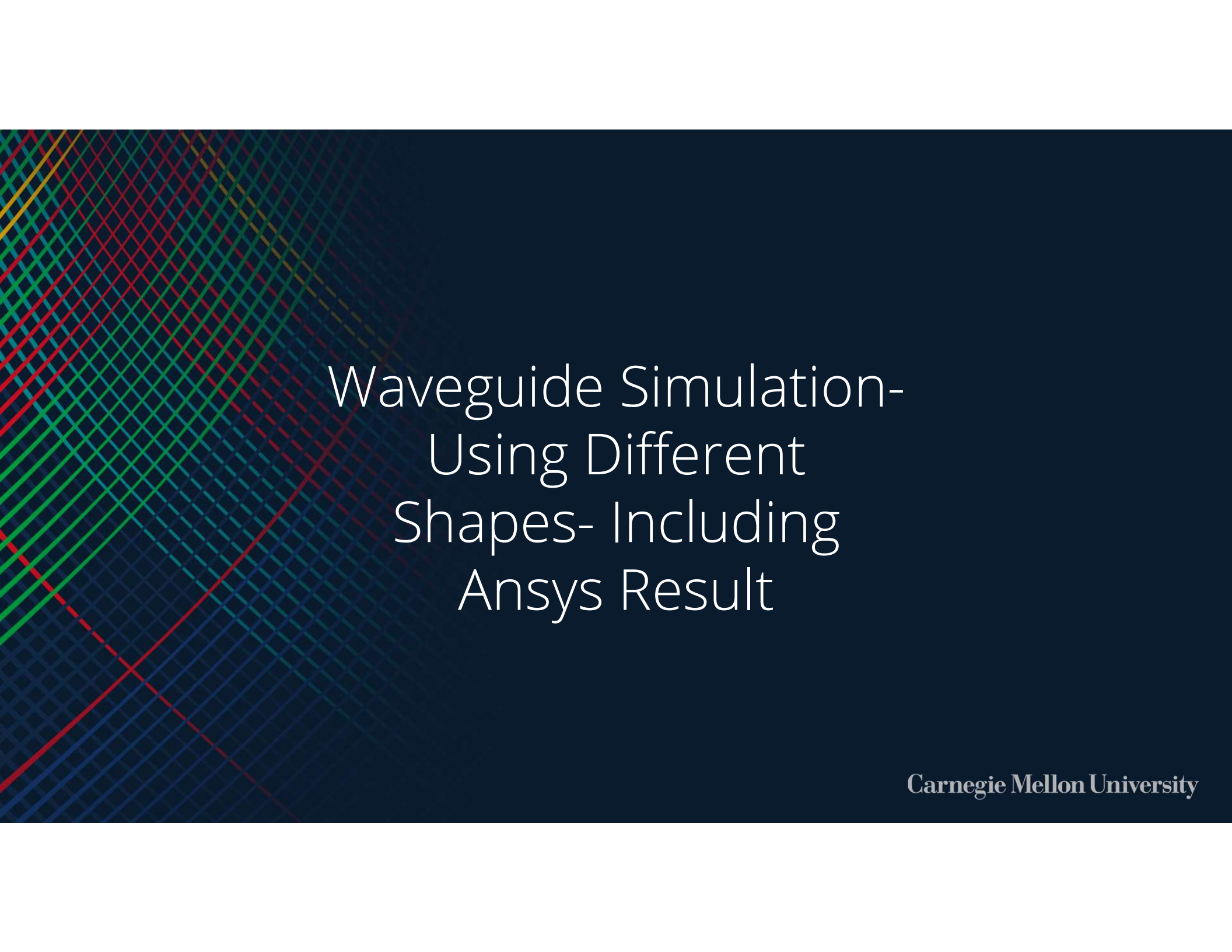


Mode Injection & Data Collection

Port1,2 and far-field data

- Port 2 is used to introduce the desired light mode.
- Due to refractive index differences, the mode remains primarily within the waveguide.
- After light propagation, E-field data is gathered at port 1. Utilized the far-field function in ANSYS FDTD for detailed analysis.
- Port 1 was designed to be bigger relative to the waveguide dimensions, optimizing precision in far-field data acquisition across various angles.





Waveguide Simulation- Using Different Shapes- Including Ansys Result



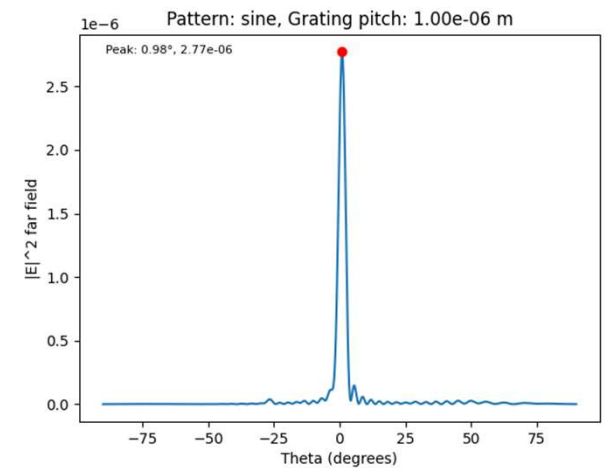
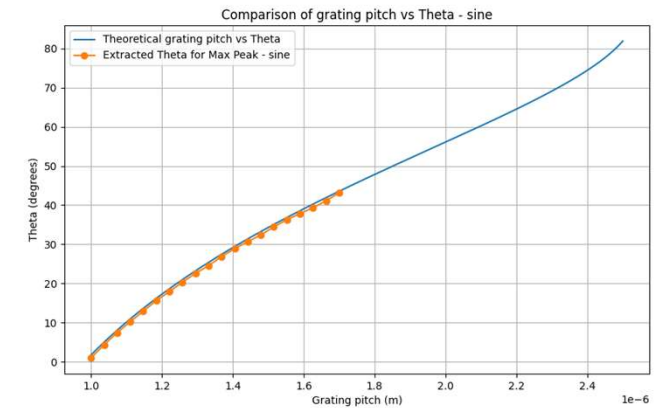
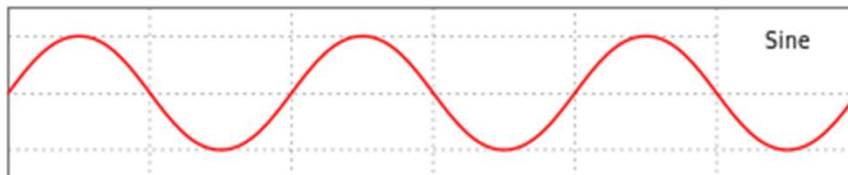
Waveguide Modulation Automation in Python

- By fine-tuning the `K_grating`, I could dynamically adjust the acoustic wavelength within the waveguide core, offering control over modulation characteristics.
- Moreover, the amplitude and shape of modulation are customizable, enabling precise control over the refractive index profile.
- I could adjust the waveguide size using the `x_span` variable for each simulation.

```
def make_nk(fdt, x_span, a):  
  
    # Define the Parameters  
    sinusoidal_amplitude = 0.05 * (a+1)  
    x_points = 100  
    y_points = 2  
    z_points = 2  
    x_start = -(x_span)/2  
    x_stop = (x_span)/2  
    y_start = -0.15*10**(-6)  
    y_stop = 0.15*10**(-6)  
    z_start = -0.15*10**(-6)  
    z_stop = 0.15*10**(-6)  
    x_range = np.linspace(x_start, x_stop, x_points)  
    y_range = np.linspace(y_start, y_stop, y_points)  
    z_range = np.linspace(z_start, z_stop, z_points)  
  
    # Data Generation  
    n_data = np.zeros((x_points, y_points, z_points))  
    k_data = np.zeros((x_points, y_points, z_points))  
  
    for i, x in enumerate(x_range):  
        for j, y in enumerate(y_range):  
            for k, z in enumerate(z_range):  
                n_data[i, j, k] = 2.0 + sinusoidal_amplitude * np.sin(K_grating * x)  
                k_data[i, j, k] = 0
```

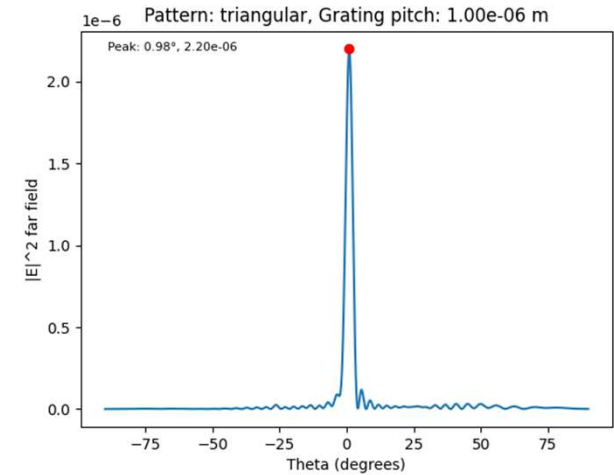
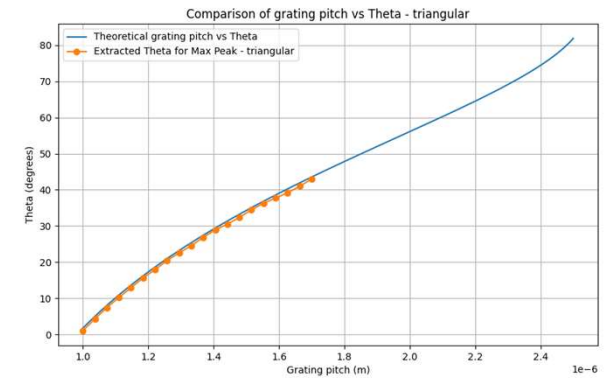
Sine Wave Grating

The peak position graphs follow the theoretical graph almost perfectly.



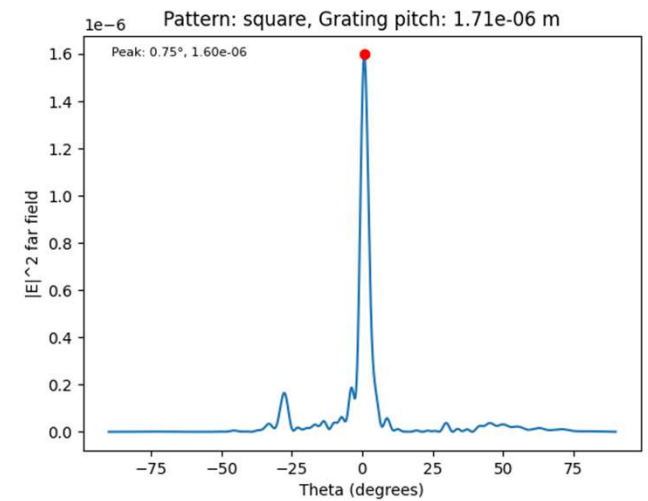
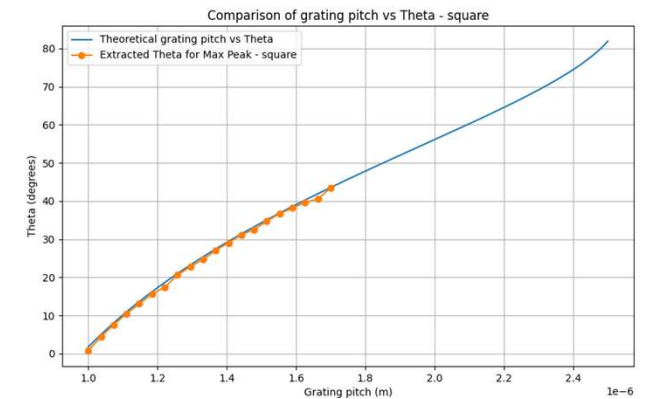
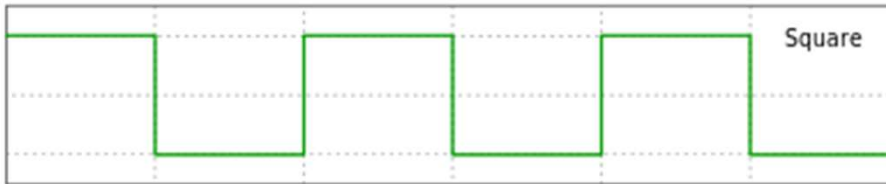
Triangle Wave Grating

Triangle wave grating generates the peak at the same theta, but the amplitude of the peak is smaller than that of sine wave grating.



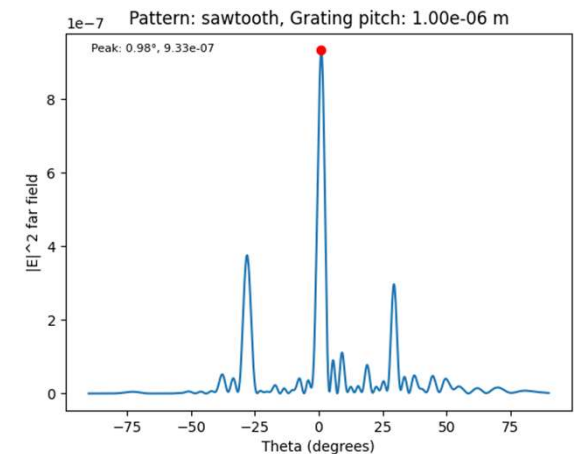
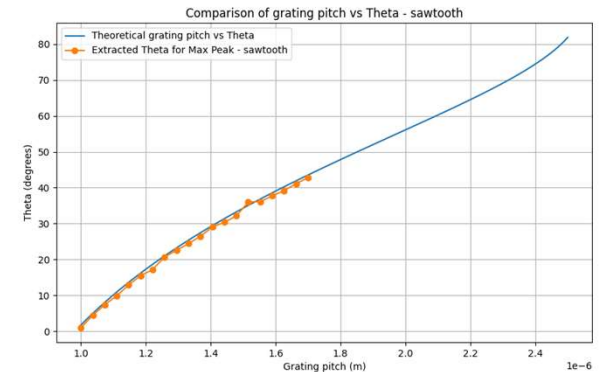
Square Wave Grating

It had a peak smaller than sine and triangle wave gratings.



Sawtooth Wave Grating

Sawtooth wave grating had the peak at the same theta. the smallest peak among the four types. There are other small peaks around the major peak and that is the reason why the major peak is smaller than other pattern wave grating.

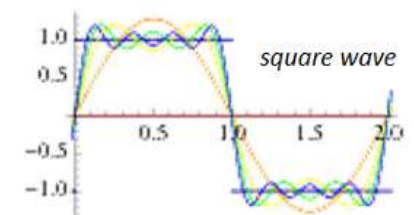
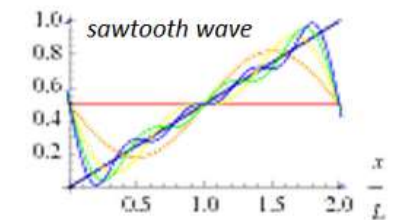
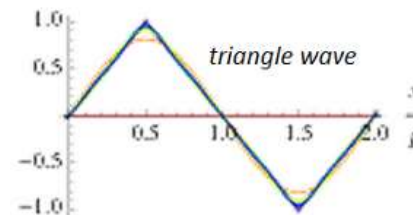


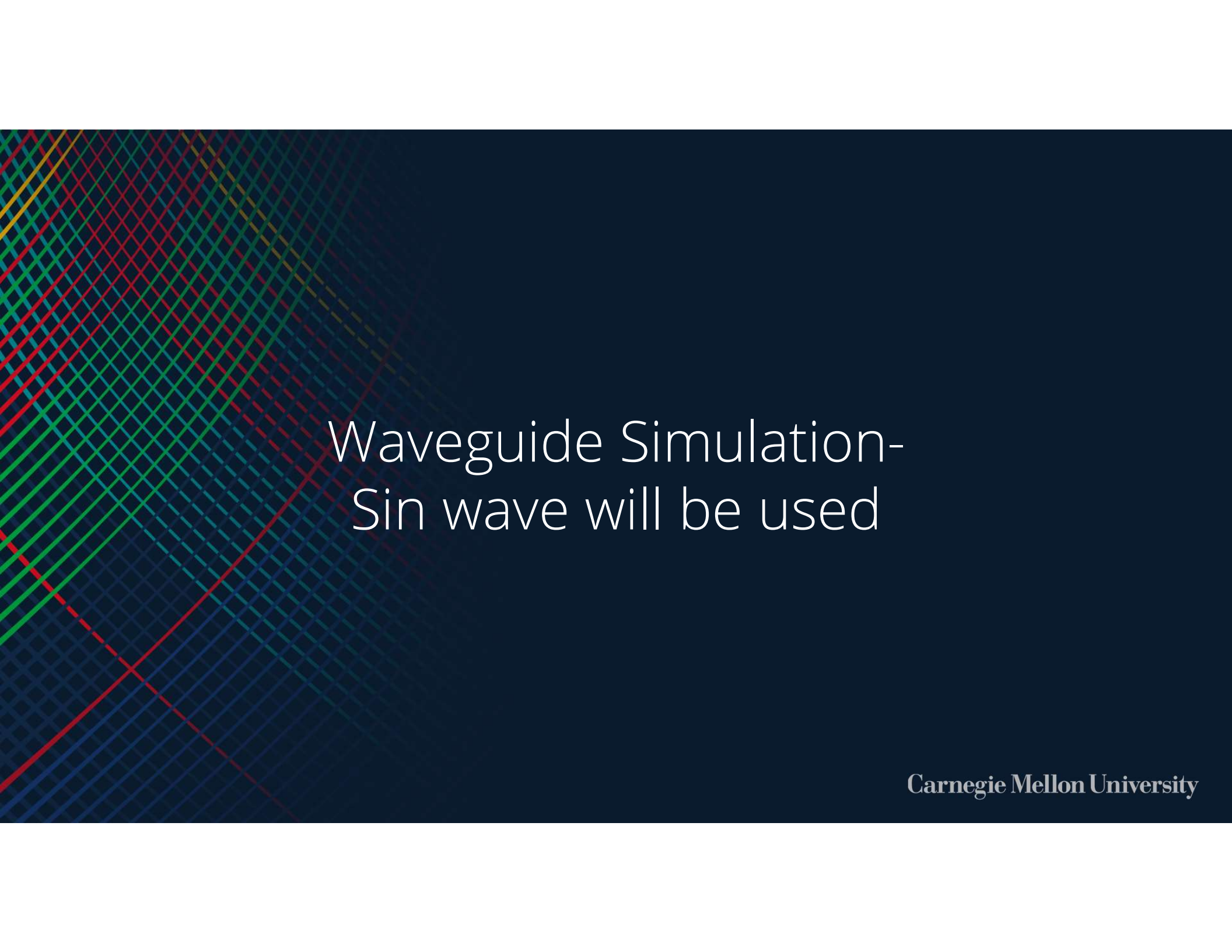
Peak Amplitude Efficiency

Sine wave grating > triangle wave grating > square wave grating > sawtooth wave grating—directly results from how each pattern's Fourier series distributes energy among the diffraction orders. This hierarchy underscores the importance of the shape of the grating pattern in determining its optical diffraction efficiency.

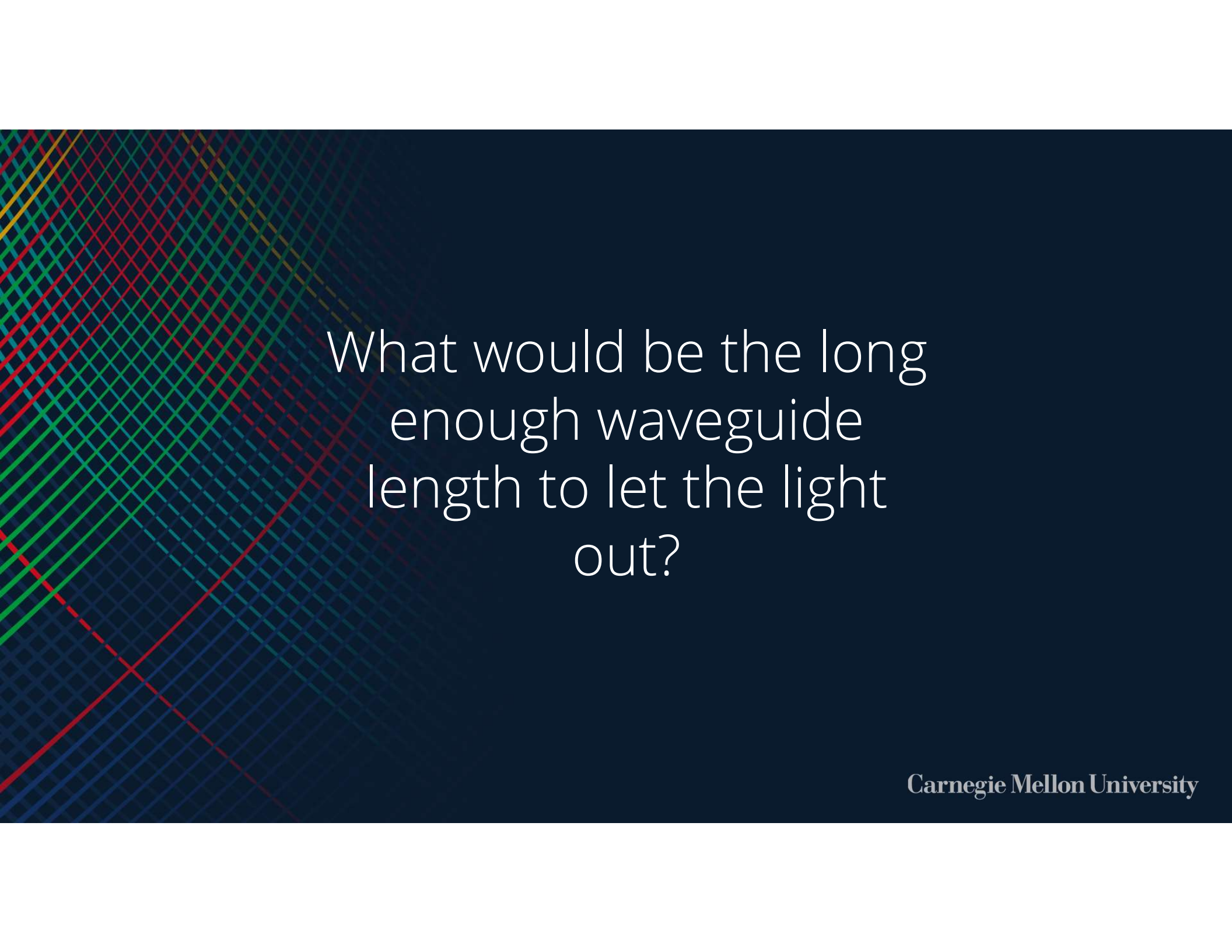
Therefore, we are going to use sine wave grating.

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$
$$a_0 = \frac{1}{L} \int_0^L f(x) dx$$
$$a_n = \frac{2}{L} \int_0^L f(x) \cos\left(\frac{n\pi x}{L}\right) dx$$
$$b_n = \frac{2}{L} \int_0^L f(x) \sin\left(\frac{n\pi x}{L}\right) dx$$





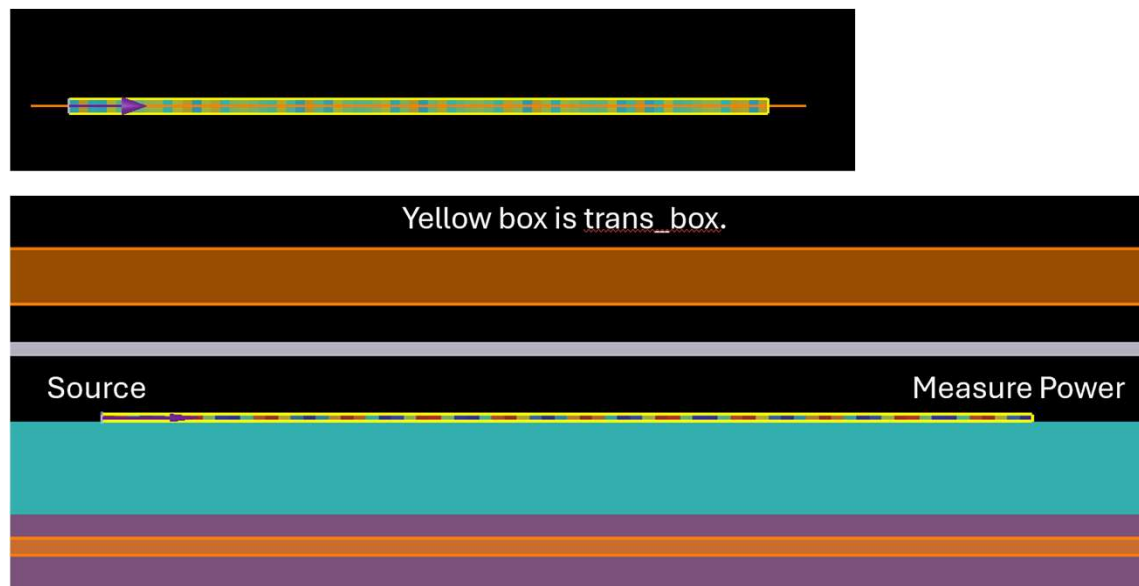
Waveguide Simulation-
Sin wave will be used



What would be the long
enough waveguide
length to let the light
out?

Transmission Box Setup to Measure the Power Out

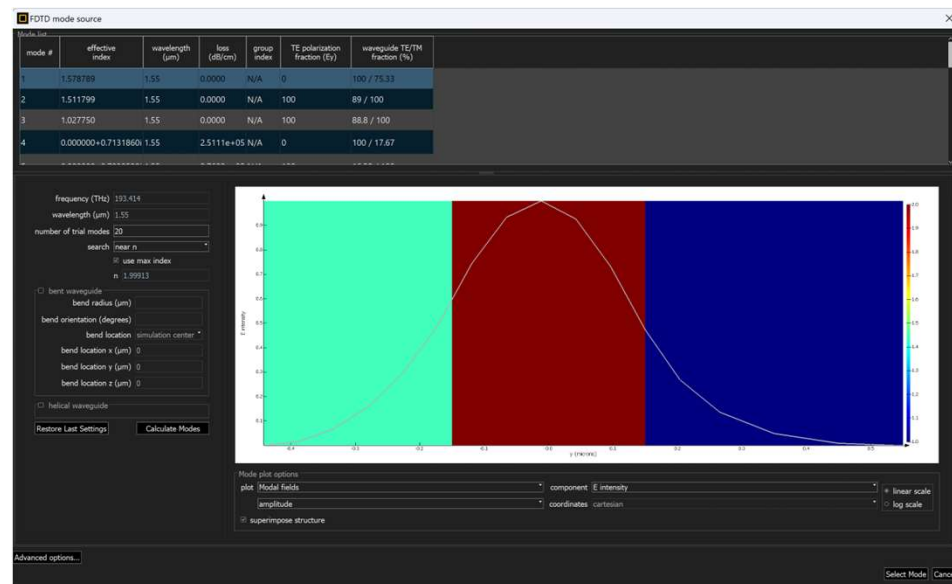
The purple arrow is the source; I measured the transmission at the end of the box



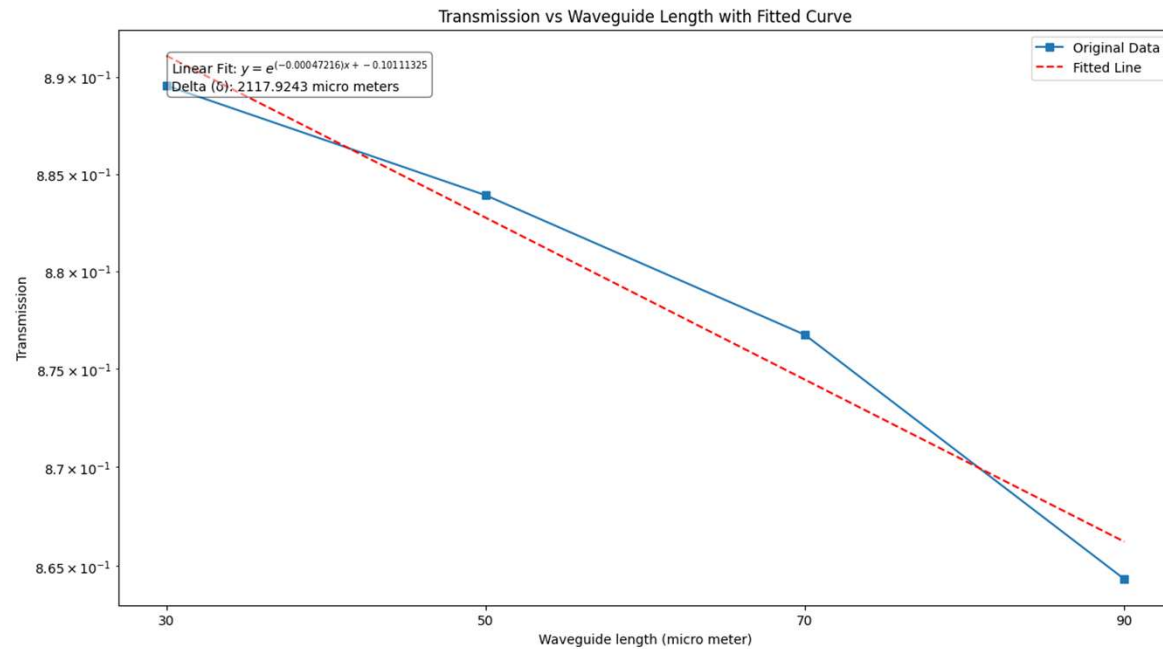
Correct Mode

The bell curve should be in mode #1.

In some instances, the bell curve distribution is positioned in mode 2 or alternative configurations. However, my selection of these configurations resulted in inaccurate transmission values.

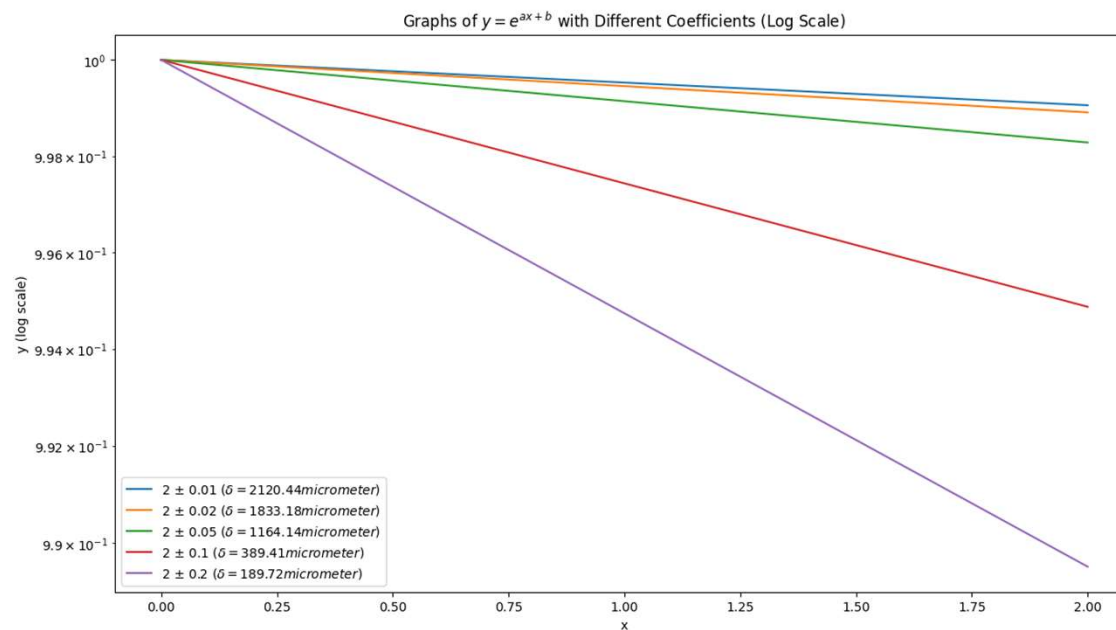


Plot and the fitting function for Single Δn Data



The parameter δ represents the decay length, defined as the distance over which the light intensity diminishes to $1/e$ of its initial value.

Losses Decrease as Δn Decreases



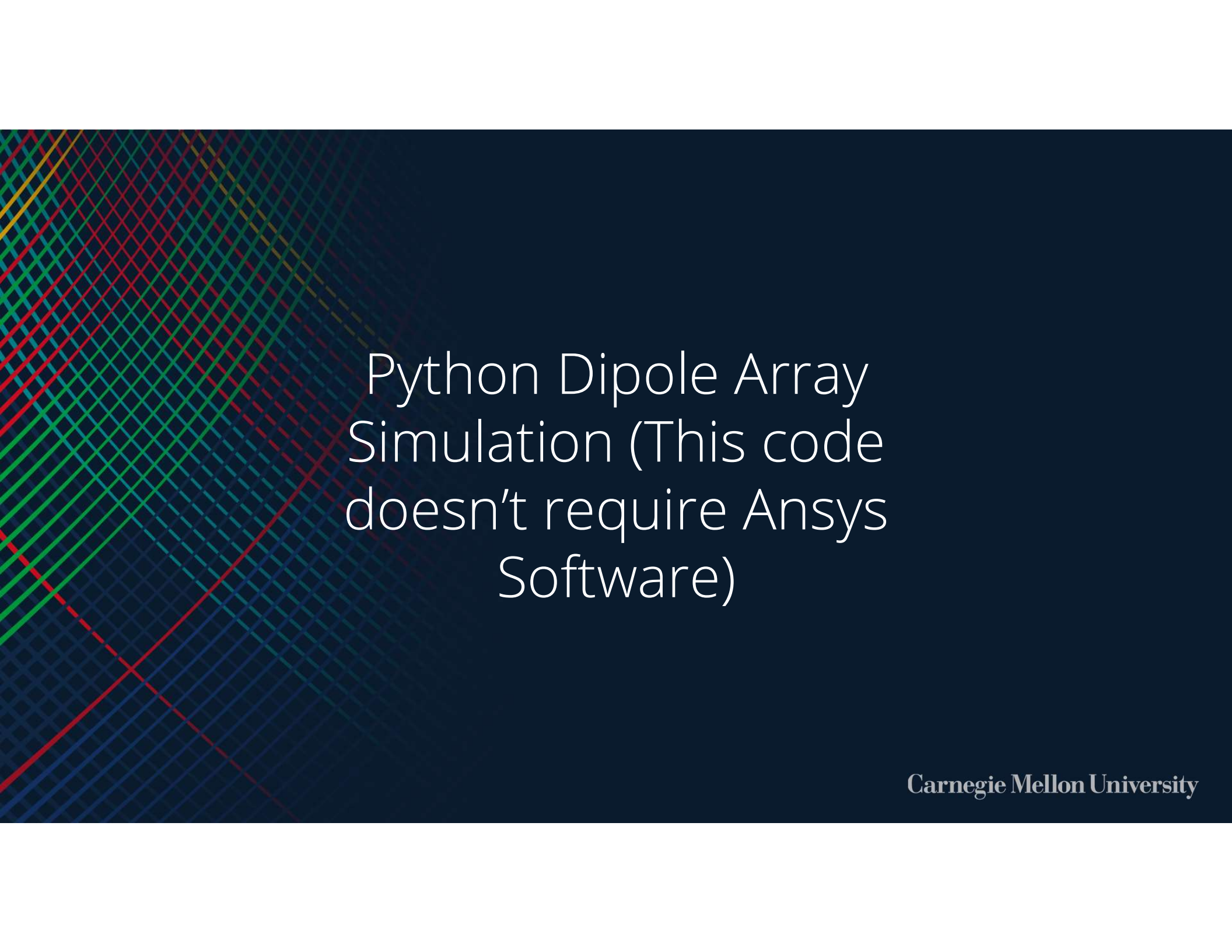
The parameter δ represents the decay length, defined as the distance over which the light intensity diminishes to $1/e$ of its initial value.



Scattering Length (Enough Length to let light out)

Δn	δ micrometer
0.01	2120.44
0.02	1833.18
0.05	1164.14
0.1	389.41
0.2	189.72

The parameter δ represents the decay length, defined as the distance over which the light intensity diminishes to $1/e$ of its initial value.



Python Dipole Array Simulation (This code doesn't require Ansys Software)

Algorithm

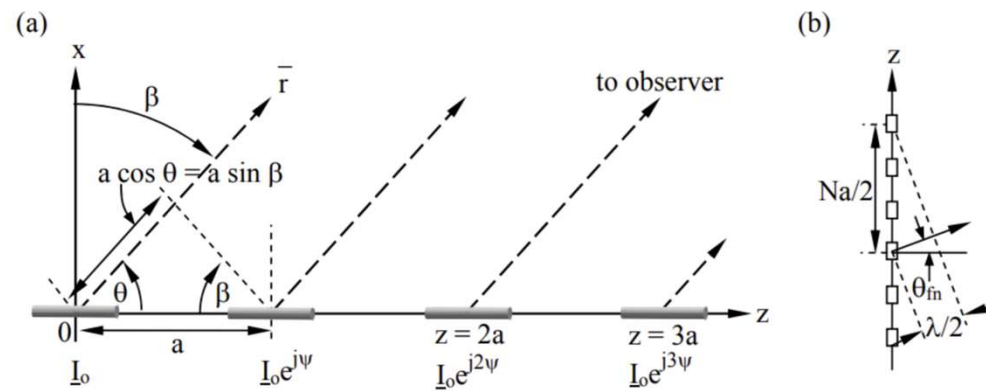


Figure 10.4.5: Uniform dipole array.

Consider the N-element array for Figure 10.4.5(a). The principal difference between these two-dipole cases and N-element uniform arrays lies in the array factor:

$$\underline{F}(\theta, \phi) = \sum_{i=1}^N I_i e^{-jk r_i} = I_0 e^{-jk r} \sum_{i=0}^{N-1} e^{j i \psi} e^{j k a \cos \theta} = I_0 e^{-jk r} \sum_{i=0}^{N-1} \left[e^{j(\psi + k a \cos \theta)} \right]^i \quad (10.4.7)$$

Algorithm

$$\underline{F}(\theta, \phi) = \sum_{i=1}^N \underline{I}_i e^{-jkr_i} = \underline{I}_0 e^{-jkr} \sum_{i=0}^{N-1} e^{j\psi} e^{jika \cos \theta} = \underline{I}_0 e^{-jkr} \sum_{i=0}^{N-1} \left[e^{j(\psi + ka \cos \theta)} \right]^i \quad (10.4.7)$$

The geometry illustrated in Figure 10.4.5(a) yields a phase difference of $(\psi + ka \cos \theta)$ between the contributions from adjacent dipoles.

Using the two identities:

$$\sum_{i=0}^{N-1} x^i = (1 - x^N) / (1 - x) \quad (10.4.8)$$

$$1 - e^{jA} = e^{jA/2} (e^{-jA/2} - e^{+jA/2}) = -2je^{jA/2} \sin(A/2) \quad (10.4.9)$$

(10.4.7) becomes:

$$\begin{aligned} \underline{F}(\theta, \phi) &= \underline{I}_0 e^{-jkr} \frac{1 - e^{jN(\psi + ka \cos \theta)}}{1 - e^{j(\psi + ka \cos \theta)}} \\ &= \underline{I}_0 e^{-jkr} \times \frac{e^{jN(\psi + ka \cos \theta)/2} \sin[N(\psi + ka \cos \theta)/2]}{e^{j(\psi + ka \cos \theta)/2} \sin[(\psi + ka \cos \theta)/2]} \end{aligned} \quad (10.4.10)$$

Since the element factor is independent of ϕ , the antenna gain has the form:

$$G(\theta) \propto |\underline{F}(\theta, \phi)|^2 \propto \frac{\sin^2[N(\psi + ka \cos \theta)/2]}{\sin^2[(\psi + ka \cos \theta)/2]} \quad (10.4.11)$$

Far Field Radiation Pattern Equation

The code on the right is the code I used to calculate the radiation pattern. It followed the equation below.

$$\begin{aligned}\underline{F}(\theta, \phi) &= \underline{I}_0 e^{-jkr} \frac{1 - e^{jN(\psi + ka \cos \theta)}}{1 - e^{j(\psi + ka \cos \theta)}} \\ &= \underline{I}_0 e^{-jkr} \times \frac{e^{jN(\psi + ka \cos \theta)/2} \sin[N(\psi + ka \cos \theta)/2]}{e^{j(\psi + ka \cos \theta)/2} \sin[(\psi + ka \cos \theta)/2]}\end{aligned}$$

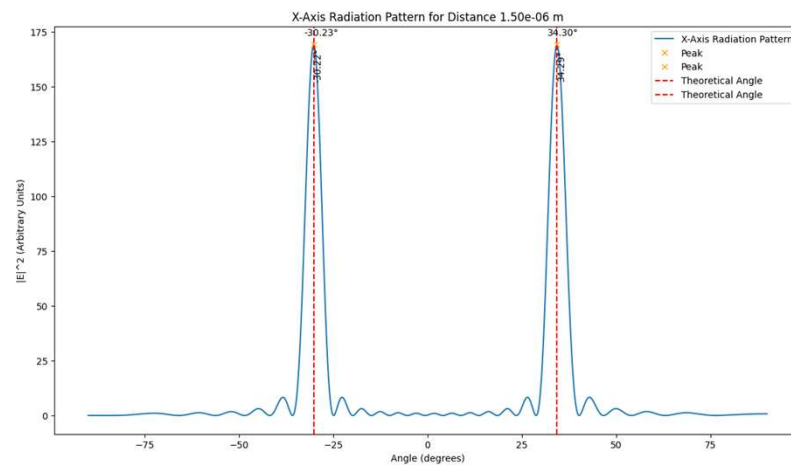
Since the element factor is independent of ϕ , the antenna gain has the form:

$$G(\theta) \propto |\underline{F}(\theta, \phi)|^2 \propto \frac{\sin^2[N(\psi + ka \cos \theta)/2]}{\sin^2[(\psi + ka \cos \theta)/2]}$$

```
def calculate_3d_radiation_pattern(angles_phi, angles_theta, k0, dx, dy, phase_x, phase_y, num_dipoles, num_arrays):  
    """  
    Calculates the 3D radiation pattern for an array of dipoles, considering both x and y axes adjustments.  
    Parameters:  
    - angles_phi (np.ndarray): 1D array of azimuthal angles in radians.  
    - angles_theta (np.ndarray): 1D array of elevation angles in radians.  
    - k0 (float): Wave number, defined as 2*pi/wavelength.  
    - dx (float): Distance between dipoles within an array along the x-axis.  
    - dy (float): Distance between arrays along the y-axis.  
    - phase_x (float): Phase difference between adjacent dipoles in the x-direction.  
    - phase_y (float): Additional phase shift applied in the y-direction.  
    - num_dipoles (int): Number of dipoles in one array along the x-axis.  
    - num_arrays (int): Number of arrays along the y-axis.  
    Returns:  
    - phi, theta (np.ndarray): Meshgrids of azimuthal and elevation angles for plotting.  
    - radiation_pattern (np.ndarray): Calculated radiation pattern intensity as a 2D array.  
    """  
    phi, theta = np.meshgrid(*[angles_phi, angles_theta])  
    x_argument = phase_x + k0 * dx * np.sin(theta) * np.cos(phi)  
    y_argument = phase_y + k0 * dy * np.sin(theta) * np.sin(phi)  
  
    x_numerator = np.sin(num_dipoles * x_argument / 2)  
    x_denominator = np.sin(x_argument / 2)  
    y_numerator = np.sin(num_arrays * y_argument / 2)  
    y_denominator = np.sin(y_argument / 2)  
  
    radiation_pattern = np.abs((x_numerator / x_denominator) * (y_numerator / y_denominator)) ** 2  
    return phi, theta, radiation_pattern
```

Result

The result matches perfectly with the theoretical values

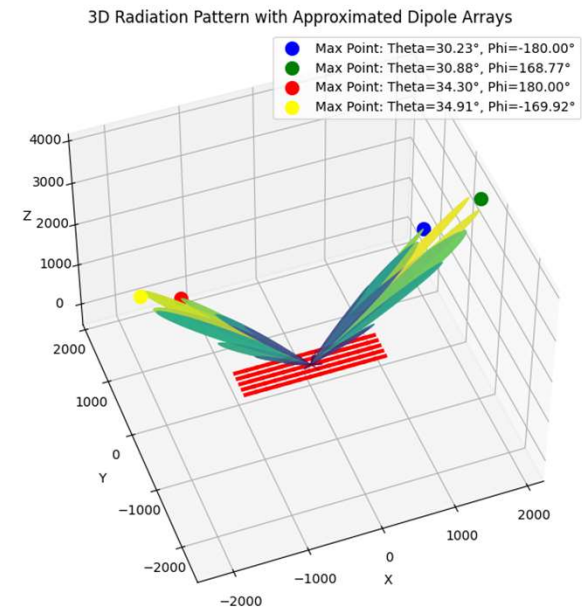


Expand the 2D Equation to a 3D Modeling Equation

This can be used to simplify the Ansys optical grating simulation. The red lines represent dipole arrays.

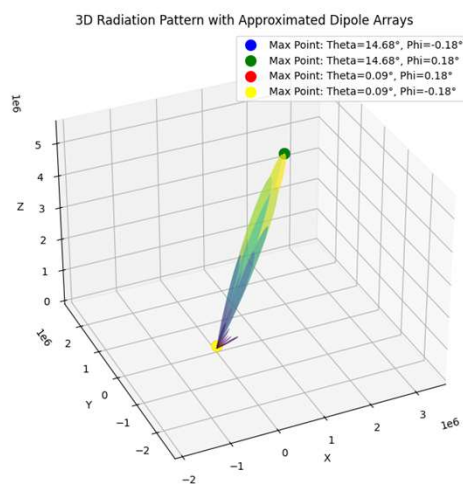
$$\frac{\sin\left(\frac{n_x(2\pi d_x \sin\theta \cos\phi + \alpha_x)}{2}\right)}{\sin\left(\frac{2\pi d_x \sin\theta \cos\phi + \alpha_x}{2}\right)} \frac{\sin\left(\frac{n_y(2\pi d_y \sin\theta \sin\phi + \alpha_y)}{2}\right)}{\sin\left(\frac{2\pi d_y \sin\theta \sin\phi + \alpha_y}{2}\right)} \frac{\sin\left(\frac{n_z(2\pi d_z \cos\theta + \alpha_z)}{2}\right)}{\sin\left(\frac{2\pi d_z \cos\theta + \alpha_z}{2}\right)}$$

The equation I used for 3D modeling

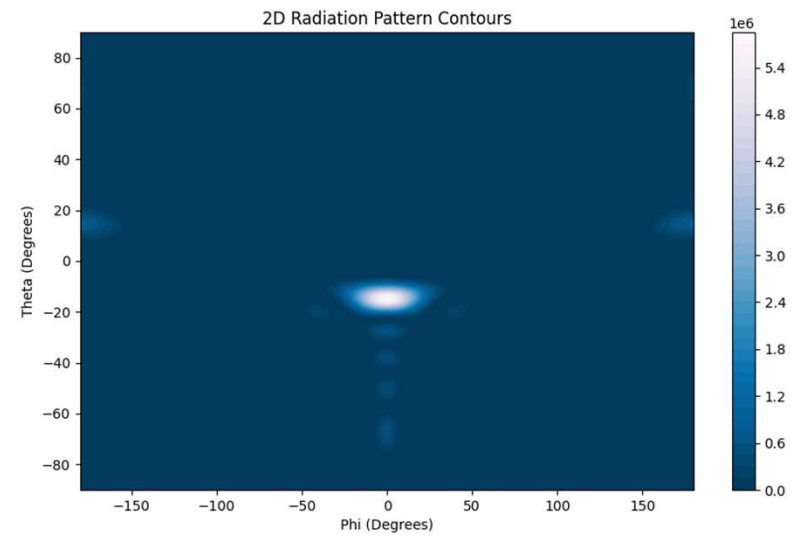


3D modeling

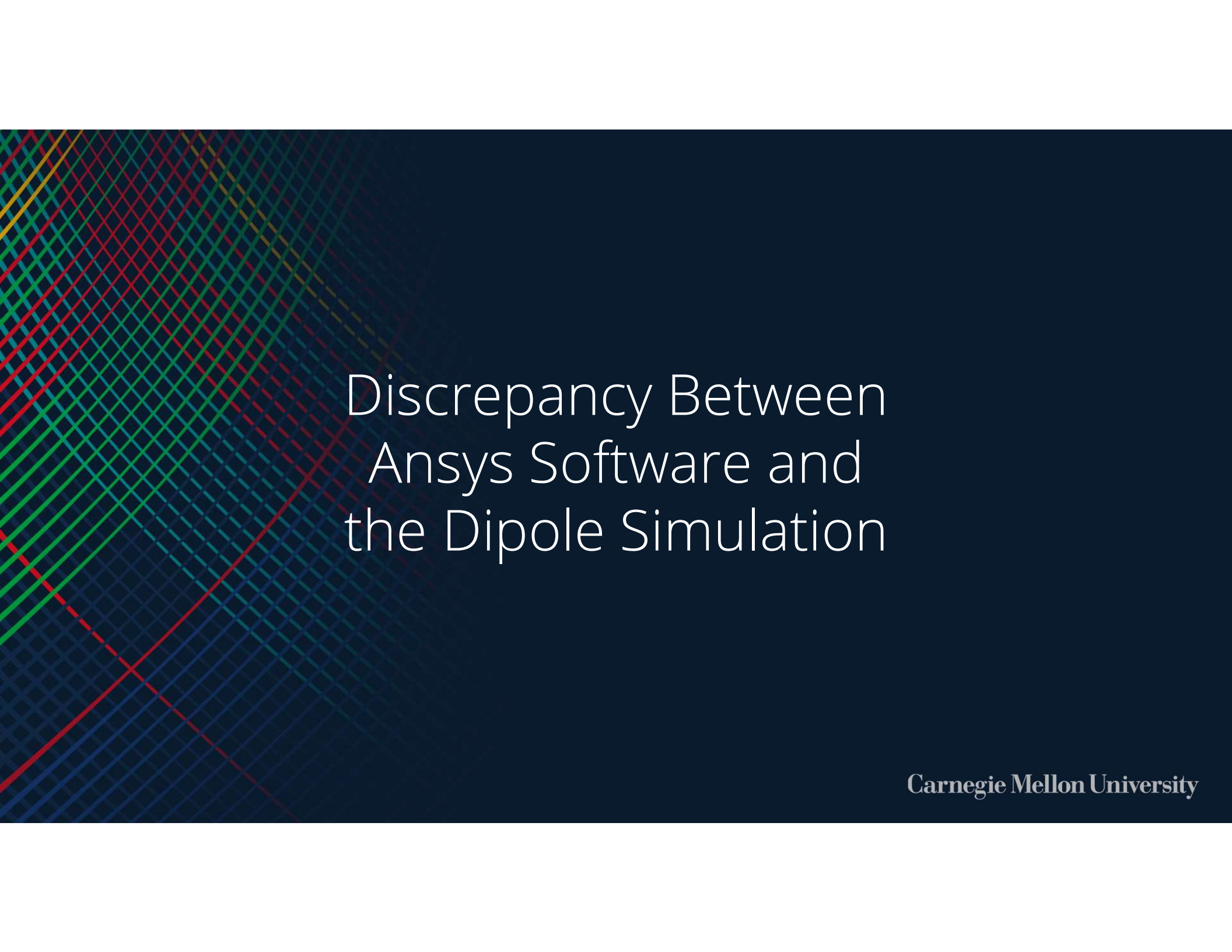
2D Contour Graph of 3D Radiation Pattern



3D modeling



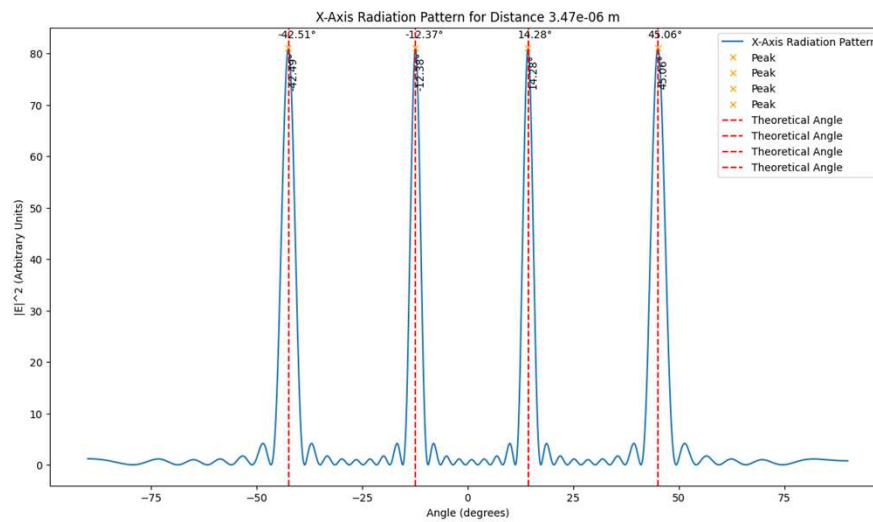
2D contour Graph



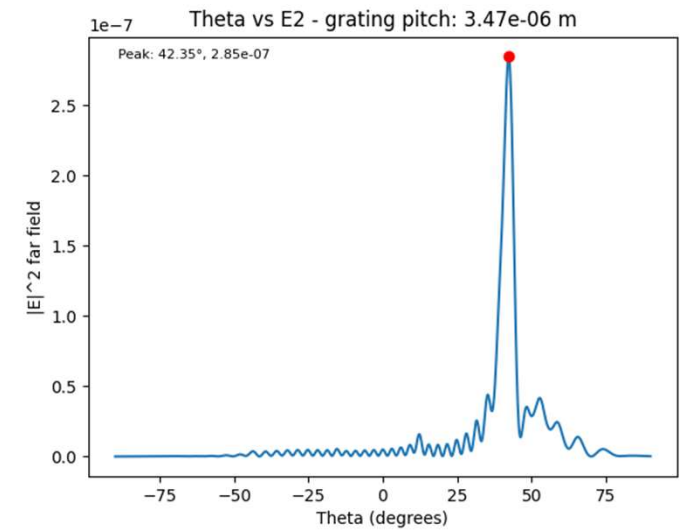
Discrepancy Between Ansys Software and the Dipole Simulation

Carnegie Mellon University

Anslys Simulation has One Major Peak



Dipole Simulation

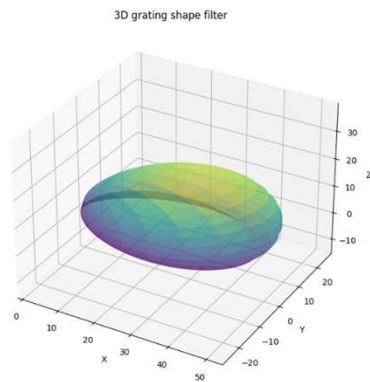


Anslys Simulation
Carnegie Mellon University

To match Dipole Simulation to Ansys Grating Simulation

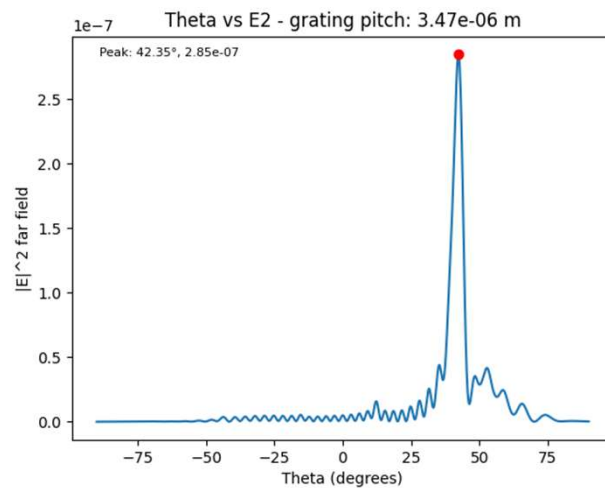
The grating shape filter is needed to fix the discrepancy in the Dipole simulation

```
grating_shape_filter = np.exp(-w * np.sin(angles_theta-lobe_theta))
```

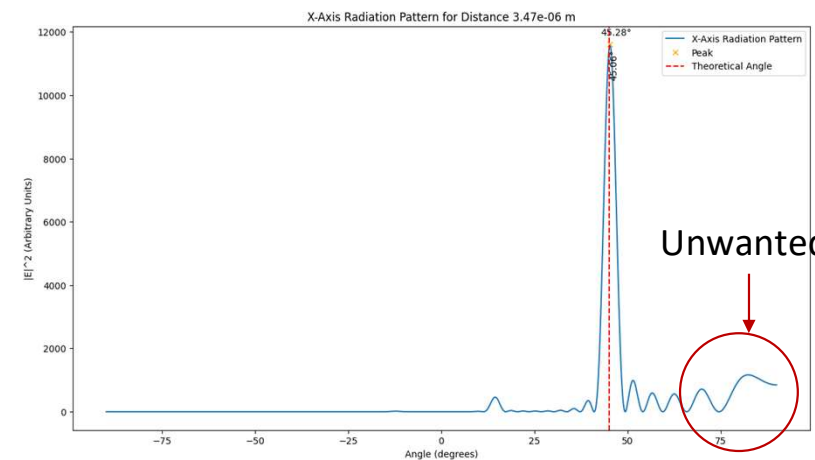


When Main Lobe Center is along x-axis

There is unwanted peak near 90 degree



Ansys Software

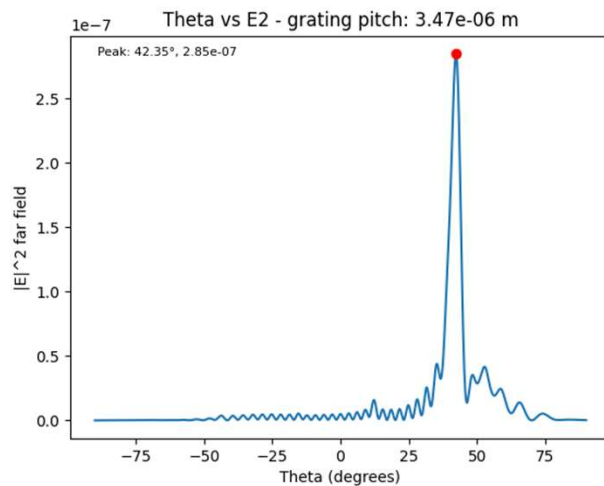


Dipole Array Simulation

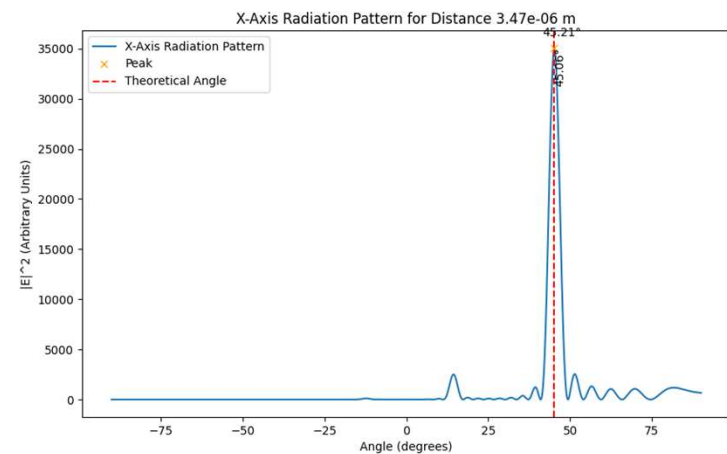
Carnegie Mellon University

Main Lobe Center at 15 Degrees

Now the result is very similar



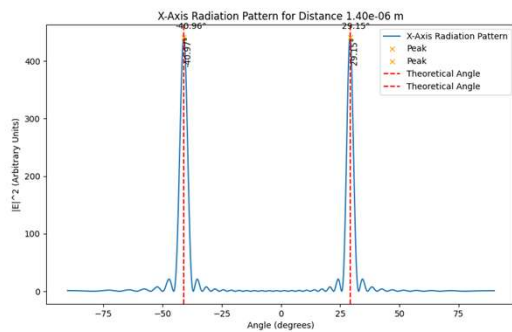
Ansys Simulation Graph



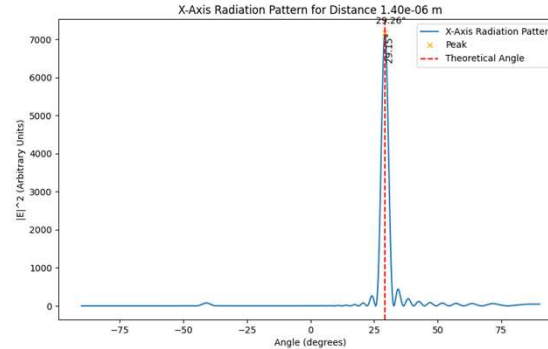
Dipole Array Simulation

The Grating Shape Filter is Affected by Light Propagation Direction.

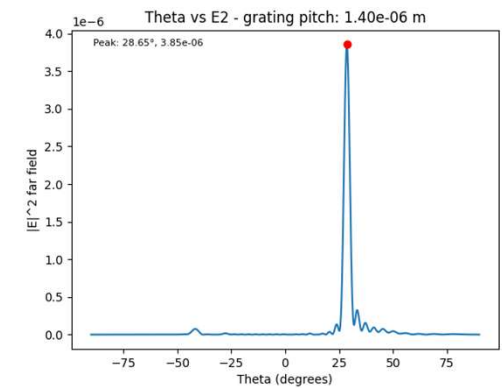
The middle graph is filtered by the grating shape filter (setting: $w=4$, $\text{lobe_theta} = \text{np.radians}(15)$), and the result looks similar to the Ansys grating simulator




Graph without grating shape filter



Dipole Array Simulation



Ansys Simulation Graph



Dipole Simulation Wave Lengths

Carnegie Mellon University



The Peak Positions

50 points between 1 and 2 micrometers for the grating pitch.

3 points between 1.5 and 1.6 micrometers for wavelength.

```
dx_range = np.linspace( start: 1e-6, stop: 2e-6, num: 50)
wavelength_range = np.linspace( start: 1.5e-6, stop: 1.6e-6, num: 3) # Wavelengths from 1.5 to 1.6 μm
data_dir = "radiation_pattern_data"

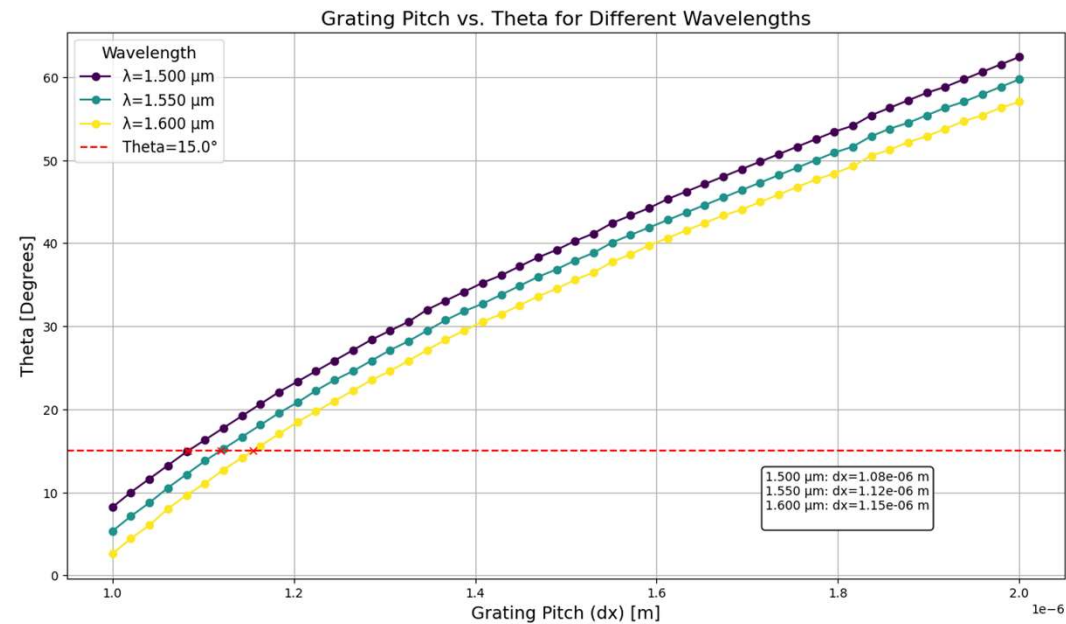
calculate_and_save_max_points(n_material, angles_phi, angles_theta, dx, dy, width, phase_y_range, num_arrays,
                             sampling_number, w, lobe_theta, dx_range, phase_y_range, wavelength_range, data_dir)
```



Other Conditions

- Refractive Index (n_{material}): Set to 1.63, this value represents the material's refractive index between the dipoles, affecting wave propagation.
- Array Span (x_{span}): The total span along the x-axis is 10 micrometers, determining the physical length of the dipole array.
- Number of Arrays: A single array configuration is analyzed.
- Physical Width of Dipole Array (width): Set to 10 micrometers, it simulates the actual size of each dipole and contributes to the overall radiation pattern.
- Distance Between Dipoles dx Range: Explored from 1 micrometer to 2 micrometers, varying the spacing to examine its impact on the radiation pattern.
- Wavelength Range: Analysis is conducted for wavelengths stretching from 1.5 micrometers to 1.6 micrometers, capturing the pattern's sensitivity to electromagnetic wavelength.

Visual Representation of Peak Positions

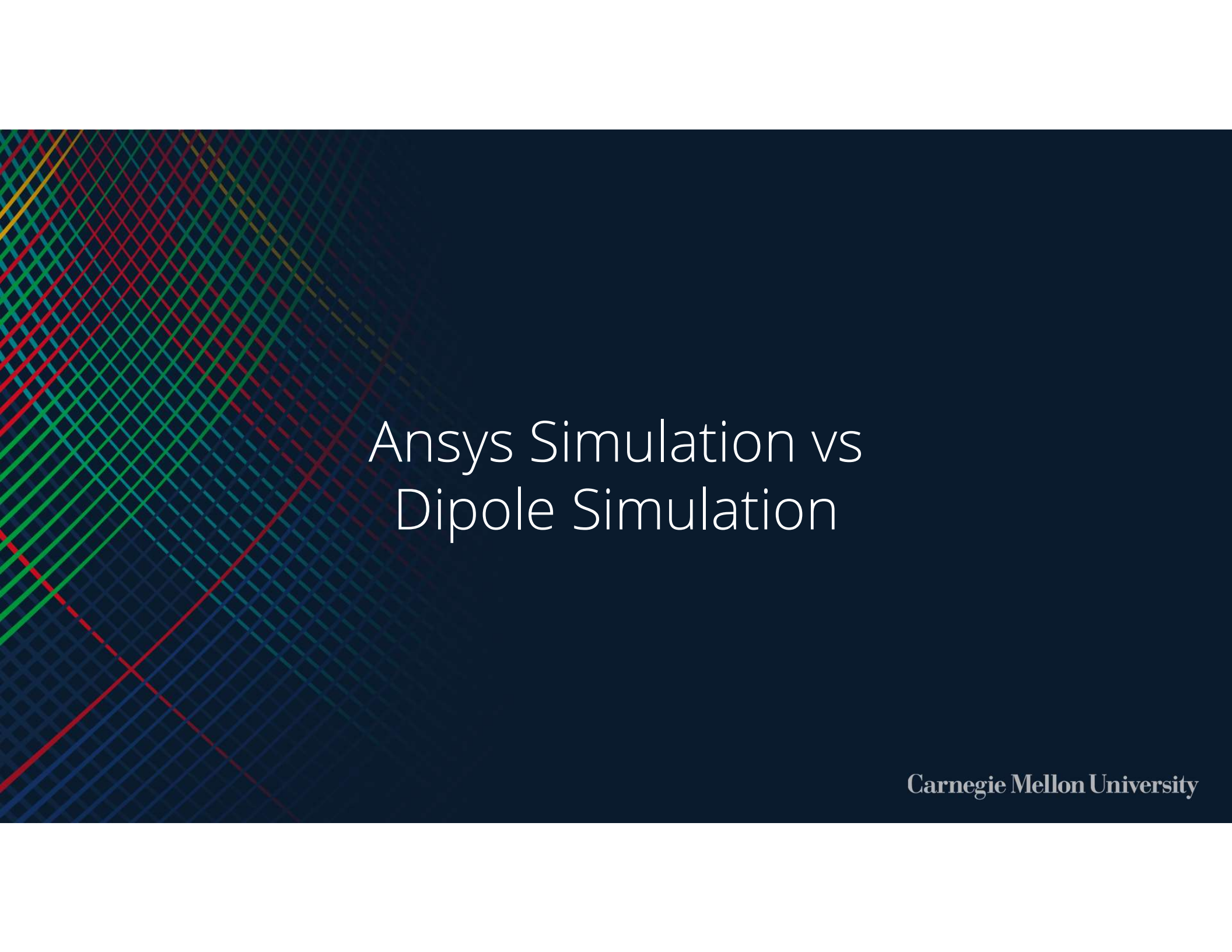




Grating Pitch Finder

The user can enter the target theta on Python code; then the code answers the grating pitch value

```
Enter the theta value (in degrees) you're interested in: 15
1.500 μm: dx=1.08e-06 m
1.550 μm: dx=1.12e-06 m
1.600 μm: dx=1.15e-06 m
```



Ansys Simulation vs Dipole Simulation

Carnegie Mellon University



Test on Ansys Simulation and Dipole Array Simulation

On the same condition:

1.59 for Refractive Index.

The grating pitch is $1.12\text{e-}6$

Other structure conditions are the same.

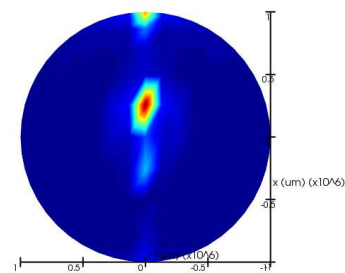
In Dipole Array Simulation:

The theta value is 15 degrees.

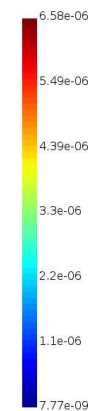


Anslys Simulated Result has a Peak at 15 Degrees

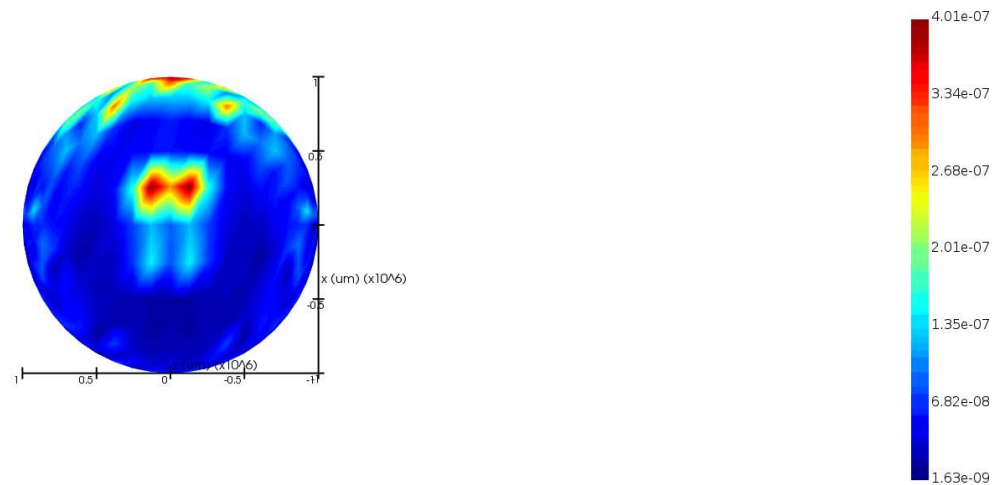
The Anslys graphing algorithm causes the hexagon pattern. (The image below was the result of 3 hours of running on a 13th-gen Intel(R) Core(TM) i7-1355U 1.70 GHz processor.)



Waveguide width: 10 micrometer

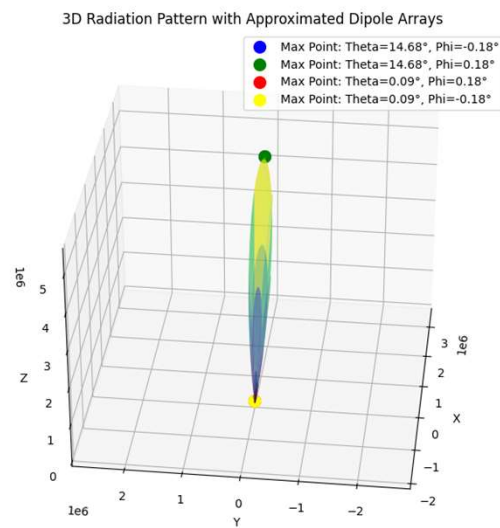


The Lobe Width gets Wider for Narrower Waveguide Width



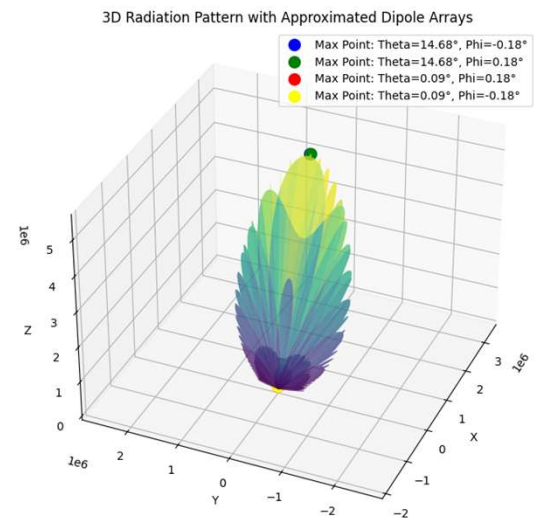
Waveguide width: 1 micrometer

Dipole Simulation shows a Similar Tendency



Waveguide width: 10 micrometer

Lobe width gets wider



Waveguide width: 1 micrometer



Answer for Ben's question

Hi Jim,

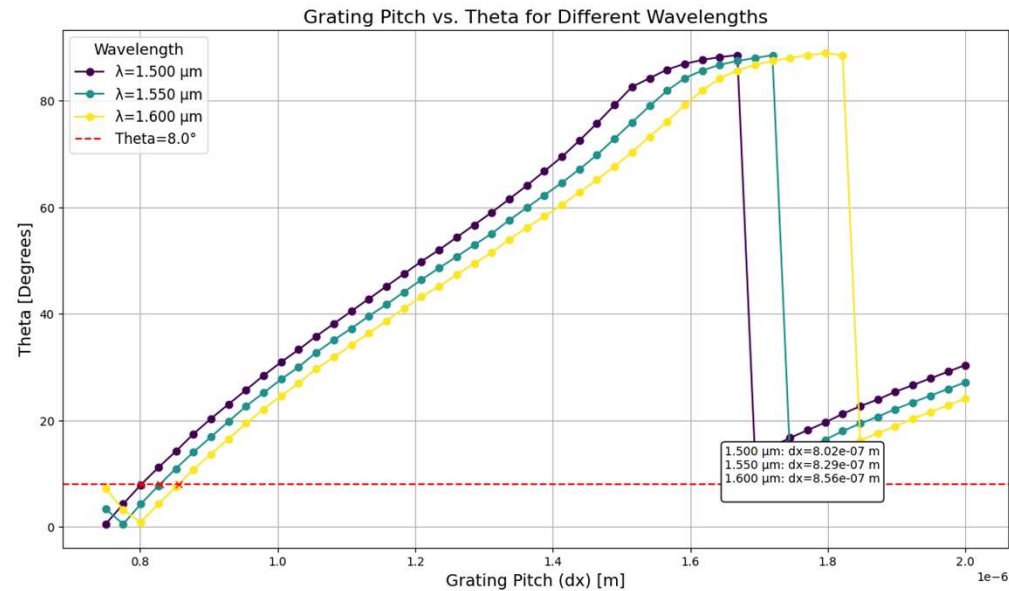
I did the quick calculation. My gratings are 50% the full stack, and 50% 100nm etched away from the core. So the neff of the full 300nm stack is 1.79 and the neff of the 200 nm core stack is 1.63. averaging those, I expect a grating neff of 1.711, pretty close to what Steve got. I think the discrepancy is because my actual pitch is $\sim 1.04\mu\text{m}$.

For the thicker case, I am not sure how much I will etch the core for the grating to work efficiently. If I stick with my 100nm etch process, the 2 indices will be 2.019 and 1.969, which average to 1.994. Your estimates are close.

Let me know if you need anything else.

Best,
Ben

$$n_{eff} = 1.994, \text{ expected } dx = 8.02 \sim 8.56e - 7$$





Summary

Ansys Simulation and Dipole Simulation can function similarly, though they have distinct operational requirements and results that are consistent. Ansys Simulation is computationally intensive and highly sensitive to mode selection, which can lead to inconsistent results if settings are not optimally chosen. In contrast, Dipole Simulation does not depend on mode selection; it simply computes results based on a specific equation, ensuring consistent outcomes.

GitHub link: <https://github.com/Sunghwan0112/Waveguide-Modulation-Project-ECE-M.S.>