

# Blackbody Simulations Usage Guide

---

Updated 8/20/2025

Jason Wang

# Outline

- Initial Setup
- Variables in the Code
- Simulation Usage
- Processing Data
- Visualizing Data
- Connecting Back to Geant4
- Future Plans
- Limitations

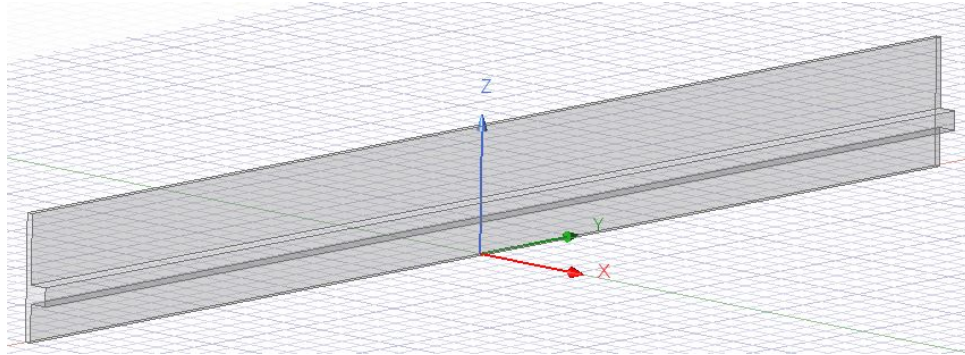
# Initial Setup

# Big Picture

- Simulate the response of a waveguide to an incoming photon, modeled as a plane wave in a vacuum
  - What is the probability the photon makes it to the other side of the waveguide without being absorbed?
  - Where on the outgoing face will the photon be emitted?
  - Toward which direction will the outgoing photon be emitted?
- Frequency dependent, polarization dependent, incoming angle dependent...
- Run Ansys HFSS simulations!

# Importing the Geometry

- On the HFSS main menu, open a new project, go to Modeler / Import and choose the .STL file to import
  - Alternatively, build your own model geometry
  - The ingoing and outgoing face should be planar and preferably (though not strictly necessarily) aligned with one of the XY, XZ, or YZ planes.



# Opening the Code

- Create an Anaconda environment with the following packages: pyaedt, scipy, pandas, numpy, seaborn
- On Github, make a copy of the latest version of the bbsim.py code
- Following the import statements, there is a list of parameters to be filled. We'll go through them one at a time

# Variables in the Code

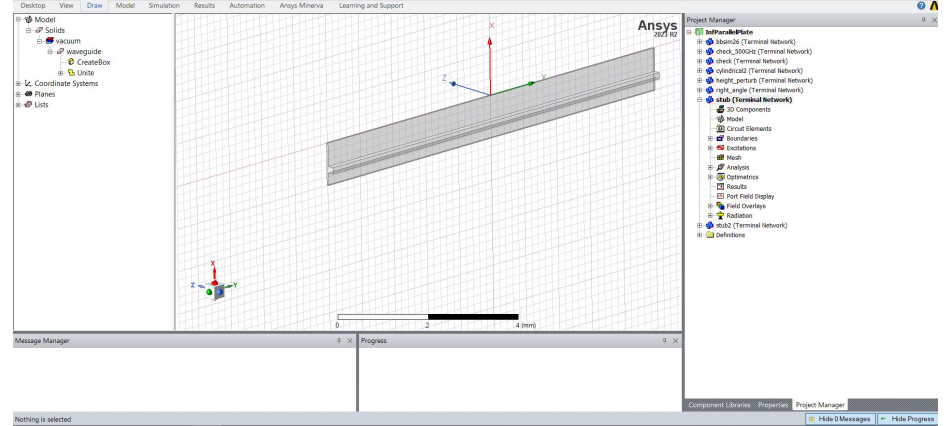
# Frequency

- Frequency: the frequency at which the simulation is run (GHz)
  - Specified via the command-line argument in terminal, e.g. `python bbsimlfreq.py 500`
    - Runs the simulation at 500GHz
- The simulation creates a COPY of the existing design in HFSS, where the new design name is {old\_design\_name}\_{frequency}GHz



# Project, Design, Feature Name

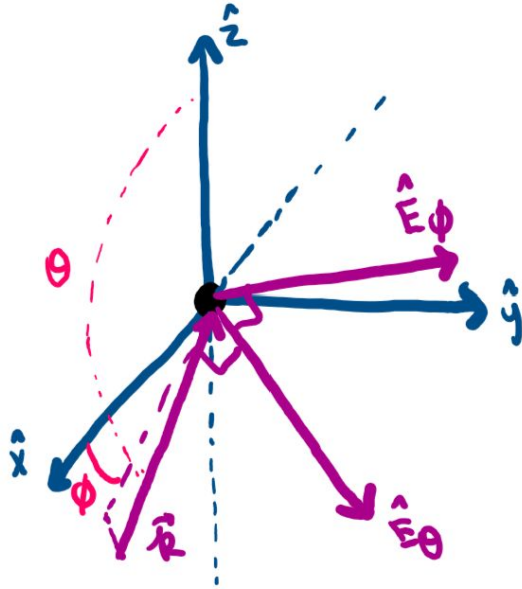
- project\_name, design name, feature name: Name of the project and design (project manager) and the feature to model (layout manager)
  - In the example
    - project\_name = InfParallelPlate
    - design\_name = stub
    - feature\_name = waveguide



# import\_from\_existing\_csv

- `import_from_existing_csv`: Whether to import the data file from an existing.csv.
  - If `import_from_existing_csv = True`, then rather than running the simulation from the start, the code runs a refined simulation from the imported csv file
    - Need to specify the waveguide csv (giving  $|S_{21}|$  and the electric field at the outgoing face as a fxn of ingoing angles) and the far-field csv (giving the far-field electric field as a fxn of ingoing angles)
    - The code automatically infers the frequency and polarization from the name of the file, so it should be of the form `{project_name}_{design_name}_{frequency}GHz_Ephi={0 or 1}/{waveguide or far_field}.csv`
      - $E_{\phi} = 0$  corresponds to polarization in the  $\theta$  direction;  $E_{\phi} = 1$  corresponds to polarization in the  $\phi$  direction (see conventions on next slide)
    - Following completion of refinement process, data is saved in `refined_{waveguide/far_field}.csv`
- Most of this time, we can safely set `import_from_existing_csv` to be False (default)

## Conventions for $E_\varphi$ and $E_\theta$



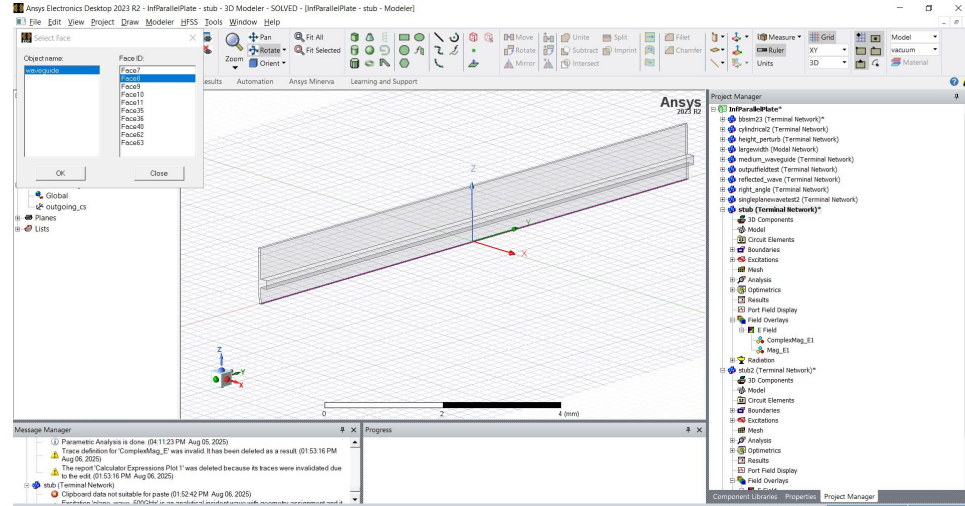
- $\theta$ -direction is  $(\cos \theta \cos \varphi, \cos \theta \sin \varphi, -\sin \theta)$
- $\varphi$ -direction is  $(-\sin \varphi, \cos \varphi, 0)$

# HFSS Analysis Parameters

- $E_i$ : Strength of ingoing electric field, used to calculate the ingoing power. For the purposes of the simulation, there is no reason to change it, since probability distributions are agnostic to the absolute strength of the electric field
- `max_delta_E`, `max_passes`, `num_cores`: HFSS parameters
  - `max_delta_E`: Criterion for HFSS simulation convergence. 0.02 is a good balance between speed and accuracy
  - `max_passes`: Maximum number of adaptive passes the HFSS simulation runs before termination. 10 is a good choice.
  - `num_cores`: Number of cores used in the HFSS simulation

# Boundary and Excitation Setup

- ingoing\_face\_id, outgoing\_face\_id: the numerical IDs of the faces where the plane wave should enter and exit
  - Where it says “Select” on the HFSS home screen, change “Object” to “Face”
  - Under “Object Name”, click on your model object
  - Identify which faces correspond to the ingoing and outgoing faces.
    - In the image, we see that in this geometry ingoing\_face\_id = 8
  - Simulation will initiate a plane wave excitation on the ingoing face
- conductivity: conductivity of remaining faces in S/m



# Outgoing Angle Sweeps

- `outgoing_face_cs_x`, `outgoing_face_cs_y`
  - Allows the user to redefine a coordinate system at the outgoing face
  - `outgoing_face_cs_x` is the direction of the x-axis in the new coordinate system, and is specified with coordinate with respect to the old coordinate system
    - Example: `outgoing_face_cs_x = [0, 0, 1]` means that the x-axis of the new coordinate system is the old z-axis. `outgoing_face_cs_y = [0, 1, 0]` means that the y-axis is left unchanged
- `rad_theta_lower`, `rad_theta_upper`, `rad_phi_lower`, `rad_phi_upper`
  - The code will scan the outgoing angles for  $\theta$  from `[rad_theta_lower, rad_theta_upper]` and  $\phi$  from `[rad_phi_lower, rad_phi_upper]`. These angles are defined relative to the new coordinate system

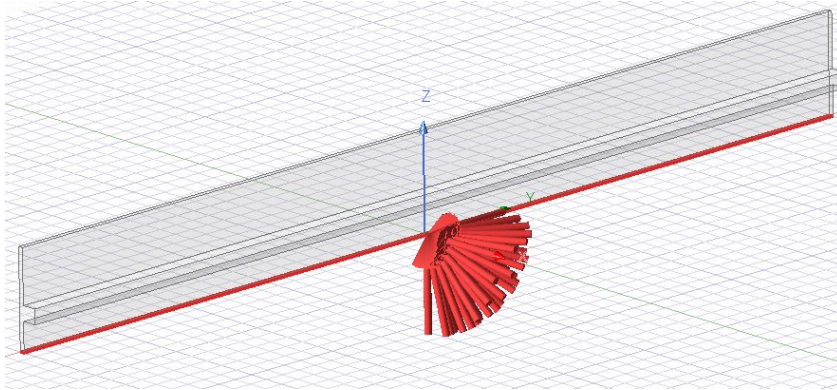
# Electric Field at Outgoing Face

- `manual_field`, `outgoing_face_boundary`, `outgoing_face_field_resolution`:
  - If `manual_field` is set to `False`, HFSS will infer the appropriate sampling resolution, but it is coarse (coordinates relative to global CS)
  - If `manual_field` is set to `True` and `outgoing_face_boundary` is set to `None`, the bounding region for the outgoing face is inferred, but it only works for rectilinear faces (coordinates relative to outgoing CS)
  - `outgoing_face_boundary` can also be manually specified for non-rectilinear geometries (coordinates relative to outgoing CS)

```
manual_field = True
outgoing_face_boundary = None # Or specify ([x1, y1, z1], [x2, y2, z2]), e.g.([0, -5, -0.025], [0, 5, 0.025]) Boundary of the outgoing face
outgoing_face_field_resolution = ["0mm", "0.1mm", "0.001mm"] # Resolution in outputting the electric field at the outgoing face
```

# Ingoing Angle Sweeps

- $i\_theta\_lower$ ,  $i\_theta\_upper$ ,  $i\_phi\_lower$ ,  $i\_phi\_upper$ 
  - The code will scan the ingoing angles for  $\theta$  from  $[i\_theta\_lower, i\_theta\_upper]$  and  $\phi$  from  $[i\_phi\_lower, i\_phi\_upper]$ . These angles are defined relative to the global coordinate system, and the angle is defined relative to the tail of the ingoing wavevector
- $i\_theta\_step$ ,  $i\_phi\_step$ : the coarseness of the initial sweep, in degrees. 0.8 degrees is a good starting point.

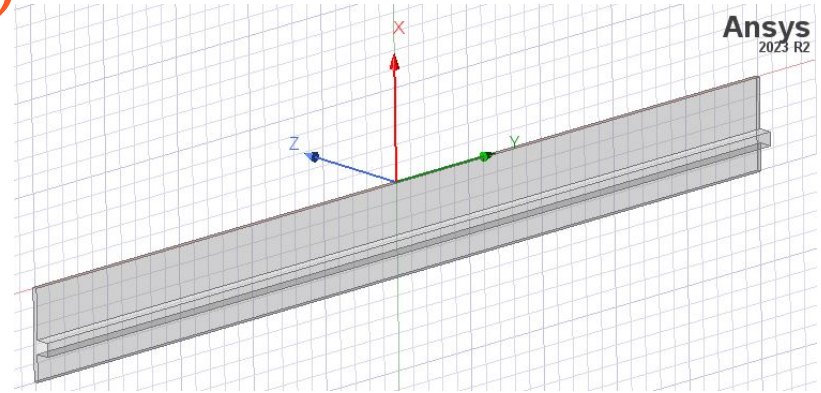


\*Corresponds to  $i\_theta\_lower = 90$ ,  
 $i\_theta\_upper = 180$ ,  $i\_phi\_lower = 0$ ,  
 $i\_phi\_upper = 90$ ,  $i\_theta\_step = 15$ ,  
 $i\_phi\_step = 15$



# Outgoing Angle Sweeps (cont.)

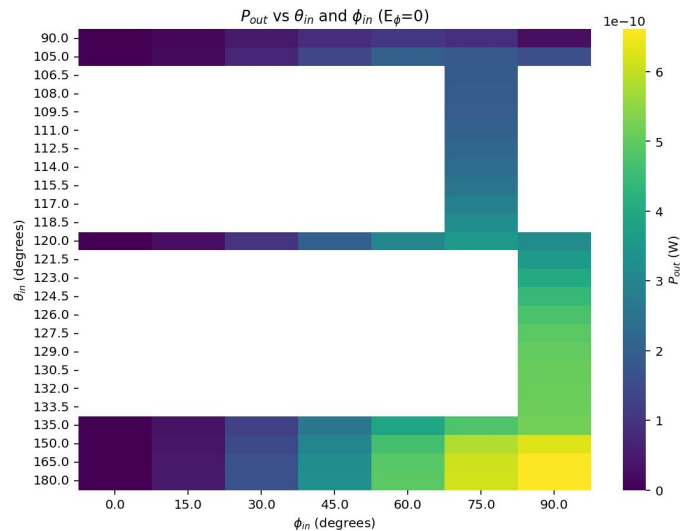
- a, b
  - a is the length scale [mm] of the dimension coinciding with the sweep over outgoing  $\phi$  (outgoing angles defined with respect to newly defined coordinate system)
  - b is the length scale [mm] of the dimension coinciding with the sweep over outgoing  $\theta$
  - The wider the slit in the  $\phi/\theta$  direction, the narrower the diffraction pattern
- fineness
  - $\lambda/(a \cdot \text{fineness})$  is the angular resolution (in radians) of the outgoing sampling. This follows from diffraction theory. fineness = 10 is a good start
- minimum\_coarseness, maximum\_coarseness
  - Hard bounds for the minimum and maximum angular resolutions to sample the far-field radiation pattern from



\*Corresponds to  $a = 10$  mm (the sweep over  $\phi$ , i.e. a sweep over outgoing angles in the XY plane) and  $b = 0.05$  mm (a sweep over outgoing angles in the XZ plane)

# Sweep Mode

- sweep = “discrete” or sweep = “adaptive”
  - Discrete sweep has no refinement process, and just runs the simulation and exports the data
  - Adaptive sweep will further look at the data, find the maximum point-to-point outgoing power difference, and find the most important regions to refine based on that
- max\_difference:
  - The maximum point-to-point difference in the outgoing power under which HFSS declares that the simulation has concluded successfully. max\_difference = 0.015 is a good start, and one can interpolate data with this 1.5% difference as a baseline



\*Illustration of adaptive sweep:  
large jumps between  $(\phi, \theta) = (75, 120)$  and  $(75, 105)$  are patched.  
Similar, large jumps between  $(\phi, \theta) = (90, 135)$  and  $(90, 120)$  are patched

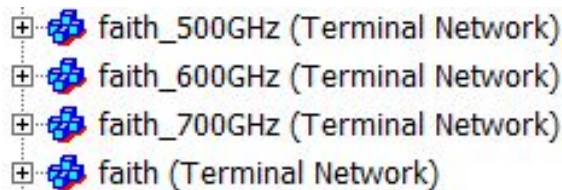
# Simulation Usage

# No Need to Manually Input Anything Except the Code Variables

- The code is designed such that there is no need to set up analysis sweeps, parametric radiation boundaries, calculator expressions etc.
  - Simply change the variables in the python code and run via command line
- Additionally, if a mistake is made and the simulation needs to be rerun for a given geometry, RENAME the design, change the design\_name variable in the code, and run via command line again

# Run Code Via Terminal

- Accepts one frequency at a time via command line argument
  - E.g. `python bbsimlfreq.py 500` runs the simulation at 500GHz
- The simulation creates a COPY of the existing design in HFSS, where the new design name is `{old_design_name}_{frequency}GHz`,
  - Makes it easier to run the different designs in parallel using HPC



A screenshot of the HFSS project hierarchy. On the left, a vertical dashed line with square nodes indicates a collapsed tree view. To the right of this line, four components are listed, each preceded by a blue cube icon with a red cross. The components are: `faith_500GHz (Terminal Network)`, `faith_600GHz (Terminal Network)`, `faith_700GHz (Terminal Network)`, and `faith (Terminal Network)`.

# Other Useful Things

- Due to the extensive sweeps, the code may take a while to run. Be patient!
- Calculating and exporting the data from HFSS takes a large amount of CPU, so be wary of that
- If, in the middle of the simulation, you need to stop the simulation, you can `KeyboardInterrupt`, which will then ask you if you want to save the existing data to a csv file
  - By default, the simulation only saves the initial coarse sweep and the final, completed refined sweep to a .csv file. You can save a .csv file in the middle of a simulation via this method
- Make sure there are no additional variables defined in the HFSS GUI! This can lead to errors in exporting far field data due to a technicality

# Workflow Summary

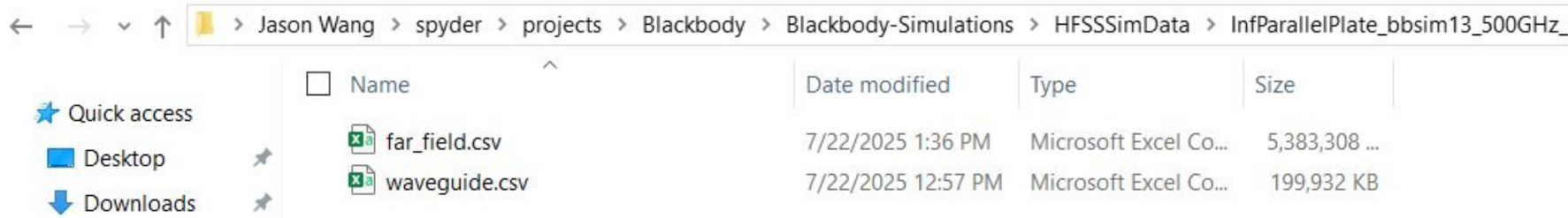
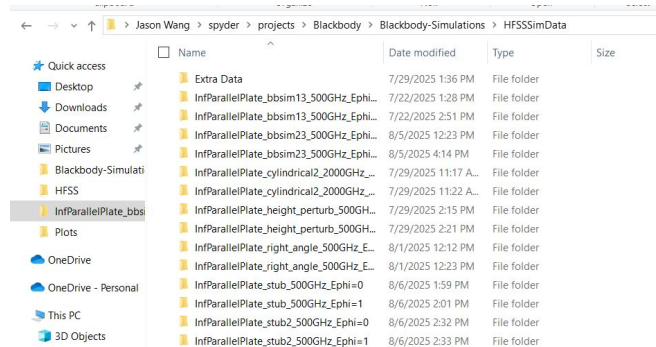
- Import/build new geometry
- Define the variables relevant to the geometry, sweep, etc. desired
- Run simulation via command line
- Process data (next section)

# Processing Data



# File Structure

- All data files from the simulation will be stored in the same directory where bbsim.py is stored in a folder called HFSSSimData, organized by frequency and polarization
- Inside each folder, there are 2 csv files:
  - (refined)\_far\_field.csv
  - (refined)\_waveguide.csv
  - Whether or not there are files that have “refined” in front depends on whether adaptive or discrete sweep is chosen



## (refined)\_far\_field.csv

- (refined)\_far\_field.csv: gives the electric field at outgoing angles specified as a function of ingoing angles specified
  - Gives complex-valued (real + i\*imag)  $E_\phi$  and  $E_\theta$  (with conventions on the polarizations in the  $\phi$  and  $\theta$  direction as given before in a diagram)

	A	B	C	D	E	F	G	H	I	J
1	Freq	Ephi	IWavePhi	IWaveThet	Phi	Theta	rEphi_real	rEphi_imag	rEtheta_re	rEtheta_imag
2	500GHz	0	0	90	-90	0	0.000136	0.000133	2.26E-08	1.29E-07
3	500GHz	0	0	90	-89.6565	0	0.000136	0.000133	8.39E-07	9.29E-07
4	500GHz	0	0	90	-89.3129	0	0.000136	0.000133	1.65E-06	1.73E-06
5	500GHz	0	0	90	-88.9694	0	0.000136	0.000133	2.47E-06	2.53E-06
6	500GHz	0	0	90	-88.6259	0	0.000136	0.000133	3.29E-06	3.33E-06
7	500GHz	0	0	90	-88.2823	0	0.000136	0.000133	4.10E-06	4.13E-06
8	500GHz	0	0	90	-87.9388	0	0.000136	0.000133	4.92E-06	4.93E-06
9	500GHz	0	0	90	-87.5952	0	0.000136	0.000133	5.73E-06	5.73E-06
10	500GHz	0	0	90	-87.2517	0	0.000136	0.000133	6.55E-06	6.53E-06
11	500GHz	0	0	90	-86.9082	0	0.000136	0.000133	7.36E-06	7.32E-06
12	500GHz	0	0	90	-86.5646	0	0.000136	0.000133	8.18E-06	8.12E-06
13	500GHz	0	0	90	-86.2211	0	0.000136	0.000133	8.99E-06	8.92E-06

# (refined)\_waveguide.csv

- (refined)\_waveguide.csv: gives the electric field at different (X, Y, Z) locations specified as a function of ingoing angles specified
  - Gives complex-valued (real + i\*imag)  $E_x$ ,  $E_y$ , and  $E_z$
  - Also gives OutgoingPower, and hence  $|S_{21}|^2 = \text{OutgoingPower}/\text{IngoingPower}$

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Freq	Ephi	IWavePhi	IWaveThet	OutgoingP	IngoingPov	X	Y	Z	Ex_real	Ey_real	Ez_real	Ex_imag	Ey_imag	Ez_imag		
2	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.005	0.001	0	0	0	0	0		
3	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.0049	0.001	0.091969	-0.0055	0.000138	3.97E-05	1.42E-05	4.08E-06	
4	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.0048	0.001	0.185312	-0.01488	0.000173	-2.24E-05	1.69E-05	-3.47E-06	
5	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.0047	0.001	0.278649	-0.03699	0.000389	0.000566	4.55E-05	6.55E-05	
6	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.0046	0.001	0.360452	-0.07104	8.54E-05	-4.87E-06	-9.66E-05	2.76E-05	
7	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.0045	0.001	0.432306	-0.13332	0.002187	0.000367	-0.00203	0.000645	
8	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.0044	0.001	0.473082	-0.2008	-0.00016	-2.85E-05	0.000153	-4.78E-05	
9	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.0043	0.001	0.494102	-0.27664	-0.00039	-0.00013	0.000694	0.000109	
10	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.0042	0.001	0.493856	-0.35018	0.009855	0.003069	-0.0174	-0.00231	
11	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.0041	0.001	0.478974	-0.40913	-0.0075	-0.0003	-0.04549	-0.00048	
12	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.004	0.001	0.412717	-0.43842	-0.00296	0.000181	-0.01245	7.66E-05	
13	500GHz	0	0	90	1.67E-10	6.64E-10		0	-0.0039	0.001	0.355649	-0.44797	0.000886	-0.00023	0.004575	0.000105	

# Visualizing Data

# plot.py

- Also found in [Github repo](#)
  - Many useful plots:
    - Heatmap of the outgoing power as a function of ingoing angles (angular sweep)
    - Outgoing power as a function of ingoing angles, with one ingoing angle ( $\varphi$  or  $\theta$ ) held fixed
    - Magnitude of the complex electric field at the outgoing face as a function of ingoing angles
    - Heatmap of the far-field electric field as a function of ingoing angles and outgoing angles (angular sweep)
    - Far-field electric field as a function of ingoing angles and outgoing angles, with one outgoing angle ( $\varphi$  or  $\theta$ ) held fixed

# Connecting Back to Geant4

# Interpolation

Finally, let us discuss methods for interpolating simulation data for reintroduction into Geant4. The electromagnetic outputs – scalar transmission coefficient  $|S_{21}|^2$ , complex electric field at the outgoing face, and far-field electric field – are sampled over a multi-dimensional parameter space of frequency, input and output angles, and spatial coordinates. For each of the two orthogonal polarization directions, the input parameters and corresponding outputs of the three relevant quantities are shown in table 1.

---

Quantity	Input Variables	Output Variables
$ S_{21} ^2$	$f, \theta_{in}, \phi_{in}$	$ S_{21} ^2(f, \theta_{in}, \phi_{in})$
E-field at Exit Face	$f, \theta_{in}, \phi_{in}, x, y, z$	$\mathbf{E}(f, \theta_{in}, \phi_{in}, x, y, z)$
Far-field E-field	$f, \theta_{in}, \phi_{in}, \theta_{out}, \phi_{out}$	$\mathbf{E}(f, \theta_{in}, \phi_{in}, \theta_{out}, \phi_{out})$

Table 1: Input and output variables for the three quantities of interest. The output variables directly correspond to the probability distributions described in section 3.1, so this table describes the input and output variables of the desired lookup table.

Classical approaches such as radial basis function interpolation, tensor-product splines, and kriging provide smooth, physically consistent estimates but can be computationally expensive for large datasets. Reduced-order modeling mitigates this by projecting the fields onto a small set of modal basis functions (e.g., waveguide or spherical harmonics) and interpolating only the modal coefficients. A particularly effective strategy combines this with machine learning: the fields are first compressed via proper orthogonal decomposition (POD) into parameter-independent, dominant modes, and neural networks are trained to map input parameters (frequency, angles, and spatial coordinates) to the corresponding modal coefficients [7]. The full fields can then be efficiently reconstructed from these coefficients, yielding smooth, accurate, and physically consistent predictions across the multi-dimensional parameter space. Once the fields are obtained, they can be sampled from as described section 3.1.

# interpolate.py

- Also found in [Github repo](#)
- INCOMPLETE draft of interpolation scheme described on the previous slide



# Determining the Probability Distributions

The mechanism for calculating the necessary output probability distributions is modularized over three steps.

- (i) Firstly, an incident plane wave is propagated through the waveguide via an incident plane wave excitation in HFSS. The time-averaged power incident on the waveguide can be calculated as

$$P_{in} = \int_I |\mathbf{S}_{in}| dA, \quad (52)$$

where  $\int_I$  refers to an integral over the outgoing face and  $\mathbf{S}_{in} = \frac{1}{2} \frac{\mathbf{E} \times \mathbf{B}}{\mu_0}$  indicates the time-averaged Poynting vector of the incoming wave. Notice that in this case, we are not interested in the flux of the Poynting vector ( $\int_I \mathbf{S}_{in} \cdot d\mathbf{A}$ ) through the input face, because the propagation of the wave is not necessarily aligned with the direction of the waveguide.

From the simulation, we obtain a value of the time-averaged transmitted power by integrating the Poynting vector  $\mathbf{S}_{out}$  over the outgoing face, i.e.

$$P_{out} = \int_O |\mathbf{S}_{out}| dA, \quad (53)$$

where  $\int_O$  refers to an integral over the input face. As for the input face, we are concerned not about the flux of the Poynting vector through the outgoing face but rather the total power coming out of the output face regardless of directionality. This is because photons leaving the waveguide reenter free space, and we wish to account for the full power carried away, even if it is not perfectly aligned with the waveguide axis.

For the first stage, then, the probability that the photon propagates to the other end of the waveguide without being reflected or absorbed is given as

$$|S_{21}|^2 = \frac{P_{out}}{P_{in}}, \quad (54)$$

where  $|S_{21}|^2$  represents the forward transmission coefficient.

## Determining the Probability Distributions (cont.)

- (ii) Secondly, given that the photon arrives at the output face, the probability distribution of output locations on the output face to feed back to Geant4 is given by  $P(x, y) \propto |\mathbf{E}(x, y)|^2$ , i.e. the probability for the photon to be emitted from any given point on the output face is proportional to the electric field squared at that point. We use the complex magnitude squared of the electric field ( $|\mathbf{E}(x, y)|^2 = |E_x|^2 + |E_y|^2 + |E_z|^2$ , where  $E_x, E_y, E_z \in \mathbb{C}$ ), rather than the magnitude squared of the instantaneous electric field ( $|Re[\mathbf{E}(x, y)]|^2$ ) because we are interested in the time-averaged intensity of the electric field over the outgoing face independent of the phase of the wave. Then, the numerical probabilities can be calculated from the discretized electric field positions  $\{(x_i, y_i)\}$  as

$$P(x, y) = \frac{|\mathbf{E}(x, y)|^2}{\sum_{(x_i, y_i)} |\mathbf{E}(x_i, y_i)|^2}. \quad (55)$$

- (iii) Thirdly, given that the photon arrives at the output face, the probability distribution of output wavevectors (parametrized by  $\theta_{out}$  and  $\phi_{out}$ ) to feed back to Geant4 can be found by analyzing the far-field radiation pattern, which is calculated in HFSS as the Fourier transform of the electric field at the output face. We can then use the same method as equation 55, i.e. from the discretized output angles  $\{(\theta_i, \phi_i)\}$  as

$$P(\theta, \phi) = \frac{|\mathbf{E}(\theta, \phi)|^2}{\sum_{(\theta_i, \phi_i)} |\mathbf{E}(\theta_i, \phi_i)|^2}. \quad (56)$$

where again  $|\mathbf{E}(\theta, \phi)|^2$  denotes the complex magnitude squared of the far-field electric field at outgoing angles  $\theta$  and  $\phi$ . Given an outgoing angle parametrized by  $(\theta, \phi)$ , the polarization of the outgoing photon can be inferred through the far-field radiation pattern as well:

$$\hat{e}_{out} = \frac{\mathbf{E}(\theta, \phi)}{|\mathbf{E}(\theta, \phi)|}. \quad (57)$$

# Determining the Probability Distributions (cont.)

The main difficulty in creating these lookup tables lies in the need to evaluate many combinations of input parameters, including frequency, wavevector direction, and input electric field polarization. One simplification is to use symmetry to simulate only a subset of the possible input angles, as shown in figure 3. Another simplification is to simulate only two orthogonal polarization directions. We denote the two orthogonal directions by  $\hat{\mathbf{E}}_\theta$  and  $\hat{\mathbf{E}}_\phi$ , where the unit vector in the  $\theta$  direction is  $(\cos \theta \cos \phi, \cos \theta \sin \phi, -\sin \theta)$ , and the unit vector in the  $\phi$  direction is  $(-\sin \phi, \cos \phi, 0)$ . A depiction of the two polarization states is given in figure 4. Results for any arbitrary polarization can then be reconstructed by linearly combining the outcomes from this basis due to linearity of Maxwell's equations. In particular, if the ingoing electric field polarization is  $\hat{\mathbf{e}}_{in} = E_\theta \hat{\mathbf{E}}_\theta + E_\phi \hat{\mathbf{E}}_\phi$ , then  $\mathbf{E}(x, y)$  and  $\mathbf{E}(\theta, \phi)$  become

$$\begin{aligned}\mathbf{E}(x, y) &\rightarrow E_\theta \mathbf{E}_{E_\theta=1, E_\phi=0}(x, y) + E_\phi \mathbf{E}_{E_\theta=0, E_\phi=1}(x, y) \\ \mathbf{E}(\theta, \phi) &\rightarrow E_\theta \mathbf{E}_{E_\theta=1, E_\phi=0}(\theta, \phi) + E_\phi \mathbf{E}_{E_\theta=0, E_\phi=1}(\theta, \phi)\end{aligned}\tag{58}$$

where the subscripts on  $\mathbf{E}$  indicate one of the two orthogonal polarization directions. Equations 55, 56, and 57 then proceed as before with the replacements given in equation 58.

## postprocess.py

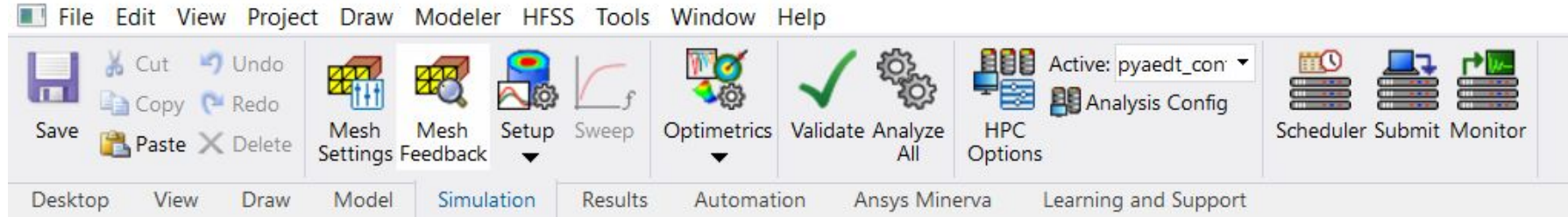
- Also found in [Github repo](#)
- Implements the procedure described on the previous few slides
  - Given photon frequency, polarization vector, incoming angles, determine transmission probability, probability of emission from different points, probability of emission into different outgoing angles
- Can be used after interpolating and evaluating onto a set of discrete points

# Future Plans

# Different Geometries

- Run simulations on actual geometries
  - So far, we've run it on standard rectangular waveguide, cylindrical waveguide, height-perturbed waveguide, right-angle waveguide, stub-filter waveguide
  - Run on SLAC fridge geometries

# HFSS HPC (High-Performance Computing)



- Parallelize sweep over many frequencies
- Needs to run HFSS at many different frequencies (many different instances of HFSS)

# Limitations



# Limitations of the Current Simulation

- Runtime
- Non-planar ingoing/outgoing surfaces are definitely a concern
  - On the input side, non-planar surfaces are tolerable and the incident plane wave should work as expected
  - On the output side, non-planar surfaces makes defining a CS to export electric fields more difficult
    - Use the `manual_fields = False` option?
- Need to manually specify many parameters to run