Hello there! This tutorial will provide you with instructions to create your own Paraview visualization for the Purdue Computational Quantum Electromagnetics Lab.

After downloading the Paraview software through this link (https://www.paraview.org/download/), we can begin.

Contents:

1) Creating VTK files:

A Python code is made for the Lab, which takes as input the data files and creates the appropriate VTK file; however, the process to create one yourself will be explained here, and additional links with extra resources will be provided along with the document.

To begin, the first line necessary in a VTK file is the line specifying the version of the Visualization Toolkit (VTK) being used (May 2021 when this was written); the version being used is version 3, but as new updates are released, this document, as well as the Python code, shall be updated.

To exemplify, in any VTK file you write, line one should be EXACTLY as follows: # vtk DataFile Version 3.0

Line two will always specify the header, so you can write whatever title you wish for your data pool. Example: “ParaviewTestVTK for CQEM Lab.”

The third line in a VTK file will declare the file format, so you get to choose if your data will be written in Binary or ASCII characters. ASCII characters are not only easier to use but better documented. Example for the third line, “ASCII.”

Line four will declare the data structure of your file, which includes Polygonal Data, Structured Points, Structured Grid, Rectilinear Grid. The following real-world examples will be that of Polygonal data, which is the easiest to use. You should declare the data structure in line four, where you should write the following “DATASET POLYDATA” (without the quotations).

So far, you should have written the same as what is below, except for the header, which can be of your own choice.

vtk DataFile Version 3.0
VTK File created here
ASCII
DATASET POLYDATA

For the next step, we will start writing the actual structure of your visualization. The first thing you’ll want to do is declare the nodes. To do that, you’ll have to know how many points there are in your data, which is the number of lines in your node file. Next, you’ll have to declare the datatype of your points (double, float, int…). An example will be written below with 20 points of the double datatype.

POINTS 20 double
2.5 -2 1.5
2.5 2 1.5
2.5 0 1.5
-2.5 2 1.5
0 2 1.5
-2.5 -2 1.5
-2.5 0 1.5
0 -2 1.5
2.5 2 -1.5
2.5 -2 -1.5

```
2.5 0 -1.5
-2.5 -2 -1.5
0 -2 -1.5
-2.5 2 -1.5
-2.5 0 -1.5
0 2 -1.5
-2.5 -2 0
2.5 -2 0
-2.5 2 0
2.5 2 0
```

It is important to remember that the point coordinates and any other data written in VTK format cannot be separated by a comma but have to be separated by a space instead.

In sequence, for the POLYDATA data structure chosen for your endeavors, it will be necessary to define the cells that compose the mesh. To that end, it will be essential to know the specific polygons that make the mesh (triangles, squares, pentagons, etc.), the number of cells (how many polygons there are), and the size of the cell list (number of lines multiplied by the number of items per line). An example and explanation will follow below:

```
POLYGONS 36 144
3 0 2 7
3 2 1 4
3 3 6 4
3 6 5 7
3 6 7 4
3 4 7 2
3 8 10 15
3 10 9 12
3 11 14 12
3 14 13 15
3 14 15 12
3 12 15 10
3 5 16 7
3 16 11 12
3 9 17 12
3 17 0 7
3 17 7 12
3 12 7 16
3 3 18 6
3 18 13 14
3 11 16 14
3 16 5 6
3 16 6 14
3 14 6 18
3 1 19 4
3 19 8 15
3 13 18 15
3 18 3 4
3 18 4 15
3 15 4 19
3 0 17 2
3 17 9 10
3 8 19 10
3 19 1 2
3 19 2 10
3 10 2 17
```

In the first line, you can see that we specified that the following data would be related to the cells by writing POLYGONS. Next, we see the number 36, which is the total number of cells or the total number of lines (feel free to count!). Following the number 36 on the top line, we see the number 144, which represents the number of items in the cell list. Since there are 36 lines and 4

items per line, we have 36 x 4 for the total number of items that equal 144. Now we move forward to the numbers below, and to start off, you might have noticed that every line starts with the number 3. That number is an indicator of the cell type. As you might have already guessed, 3 represents a triangle, while a 4 would represent a quadrilateral, and so on. The following numbers in each line are generated by your data and represent the indices of the nodes that compose your cells. The first thing to be aware of when writing the indices in Paraview is the fact that the software reads on base 0, so if your data is generated by software that uses base 1, you'll need to make a correction when outputting the contents to your VTK file. The second thing to be aware of is that the indices written are only read when they are integers, so while your nodes, vectors, scalars, and all else can be in whatever datatype you prefer, the indices have to be integers.

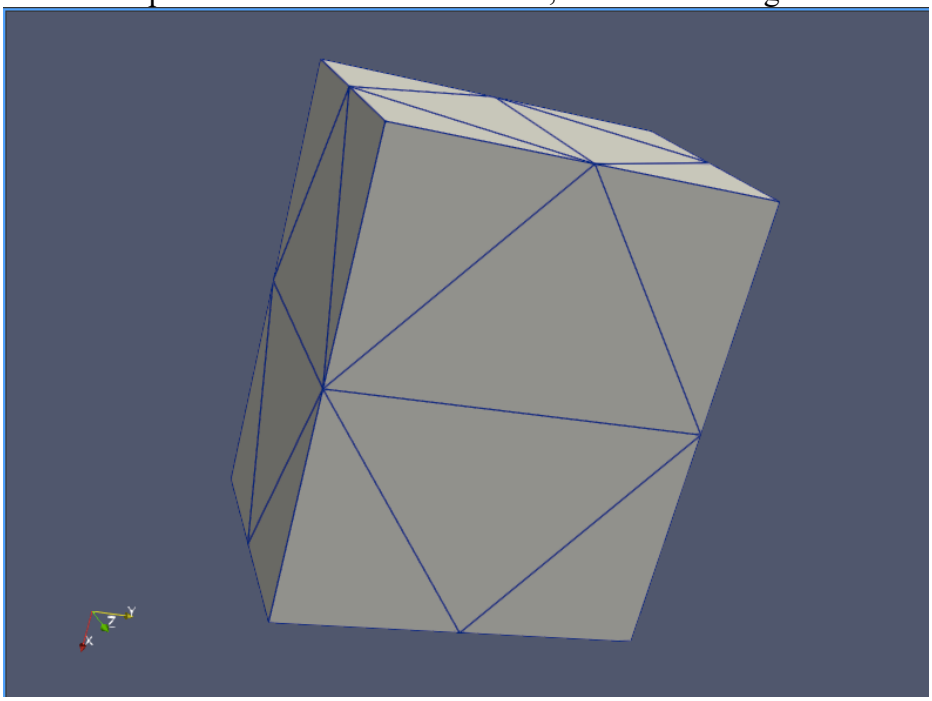
Now let's put everything we wrote down in this tutorial together and save the file as "cubeTest.vtk" (remember to keep your files as ".vtk").

```
# vtk DataFile Version 3.0
VTK File created here
ASCII
DATASET POLYDATA

POINTS 20 double
2.5 -2 1.5
2.5 2 1.5
2.5 0 1.5
-2.5 2 1.5
0 2 1.5
-2.5 -2 1.5
-2.5 0 1.5
0 -2 1.5
2.5 2 -1.5
2.5 -2 -1.5
2.5 0 -1.5
-2.5 -2 -1.5
0 -2 -1.5
-2.5 2 -1.5
-2.5 0 -1.5
0 2 -1.5
-2.5 -2 0
2.5 -2 0
-2.5 2 0
2.5 2 0
POLYGONS 36 144
3 0 2 7
3 2 1 4
3 3 6 4
3 6 5 7
3 6 7 4
3 4 7 2
3 8 10 15
3 10 9 12
3 11 14 12
3 14 13 15
3 14 15 12
3 12 15 10
3 5 16 7
3 16 11 12
3 9 17 12
3 17 0 7
3 17 7 12
3 12 7 16
3 3 18 6
3 18 13 14
3 11 16 14
3 16 5 6
```

```
3 16 6 14
3 14 6 18
3 1 19 4
3 19 8 15
3 13 18 15
3 18 3 4
3 18 4 15
3 15 4 19
3 0 17 2
3 17 9 10
3 8 19 10
3 19 1 2
3 19 2 10
3 10 2 17
```

When we open ‘cubeTest.vtk’ in Paraview, this is what we get:



The cells are outlined for better representation and visualization in this tutorial. As you can see, there are 6 cells for each of the 6 sides of the cube (36 cells in total), and all cells are triangles as we specified.

You must have noticed that in this visualization, there are no vectors nor scalars, and the reason for that is because we did not specify any. For the final part of the file, we will be writing the dataset attributes. This part begins with the keywords POINT_DATA or CELL_DATA, followed by an integer number specifying the number of points or cells, respectively. (It doesn’t matter whether POINT_DATA or CELL_DATA comes first.) Other keyword/data combinations then define the actual dataset attribute values (i.e., scalars, vectors, tensors, normals, texture coordinates, or field data).

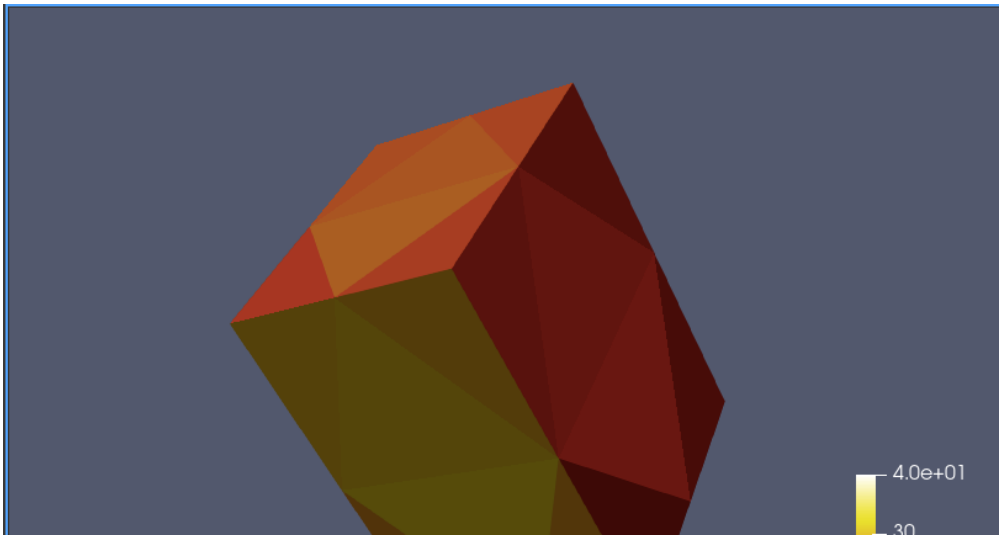
We are mostly going to use scalars and vectors, so that is what’s going to be looked at here. Building off of our cube file, we will start with the scalars, and an example is provided below to exemplify.

```
CELL_DATA 36
SCALARS colors double
LOOKUP_TABLE tableScalar
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
```

The first line specifies that the following attributes will belong to the cells (scalars most commonly match with the cells). Followed by CELL_DATA still in the first line, we specify the number of cells (will be the same as the number of polygons declared above), which is 36 in this case. In the second line, we declare which attribute we are going to write, and in this case, we define the scalars. Followed by SCALARS, still in the second line, we decide how this attribute will be named inside Paraview. In this case, it was called "colors." Lastly, for the second line, we determine the datatype for your scalar data, which is 'double' for this one. For the third line, the keywords "LOOKUP_TABLE" and a title of your choice for the table is necessary. Following this syntax, you pour down your scalar data as it came to you in your simulation.

For reference, the visualization and entire file now look like this:

```
# vtk DataFile Version 3.0
VTK File created here
ASCII
DATASET POLYDATA
POINTS 20 double
2.5 -2 1.5
2.5 2 1.5
2.5 0 1.5
-2.5 2 1.5
0 2 1.5
-2.5 -2 1.5
-2.5 0 1.5
0 -2 1.5
```



2.5 2 -1.5
2.5 -2 -1.5
2.5 0 -1.5
-2.5 -2 -1.5
0 -2 -1.5
-2.5 2 -1.5
-2.5 0 -1.5
0 2 -1.5
-2.5 -2 0
2.5 -2 0
-2.5 2 0
2.5 2 0
POLYGONS 36 144
3 0 2 7
3 2 1 4
3 3 6 4
3 6 5 7
3 6 7 4
3 4 7 2
3 8 10 15
3 10 9 12
3 11 14 12
3 14 13 15
3 14 15 12
3 12 15 10
3 5 16 7
3 16 11 12
3 9 17 12
3 17 0 7
3 17 7 12
3 12 7 16
3 3 18 6
3 18 13 14
3 11 16 14
3 16 5 6
3 16 6 14
3 14 6 18
3 1 19 4
3 19 8 15
3 13 18 15
3 18 3 4
3 18 4 15
3 15 4 19
3 0 17 2
3 17 9 10
3 8 19 10
3 19 1 2
3 19 2 10
3 10 2 17
CELL_DATA 36
SCALARS colors double
LOOKUP_TABLE tableScalar
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

For more reference, you can follow both links below. The first one is a tutorial on VTK file creation for POLYDATA and as well as OTHER types of data you might want to familiarize yourself with. The second link is a set of examples of VTK files of all kinds.

https://vtk.org/wp-content/uploads/2015/04/file-formats.pdf

https://www.visitusers.org/index.php?title=ASCII_VTK_Files#A_single_structured_cube_with_vectors

2) Modifying and Filtering Data:

2.0) Basics before moving further

Download the full Paraview tutorial through this link https://www.paraview.org/download/ under Documentation and go through it before moving along.

From now on, this document will assume you are familiar with the very basics of Paraview presented in the tutorial above.

Should you need more filters, follow this link to look at more filters available inside Paraview with a description included: https://www.paraview.org/Wiki/ParaView/Users_Guide/List_of_filters

2.1) Glyph Filter (vectors)

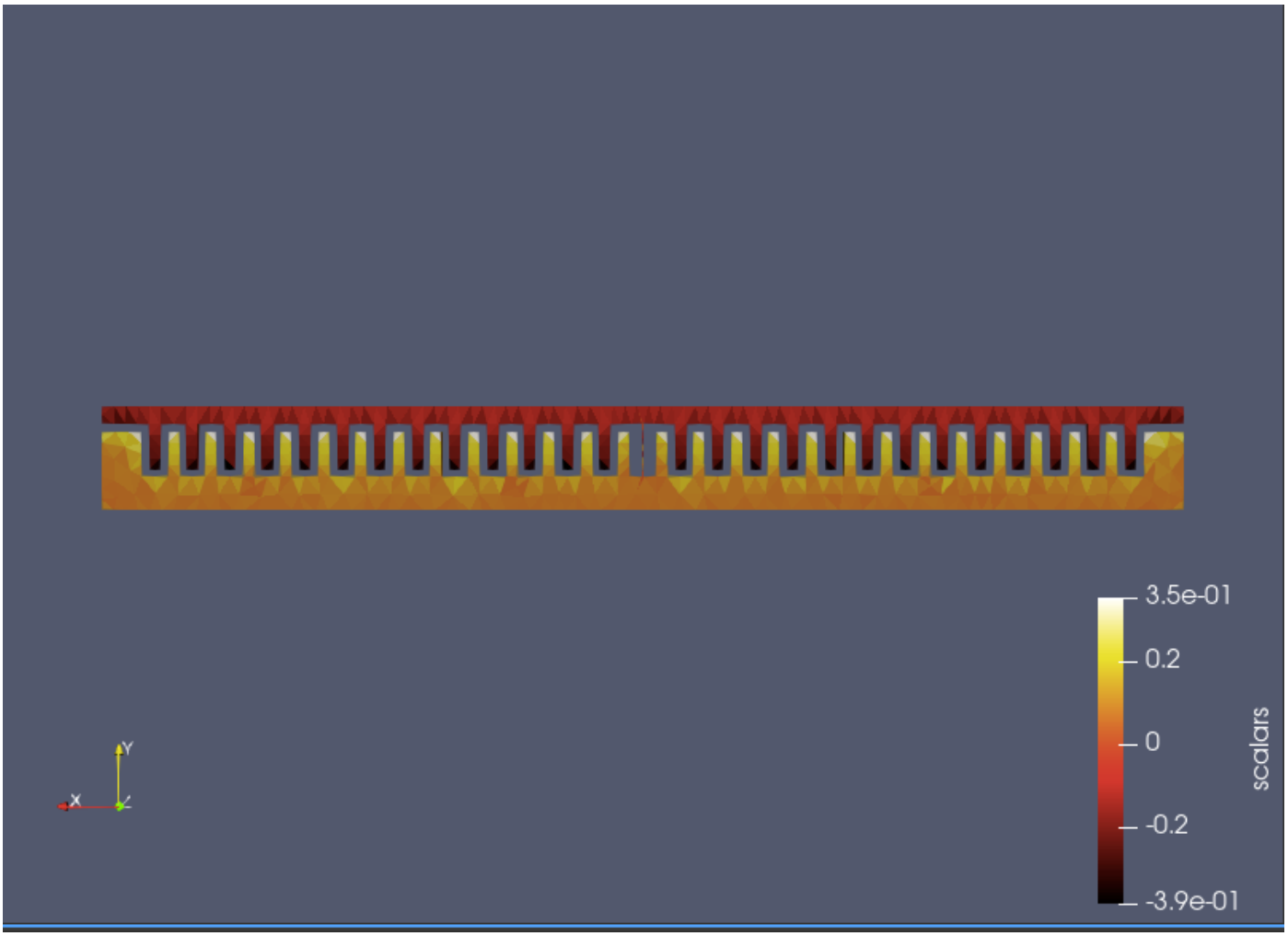
The Glyph filter is what allows us to see the vectors you included in your file and is represented by the following icon.

After applying the filter to your dataset, you can modify it through the properties tab as you wish (just be sure that the glyph filter is selected in the built-in menu).

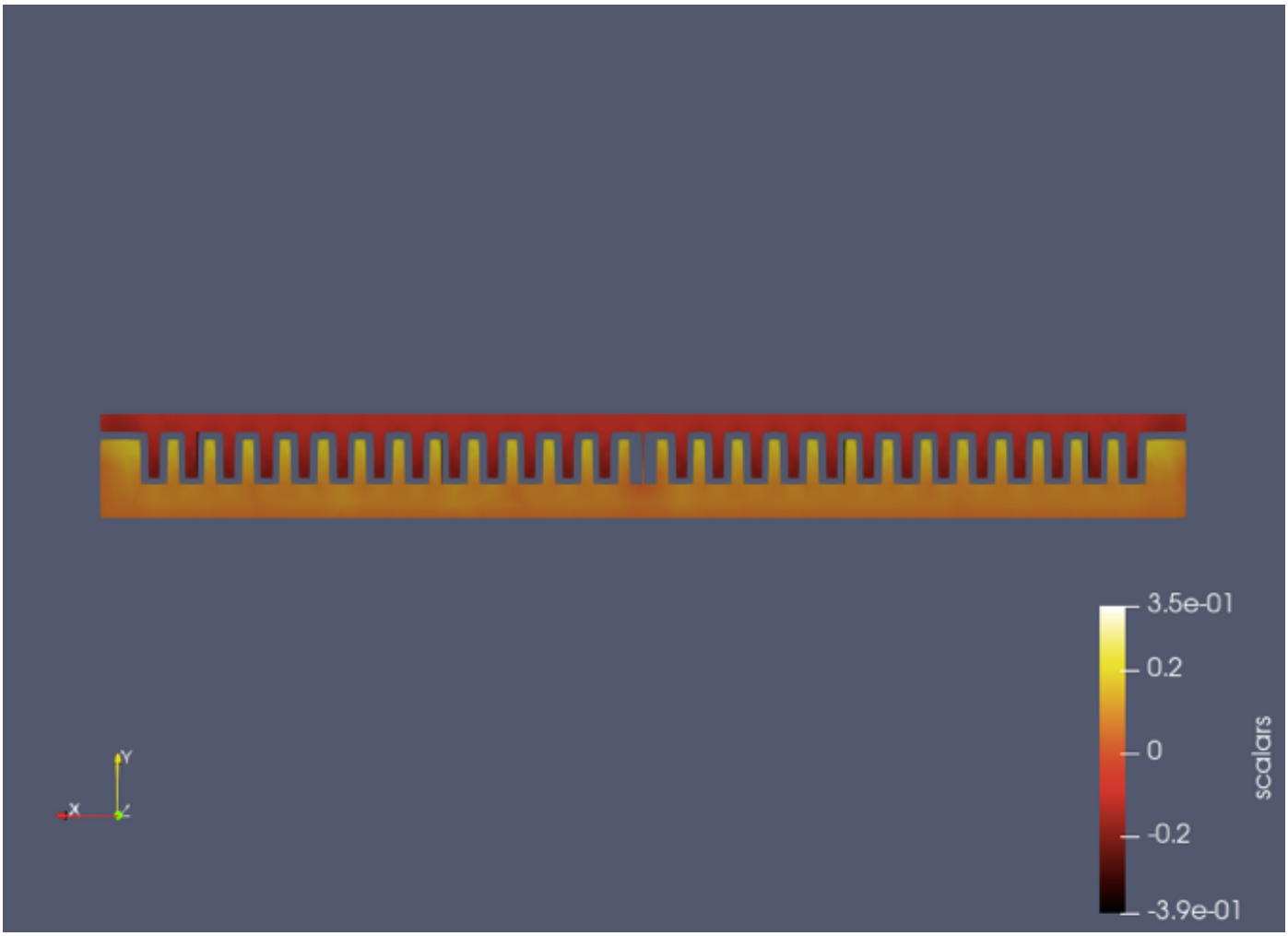
Be aware of the “Masking” property of the filter, which defines the points/cells from the dataset that get “glyphed”—generally, selecting “All Points” or “Every Nth Point” produces better-looking visualizations.

2.2) Cell Data To Point Data Filter (creating a gradient coloring)

The Cell Data to Point Data filter can be found by going through the filter options and clicking it. The reason why this filter is important is that it transforms the coloring in your figure into that of a smooth one. An example of the filters’ effect is shown below:



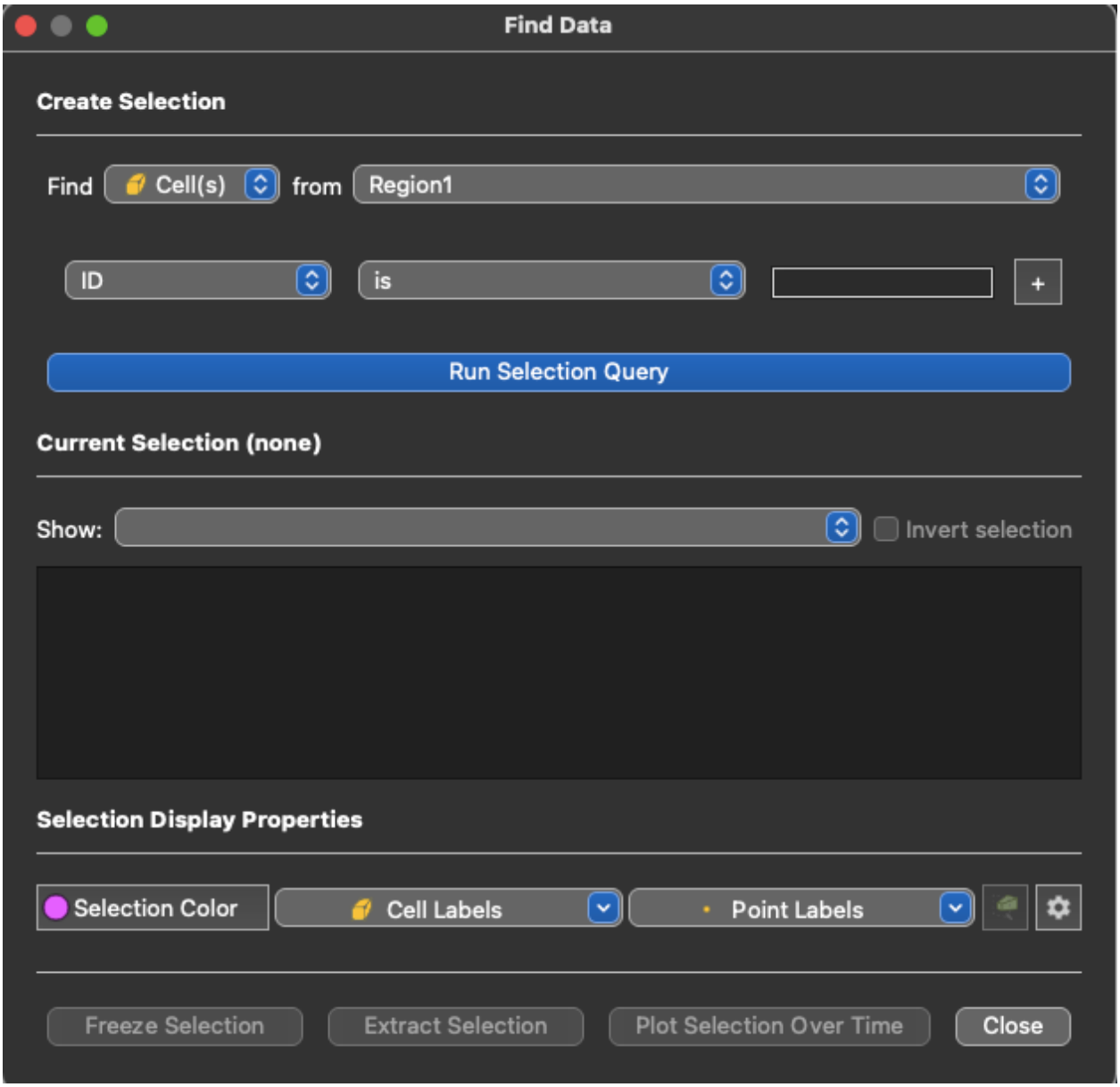
The image above shows the dataset without the Cell Data to Point Data filter.



The image above depicts the dataset after the Cell Data to Point Data Filter was applied.

2.3) Extract Selection (Selecting a specific cell ID range and plotting it separately)

There are many cases where you'll want to plot each part of your dataset separately, and for that, there is a specific procedure that you should follow. First, simply press the letter "V" or go to the Edit tab and select "Find Data." You'll see the following pop up:

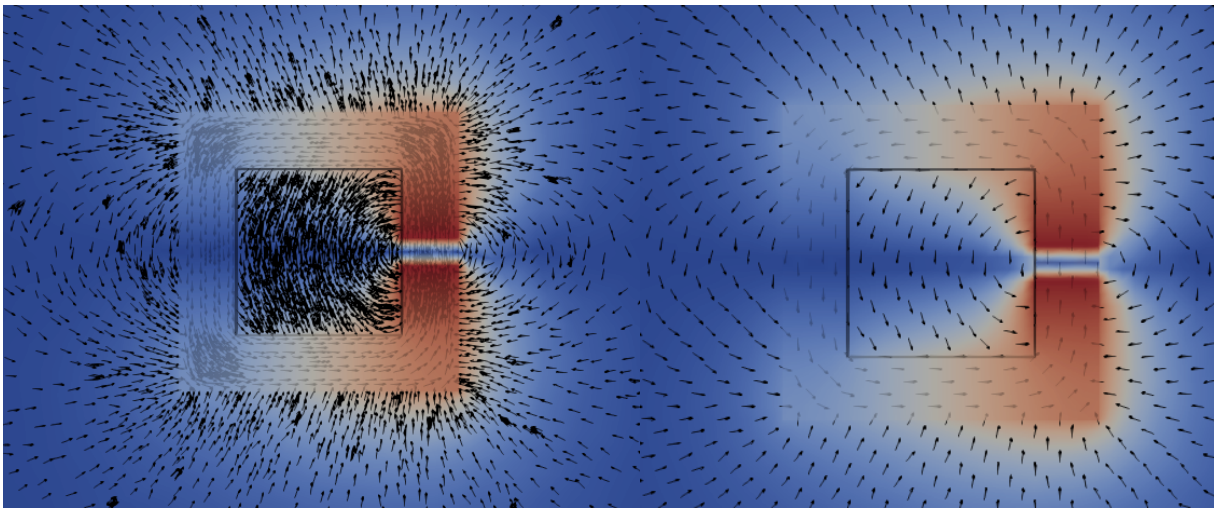


You can then use this browser to select the range of cells that fit your needs and, in sequence, use the Extract Selection button below to create a new branch in the Built-in menu with only the selected part.

2.4) ResampleToImage (Taking as an input an Unstructured Grid and converting that to a Structured Grid)

In case your data is an Unstructured Grid and has vectors included, you'll want to convert it into a Structured Grid to show your vectors in a cleaner manner. To do this, just apply the Filter through the Filters tab.

The images below serve as an example of what happens when you plot the vectors after using this filter.



The image on the left depicts the mesh before the filter and the image on the right after the filter, with the arrows on the same scale.

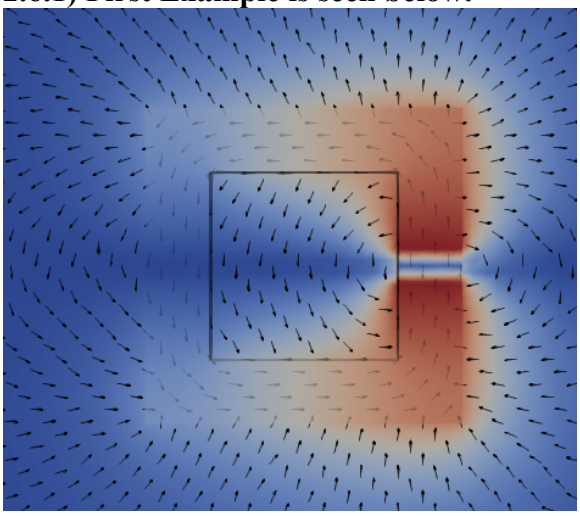
2.5) Slice (Selecting a single plane out of your 3D object)

Inside Paraview, you might want to get a look at what's going on in a single cross-sectional region in your visualization, and for that, you're going to use the Slice filter. It allows you to select the axis the plane will be normal in relation to as well as the origin point for the plane.

2.6) Visualization examples:

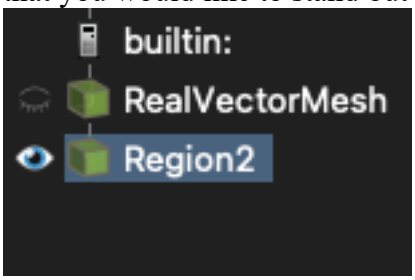
So we can put all we've learned together; following this paragraph, there are a few examples of visualizations that use the filters we've learned so far.

2.6.1) First Example is seen below.

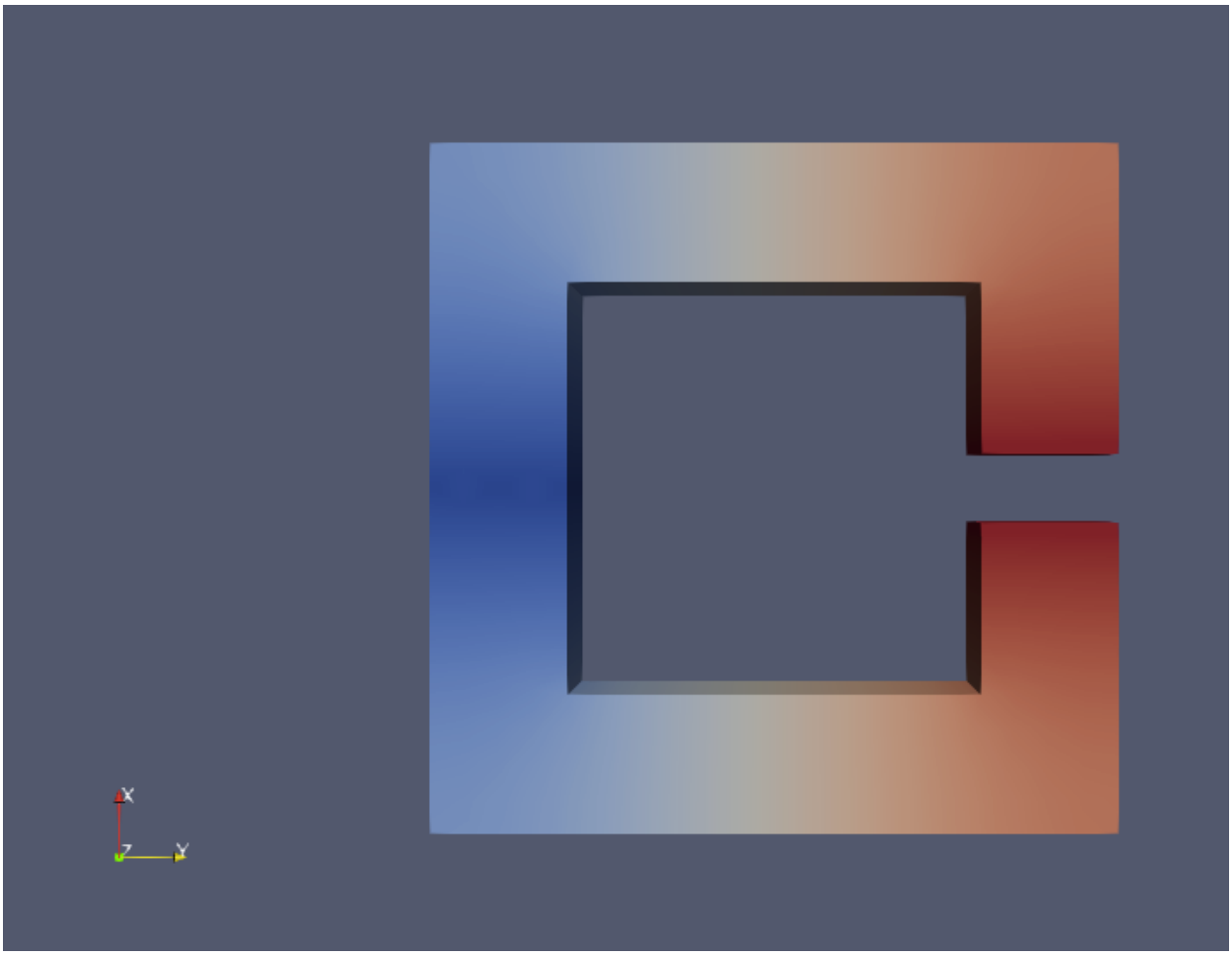


To arrive at the visualization seen in this example, there are a few steps you should take, all of which will encompass the filters we've seen in this section.

Firstly, use the process seen in the Extract Selection Filter section to separate the parts of your data that you would like to stand out or to be plotted differently.



<- This is how your built-in browser should be after the filter has been applied (the name of the pathway has been changed to Region2 instead of ExtractSelection1 to make it easier to follow along in the GUI as new filters are applied).



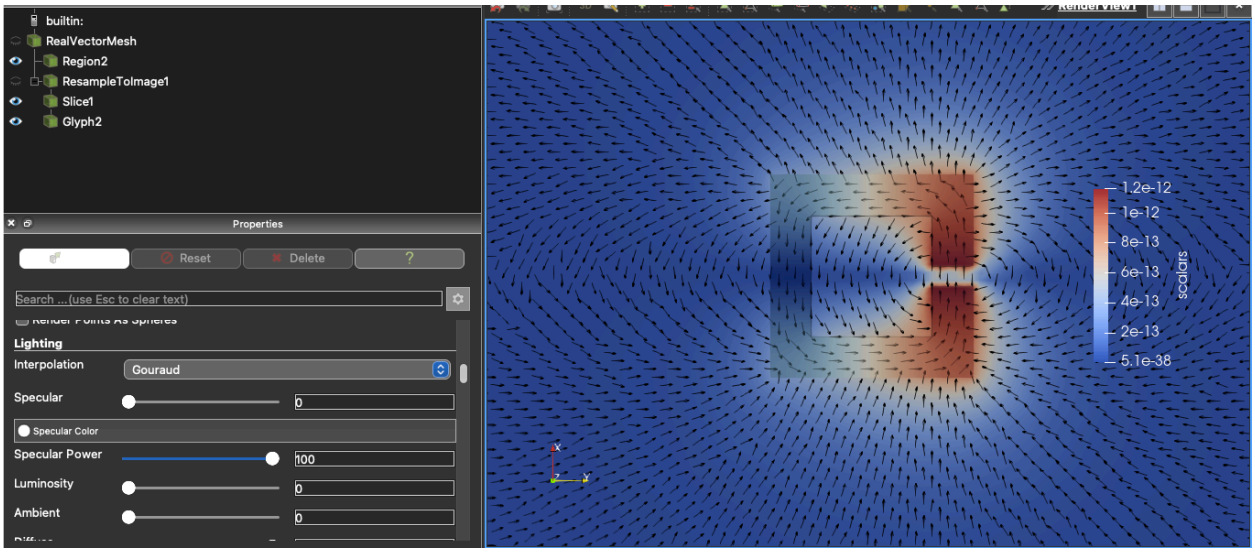
Above is what we see from the main file after applying the Extract Selection Filter.

After you've done the last part, we can move along into applying the following filter into the main file, so be sure that your main file is the one selected in the built-in browser and not the filter we just applied.

Firstly, use the filter to convert the Unstructured Grid data file into that of a Structured Grid we saw in section 2.4: ResampleToImage. Next, we'll take a slice of our object because we just care

about a single plane for now. To do that, apply the Slice filter seen in section 2.5 to the Structured grid we just created with the ResampleToImage filter. Just be sure, however, that the Slice filter is normal to the desired axis, which in this case will be the z-axis.

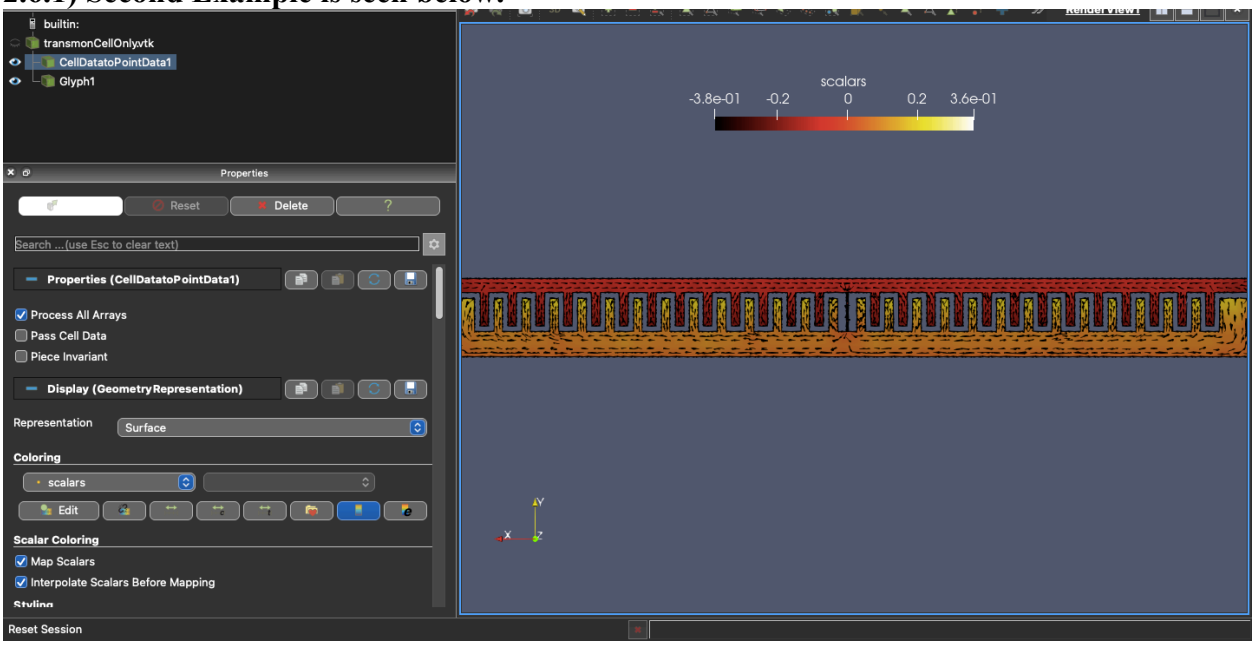
Followed by the Slice filter, we'll apply the Glyph filter to the Slice so we can see the vectors. The reason we use the glyph filter to the Slice filter pathway in the built-in browser and not to the Resample To Image filter pathway is that we want to plot the vectors only in 2D format.



After applying all of these filters, this is the visualization we have. As you can see in the filter browser to the left, the Resample To Image filter was applied to the main file titled RealVectorMesh and not to the Extract Selection filter labeled (Region2). Moreover, the Slice filter was applied to the Resample To Image filter pathway, just as the Glyph was applied to the Slice.

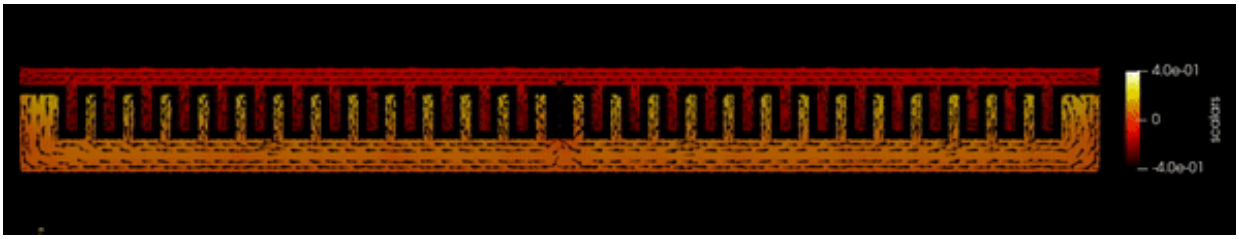
Now look closely and see that the only features selected to be seen in the visualization are filters and not the main file, something that will be very common in ParaView.

2.6.1) Second Example is seen below.



This second example is a lot simpler, but it shows the Cell Data to Point Data filter and the glyph filter both applied to the main file. In this case, the glyph filter is being applied to the main file because the vectors were attached to the cells in the VTK file during its creation.

3) Animating data:



To animate your data, you'll have to create a series of VTK files with data that changes and continues from the previous to the latter. When making that, be sure to give the same names to the files, in which only their numbering changes. To create each individual file, the procedure is exactly what was described in the first section.

After that, simply open Paraview, and it will interpret your file sequence as a single animation, so you can just open it as is. To save the animation video, just go to File, then to Save Animation, then select the directory and file's name, and then you're done.