Tutorial: 3D Wavelength Demultiplexer

In this tutorial, we will walk through the process of designing a 3D wavelength demultiplexer using Spins, the process is very similar to the 2D example. The first step is preparing a GDS file which captures the geometry of the simulation domain and material stack. This process is most easily done in KLayout with the Spins technology file. After the geometry GDS file is made, we then provide a few more simulation parameters before running our optimization.

Prepare GDS File

Opening KLayout

Once Klayout is installed, we open the application using the terminal to initialize global variables. In a Linux environment the command is:

```
$ klayout -e -rd dx=0.04 -rd fcon=0.120
```

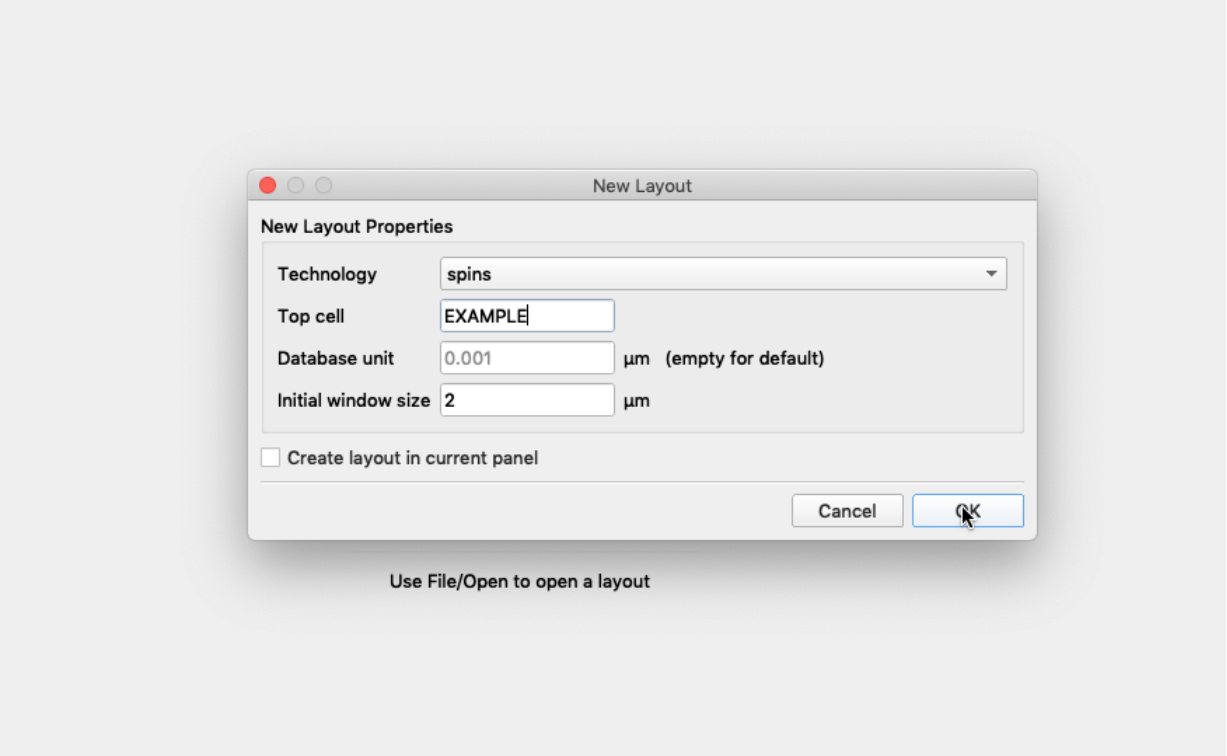
and on a Mac:

```
$ open -a klayouteditor --args -rd dx=0.04 -rd fcon=0.120
```

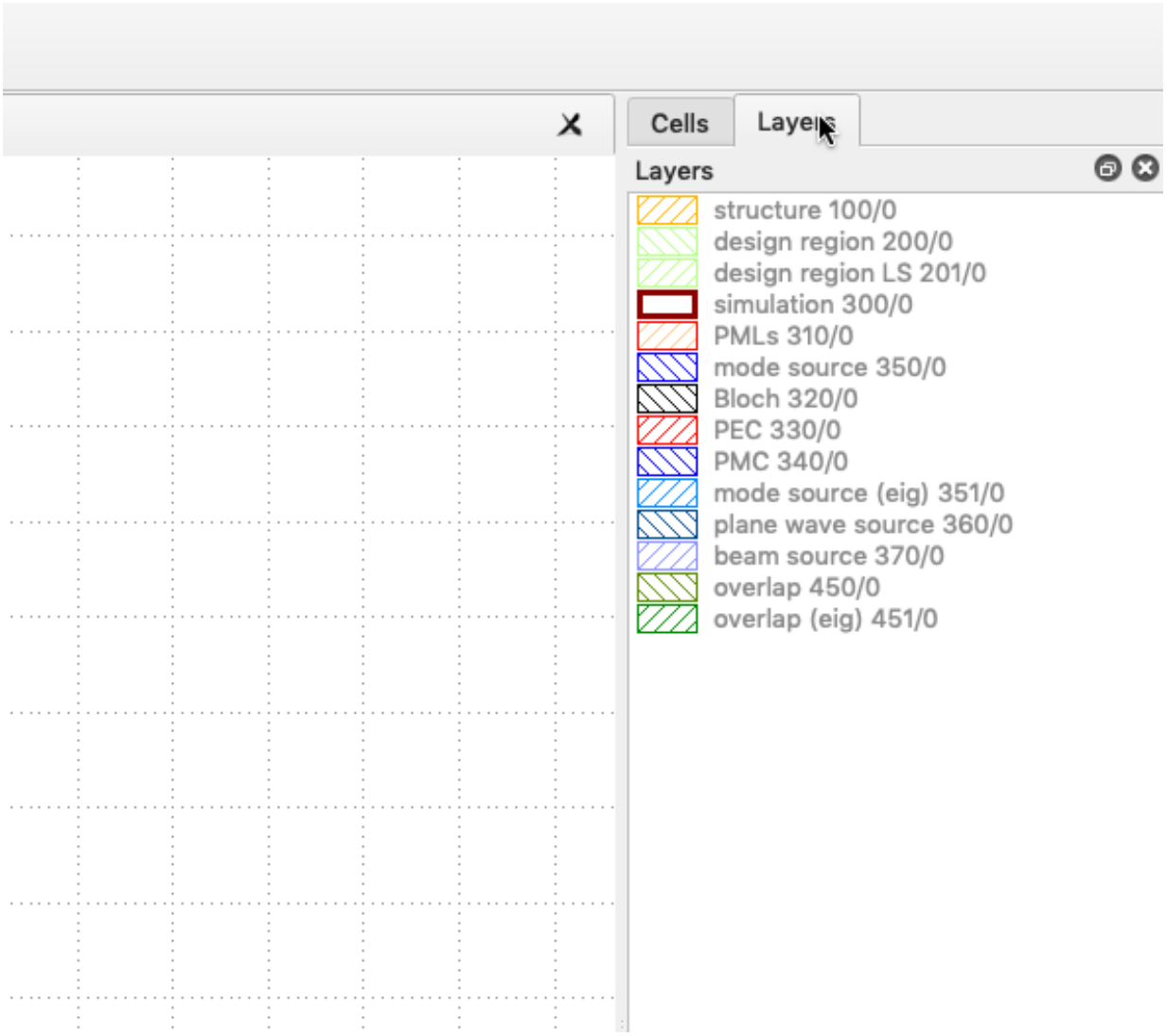
The purpose of this command is to set the value of the grid unit, *dx*, and the minimum fabrication constraint, *fcon*, which other steps in the GDS generation depend on.

Create Layout

Once we have KLayout opened, we first create a new layout with the Spins technology through **File > New Layout**. Select *spins* as the technology and choose a name for the *Top cell* and then select **OK**.

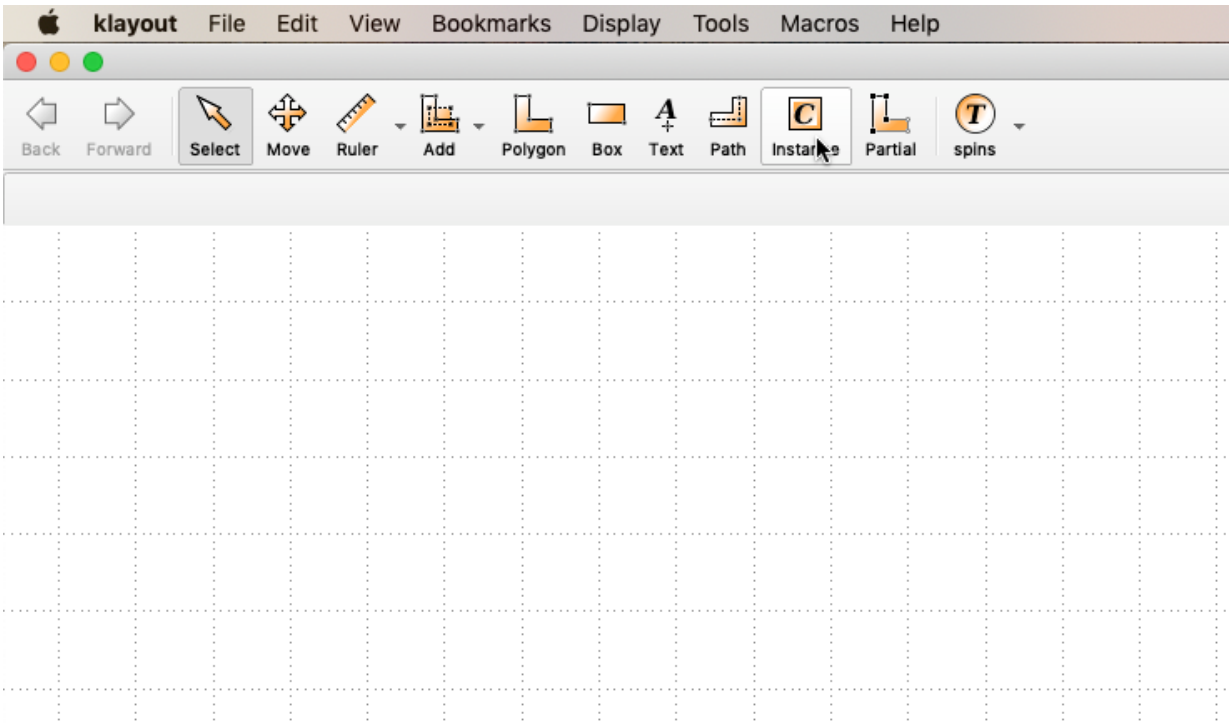


Once the new layout has been created, you should be greeted with a list of empty layers which will contain information related to the geometry of the simulation domain and material stack. We begin populating these layers by first creating cell instances for the simulation domain and design region.

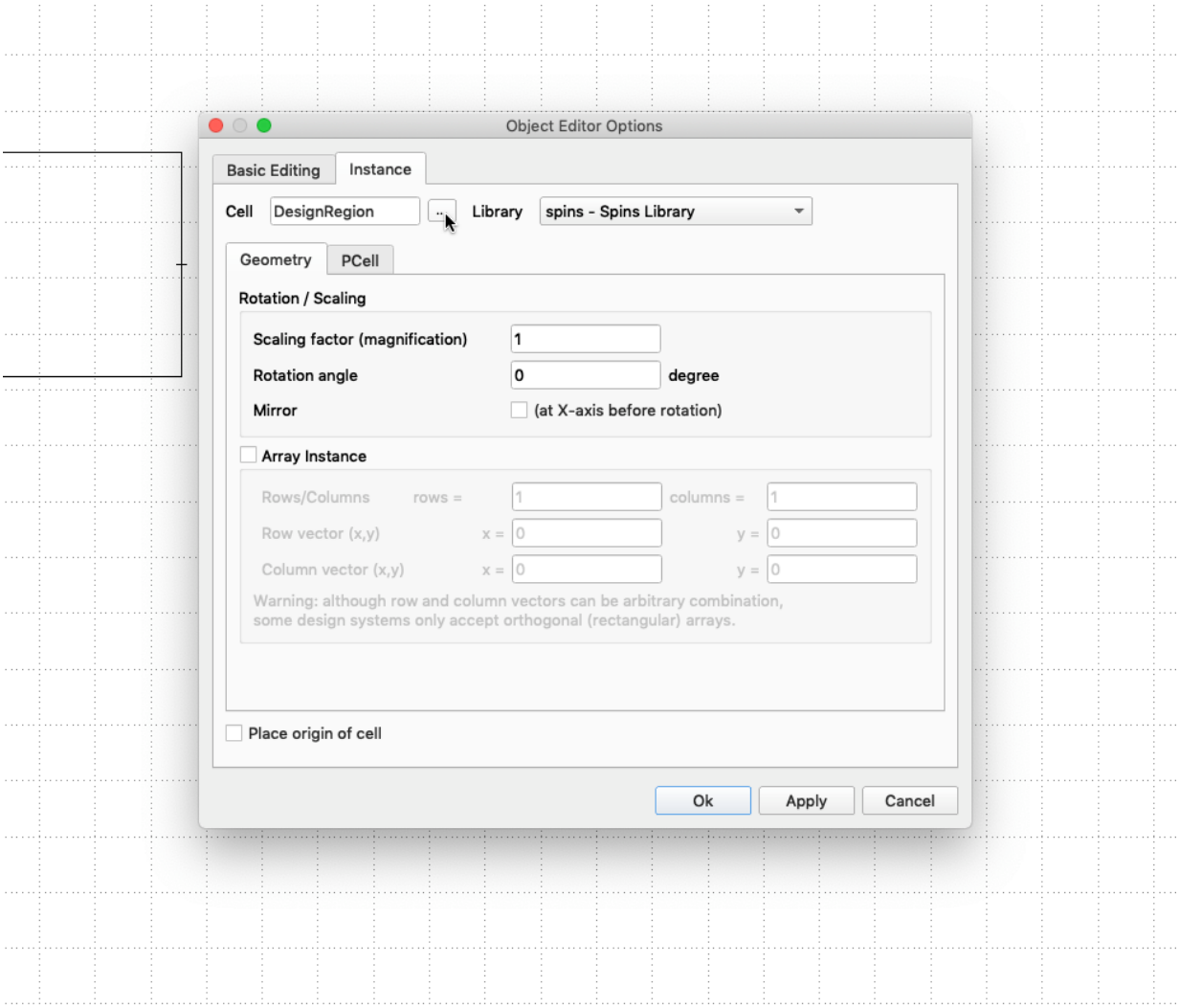


Define Simulation

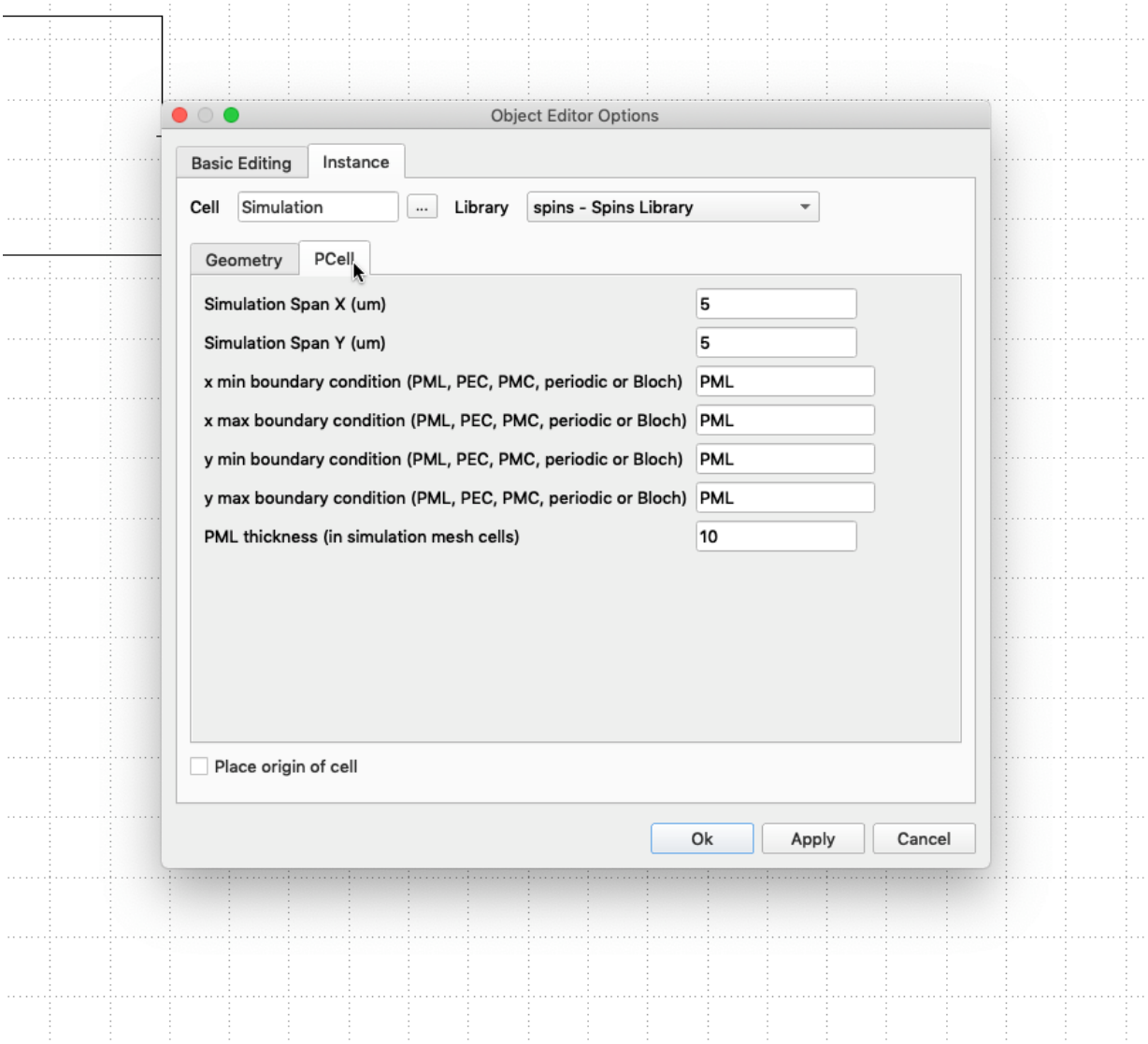
Now that we have a layout created we can begin to add object instances to our file. To do so, click **Instance** at the top toolbar.



Choose the Spins Library from the **Library** dropbar and click the **...** button beside the *Cell* field to see the pre-made Spins cells.



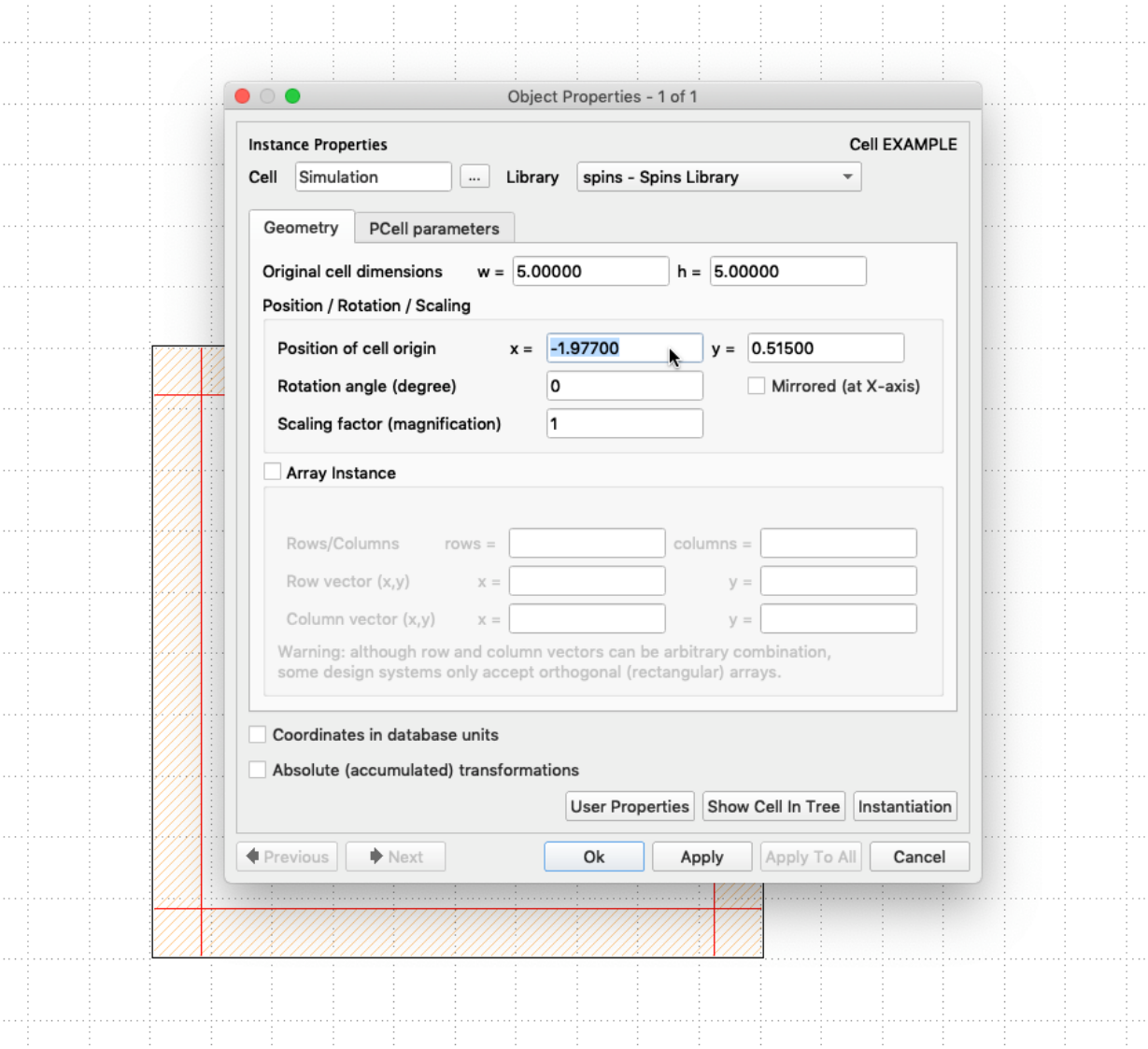
First we will define the simulation region by selecting *Simulation* in the instance list, and then pressing **OK**. We can then set the simulation domain size and set boundary condition parameters. For this example we set the *Simulation Span X* and *Simulation Span Y* to be the default 5 um and leave the boundary conditions to be the default *PML* type, with 10 cells of *PML thickness*. Click **Apply**, **OK**, and then click anywhere to insert the instance cell to our layout. Note, this is partly why we specified *dx* earlier, to convert between the number of cells of *PML thickness* to a unit of length.



Note:

If you only see a boxed outline of the created instance, press **Shift** + '+' (Plus key) to adjust the reference level to visibly see the simulation region and the PMLs just created.

We can adjust the instance we just created by selecting the object with left click, and then pressing **Q**, or double clicking the object. Under the PCell parameters tab we can adjust the variables we set at creation of the instance, and under geometry we can set the exact position of the objects. We can use this opportunity to center our simulation by clicking the **Geometry** tab, and setting the *Position of cell origin* to be 0, 0.



Click OK and then repeat the process for the Design Region instance, where again we use the default 2um span and center the instance upon creating it, through the geometry tab.

Note:

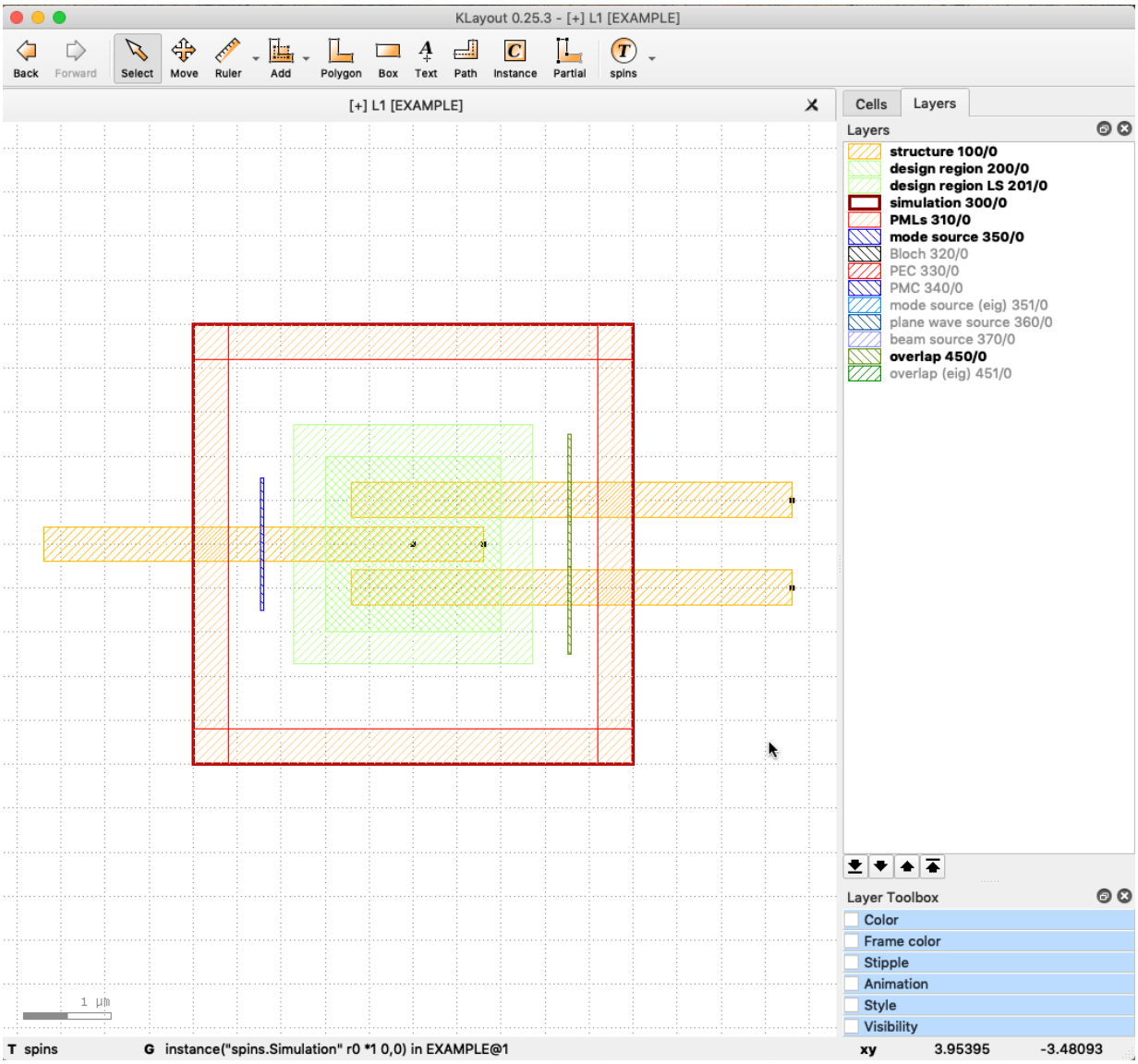
One might notice that the design region instance creates two design regions, *design region* and *design region LS*. The reason for this is that during the first stage of optimization over the *design region* (the inner of the two), sharp features may develop at the interface of the device and waveguides. To eliminate these problematic features, we extend the design region into the waveguides in the second stage of optimization. Consequently, this scheme requires that waveguides extend into *design region LS* for guarantees of fabricability. The amount that the *design region LS* extends beyond the *design region* is computed automatically as a function of *dx* and *fcon* defined at the startup of KLayout.

Add Source/Overlap Waveguides

Next, we create a waveguide instance through the same menus. We use the default values for this example

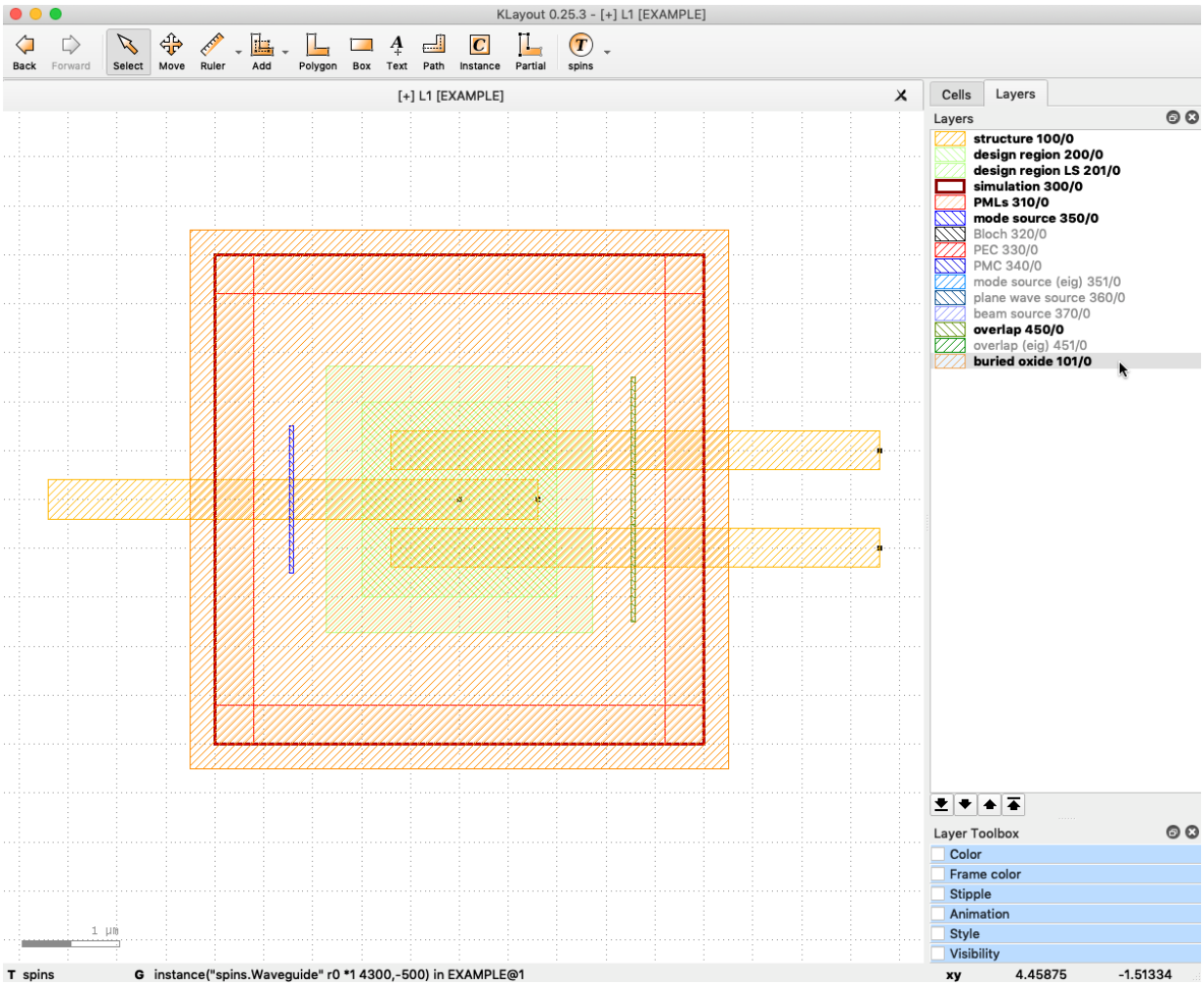
and first roughly place the waveguides near the desired location and then later edit their position precisely by changing the *Geometry* properties. While the waveguide instance can extend beyond the simulation region and within the design region, the source and overlap must not be placed inside the PML region. In addition, the overlap regions are able to overlap; however, one must be careful not to have an overlap region overlap with another waveguide, as the waveguide mode solver will result in incorrect modes.

For the example we place a single *source* waveguide on the left of the design region and two *overlap* waveguides on the right of the design region. The completed structure should look similar to below.

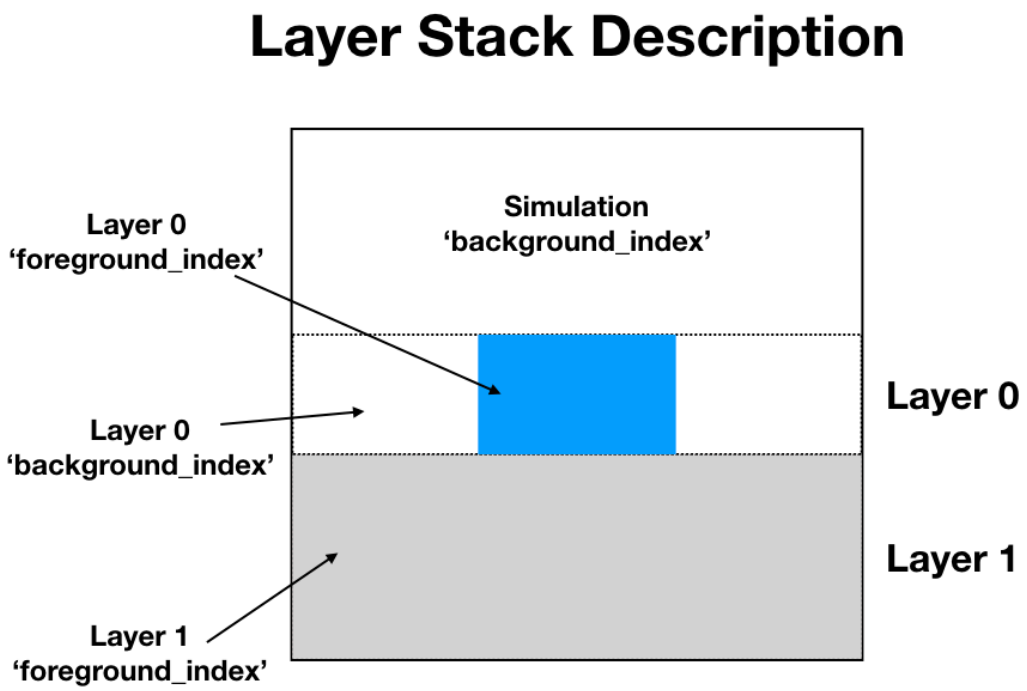


Add Buried Oxide Layer

Lastly, we provide information regarding the layer stack. We do this by creating addition layers in the GDS, where each GDS layer corresponds to a seperate layer in the material stack. For example, if we want to create a buried oxide layer for our waveguides to sit on top of, we first create a new GDS layer under **Edit > Layer > New Layer** and specify the layer number to be 101. We then draw a box in this layer with extent greater than our simulation domain to indicate that we will fill this entire layer with the same value of the refractive index.

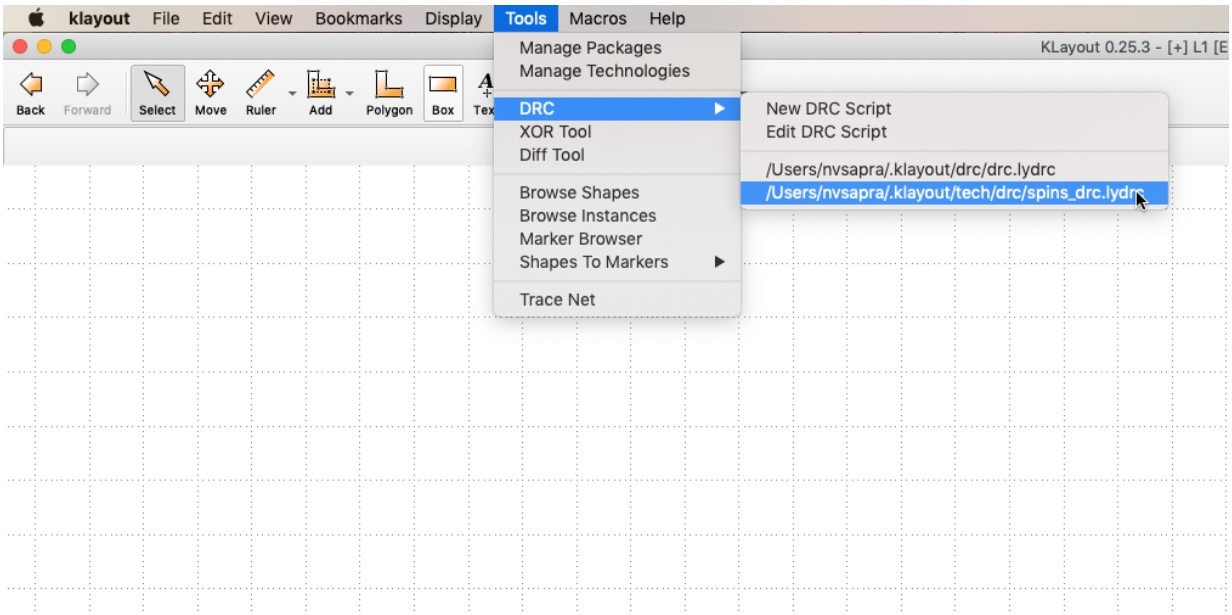


This refractive index for the areas specified with polygons with later be given the value of *foreground_index*. The image below illustrates the concepts of the layers and foreground and background indicies.



DRC Check and Save

Once the GDS file has been completed, a Spins *design rule check (DRC)* is recommended to be run. The DRC can be accessed through the **Tools > DRC** and then select the DRC option associated with Spins.



If there are no numbers listed beside the checks, then the GDS has passed the DRC. Otherwise, click

through the categories to determine what has caused the DRC to fail.

Once all DRC violations have been resolved, save the GDS file in a location where the remainder of Spins commands will be run from.

Problem Specification

After creating the GDS, we need to complete the problem specification file. This file describes the simulation, optimization, and properties of the problem geometry not captured in the GDS file (e.g. layer thicknesses).

To create the problem specification file, we use the Spins function:

```
$ spins autogen problem-spec [gds-file]
```

> **Note:**
>
> If you are unable to run the Spins command, ensure that the spins virtual environment is running. If installed properly, this is accomplished by running `$ workon spins`

Running this line outputs the problem.yml file which must be completed by the user in a text editor. Below we go through each section for defining the optimization of our 3D wavelength demultiplexer.

Simulation

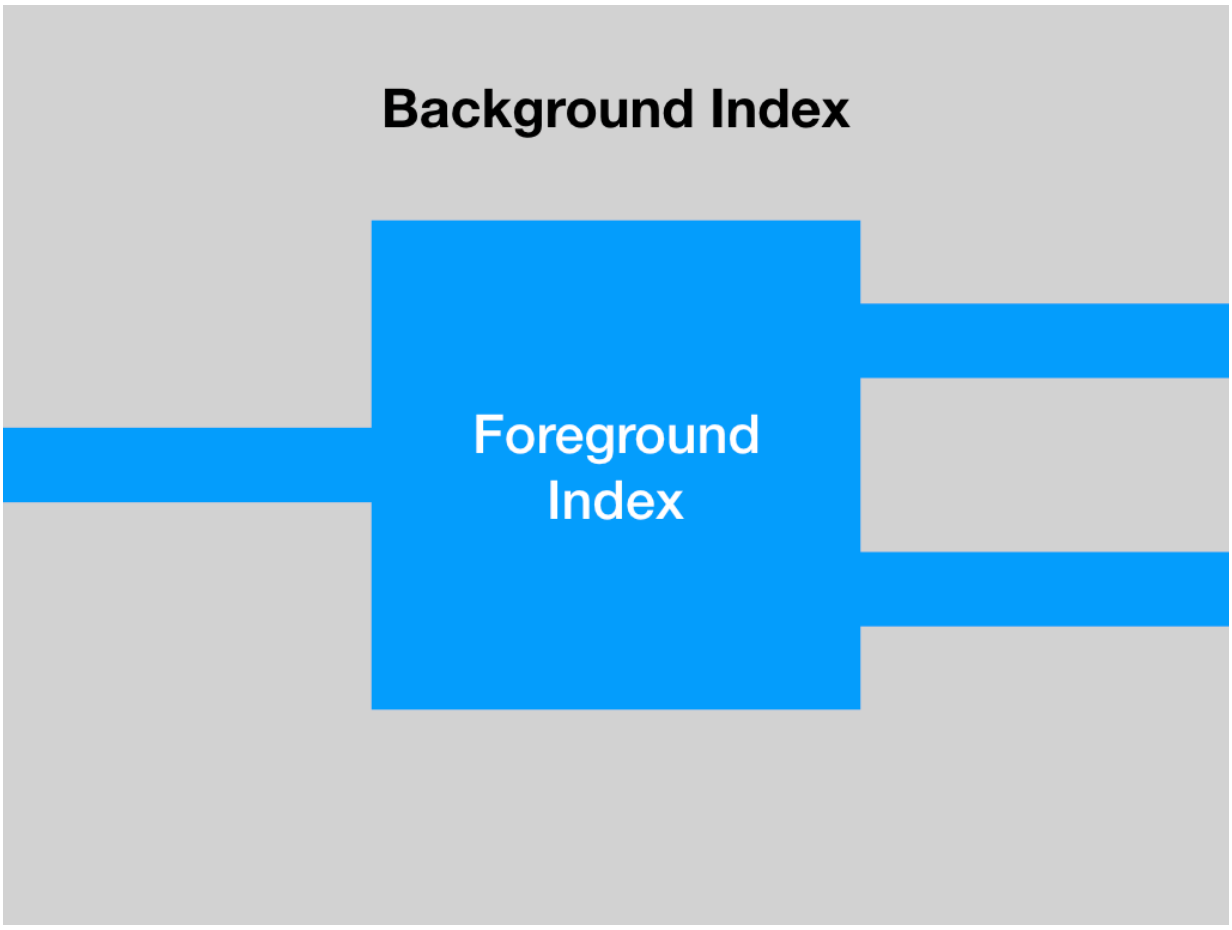
We set the value of *dx* to be 0.040 um and give a total span in the dimension to be 2um, centered about 0. The background index of the simulation can be thought of as the index of the cladding in this example, and so we set it to be 1 for air-cladding. We keep all boundary conditions as PMLs with 10 layers.

```
simulation:
  # Mesh size.
  dx: 0.04
  # Minimum z-coordinate of simulation space.
  z_min: -1
  # Maximum z-coordinate of simulation space.
  z_max: 1
  # Default background index.
  background_index: 1
  # Boundary condition to apply ('pml', 'pec', 'pmc', 'bloch', or 'periodic').
  boundary_condition_xmin: pml
  # Boundary condition to apply ('pml', 'pec', 'pmc', 'bloch', or 'periodic').
  boundary_condition_xmax: pml
  # Boundary condition to apply ('pml', 'pec', 'pmc', 'bloch', or 'periodic').
  boundary_condition_ymin: pml
  # Boundary condition to apply ('pml', 'pec', 'pmc', 'bloch', or 'periodic').
  boundary_condition_ymax: pml
  # Boundary condition to apply ('pml', 'pec', 'pmc', 'bloch', or 'periodic').
  boundary_condition_zmin: pml
  # Boundary condition to apply ('pml', 'pec', 'pmc', 'bloch', or 'periodic').
  boundary_condition_zmax: pml
  # Number of PML layers (not thickness).
```

```
    pml_xmin: 10
    pml_xmax: 10
    pml_ymin: 10
    pml_ymax: 10
    pml_zmin: 10
    pml_zmax: 10
```

Layer Stack

This section defines the layer stack for our 3D simulation. We set the cladding index above with the simulation *background_index*, but need to specify properties for the device and oxide substrate. We set the minimum and maximum z extents of the *layer0* (the layer which contains waveguides and device) to to have a total span of 0.220 um, with *foreground_index* of that of silicon and *background_index* of that of air. The meaning of these indicies can be seen in the below figure.



The second layer, *layer1*, is that of the oxide substrate. Here we set the maximum z-coordinate to coincide with the minimum of the device layer, and let the minimum coordinate exceed the simulation region to ensure the substrate is uniform below the device. As drawn in the GDS file, the entire layer is comprised of a foreground object, and so only the *foreground_index* must be set; however, we set both the foreground and background indicies to be that of silica. Only *layer0* is will be optimized, and so we list that as the singular element of the *design_layer* list.

```
    layer_stack:
    - name: layer0
```

```
    # Layer to get polygons from ([layer, datatype]).
    gds_layer: [100, 0]
    # Minimum z-coordinate of layer.
    z_min: -0.11
    # Maximum z-coordinate of layer.
    z_max: 0.11
    # Refractive index of foreground.
    foreground_index: 3.45
    # Refractive index of background.
    background_index: 1
  - name: layer1
    # Layer to get polygons from ([layer, datatype]).
    gds_layer: [101, 0]
    # Minimum z-coordinate of layer.
    z_min: -2
    # Maximum z-coordinate of layer.
    z_max: -0.11
    # Refractive index of foreground.
    foreground_index: 1.4
    # Refractive index of background.
    background_index: 1.4
  # List of stack layer names that are part of the design region.
  design_layers: [layer0]
```

Sources

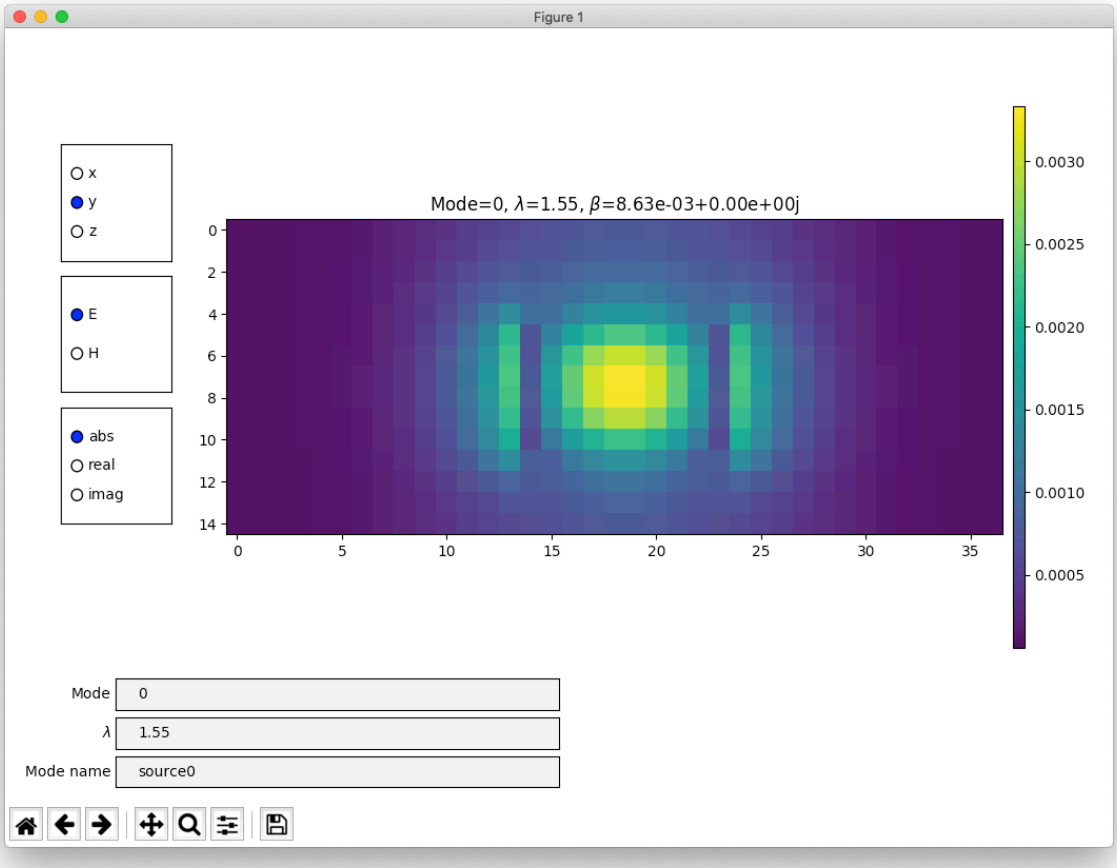
The waveguide sources are generated by solving for the waveguide mode over a slice, and then building a total-field scattered-field current source from the fields. To characterize this slice, we need to provide the additional information of the height of the slice. We set the height of this slice to be 0.6 um as that should easily contain the mode of our tightly confining waveguide. In addition, we specify the fundamental mode as our source, *mode_number* 0, and note that the axis of propagation is in the positive 'x' direction.

```
  sources:
  - name: source0
    type: waveguide_mode
    # Center location of the source/overlap region in GDS.
    center: [-1.77, 0.0]
    # Center location of source/overlap region in z-axis.
    z_center: 0
    # Height of region used for solving waveguide mode.
    height: 0.6
    # Mode number of the waveguide mode, where 0 is the fundamental.
    mode_number: 0
    # Mode propagation direction. Either '+x', '-x', '+y', or '-y'.
    axis: +x
    # Power emitted by source.
    power: 1.0
```

At this point, it may be useful to confirm the source appears as we intend. We can do so by running:

```
$ spins autogen visualize mode
```

We can then visualize the waveguide mode over all three components. This tool is useful to determine which mode number corresponds to the desired mode to inject or optimize for if unknown ahead of time.



Overlaps

The overlaps section follows similarly to the sources section. Here we again define the characteristics of the slice used to compute the waveguide mode which will be the target of the optimization for the given overlap monitor.

```
  overlaps:
- name: overlap0
  type: waveguide_mode
  # Center location of the source/overlap region in GDS.
  center: [1.73, 0.5]
  # Center location of source/overlap region in z-axis.
  z_center: 0
  # Height of region used for solving waveguide mode.
  height: 0.6
  # Mode number of the waveguide mode, where 0 is the fundamental.
  mode_number: 0
  # Mode propagation direction. Either '+x', '-x', '+y', or '-y'.
  axis: +x
  # Power emitted by source.
  power: 1.0
- name: overlap1
  type: waveguide_mode
  # Center location of the source/overlap region in GDS.
  center: [1.73, -0.5]
```

```
  # Center location of source/overlap region in z-axis.
  z_center: 0
  # Height of region used for solving waveguide mode.
  height: 0.6
  # Mode number of the waveguide mode, where 0 is the fundamental.
  mode_number: 0
  # Mode propagation direction. Either '+x', '-x', '+y', or '-y'.
  axis: +x
  # Power emitted by source.
  power: 1.0
```

Scattering Matrix

Once the input and output ports have been specified, it is then necessary to specify their behavior for each desired mode. This is accomplished in the Scattering Matrix section. First, we specify the wavelengths we will simulate a given source at. Then, for each source, we specify the target transmission at each wavelength, for every overlap monitor.

In this example, we use the single source we defined in the GDS file to inject a waveguide mode at 1.3 and 1.55 um. For the first overlap we want transmission at 1.3 um, but rejection at 1.55 um, and so we set the *target_tranmission* to `[1, 0]`. The opposite behavior is desired for the second overlap, and so we set the target_transmission accordingly to `[0, 1]`. The *target_phase* parameter is used to set a phase relationship between the waveguide modes and the *tolerance* parameter for each overlap sets how stringent the *target_tranmission* goal is.

```
scattering_matrix:
- source_name: source0
  # List of wavelengths to simulate at.
  wavelengths: [1.3, 1.55]
  # Defines behavior of overlaps for this source.
  overlaps:
  - overlap_name: overlap0
    # Target transmission where each element of the list corresponds to each
    # wavelength. A dash (-) indicates that transmission is ignored.
    target_transmission: [1,0]
    # Target phase of the mode overlap for each wavelength.
    # A dash (-) in the list means that phase is not taken into
    # account.
    target_phase: ['-','-']
    # Tolerance `tol` for corresponding transmission target `T`. The
    # optimization will try to make the transmission lie within
    # `[T - tol/2, T + tol/2]`.
    tolerance: 0
  - overlap_name: overlap1
    # Target transmission where each element of the list corresponds to each
    # wavelength. A dash (-) indicates that transmission is ignored.
    target_transmission: [0,1]
    # Target phase of the mode overlap for each wavelength.
    # A dash (-) in the list means that phase is not taken into
    # account.
    target_phase: ['-','-']
    # Tolerance `tol` for corresponding transmission target `T`. The
    # optimization will try to make the transmission lie within
    # `[T - tol/2, T + tol/2]`.
    tolerance: 0
```

Optimization

Lastly, we set simulation parameters governing the minimum feature size (min_feature_size) and the type of optimization conducted. The optimization typically runs in two stages: 'continuous', where the permittivity can vary continuously from the background to the foreground index specified, and 'discrete', where it can be only either or. We provide the option for only running one of the two modes of operation, or both. Typically 'full' is recommended for best performance.

```
optimization:
  min_feature_size: 0.120
  # Number of iterations to run optimization.
  iters: 40
  # Optimization procedure to use. Possible choices are:
  # - 'full' for full optimization
  # - 'continuous' for continuous optimization only
  # - 'discrete' for discrete optimization only.
  opt_mode: full
```

Run Optimization

Once the problem specification file has been completed, we are ready to run Spins! To do so, simply run the command:

```
$ spins autogen run -s folder_name
```

where *folder_name* corresponds to the desired folder location of the output results.

Analyze Optimization

Once the optimization has been completed, change to the directory containing the optimization results. The results can be visualized by running:

```
$ spins visualize summary
```