Scientific Visualization

Final Project: Supernovae

Anand Kamble
Department of Scientific Computing
Florida State University

1 Introduction

In this project, we aim to develop a robust framework for visualizing core-collapse supernovae simulations using the VisIt software and its built-in scripting capabilities. Our test data is a two-dimensional, inhomogeneous Sedov explosion, which mimics the essential features of more complex simulations to be analyzed in the future. The Sedov dataset, comprising 100 individual time-slice files in a variant of the HDF5 format, serves as a representative example of the challenges encountered in visualizing such extreme astrophysical phenomena.

The primary objectives of this work is first, to dynamically rescale the visualization domain and the properties of the visualized quantities (e.g., density, pressure, and temperature ranges for pseudocolor plots) to provide a comprehensive understanding of the system's evolution.

2 Data Preperation

The dataset which is used for this project is a HDF5 dataset split into 100 timeslices. This dataset is hosted on the FSU website and can be downloaded using the following command or visiting this link.

```
wget http://people.sc.fsu.edu/~wwang3/sedov-2d-inhomogeneous-ppm-4lev.tar
```

After downloading the dataset which is an archive file currently in the `tar` format, we have to extract it. You can use any utility you prefer for this process. Since I am using I have used following command, where I am making a directory named `'dataset'` and then extracting the files into it.

```
mkdir dataset &&\
tar -xf sedov-2d-inhomogeneous-ppm-4lev.tar -C ./dataset/
```

If you are using Windows, you can also use the utility named WinRAR for extracting the tar file.

3 System Configuration

Everything that is mentioned in this report was done on the classroom machine. Which has the following specifications:

- Hostname: `class113.sc.fsu.edu`
- OS: Red Hat Enterprise Linux 8.9 (Ootpa)
- Architecture: x86-64
- Python version: 3.6.8

I got this information by using the command `hostnamectl`.

4 Preliminary Experiments

4.1 Loading Dataset in Python

To load the dataset files in python I created a class named `Dataset`. This class helps to load the dataset files using regex and returns a sorted array of file names.

```
class Dataset:
    def __init__(self,filename:str, directory = None) -> None:
        self.filename:str = filename
        self.directory:str = directory
        self.files:List[str] = None
        self.data:List[any] = list()

    def searchFiles(self) -> bool:
        if self.directory is None:
            print("No␣directory␣specified,␣cannot␣load␣files.")
            exit(1)
        else:
            self.files = glob.glob(self.directory + "/" + self.filename)
            self.files.sort(key=self.__sortFiles)
            return True

    def loadDataset(self) -> None:
        if self.files is None:
            print("No␣files␣to␣load,␣cannot␣load␣dataset.")
            exit(1)
        else:
            for file in self.files:
                with h5py.File(file, 'r') as f:
                    data_dict = dict()
                    dict.update(data_dict, {k : f[k][:] for k in f.keys()})
                    self.data.append(data_dict)

    def __sortFiles(self, file:str):
        regex_filter = re.compile(self.filename)
        sort_key = file.replace(regex_filter.search(file).group(),'')
        sort_key = sort_key.replace(self.directory + "/", "")
        return int(sort_key)
```

An example implementation of this class is as below:

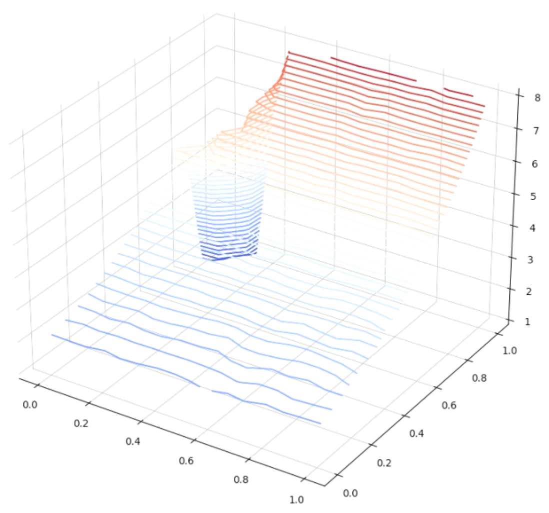
```
dataset = Dataset("sedov_hdf5_plt_cnt_*","dataset")
dataset.searchFiles()
dataset.loadDataset() # You can load the dataset only when needed.
```

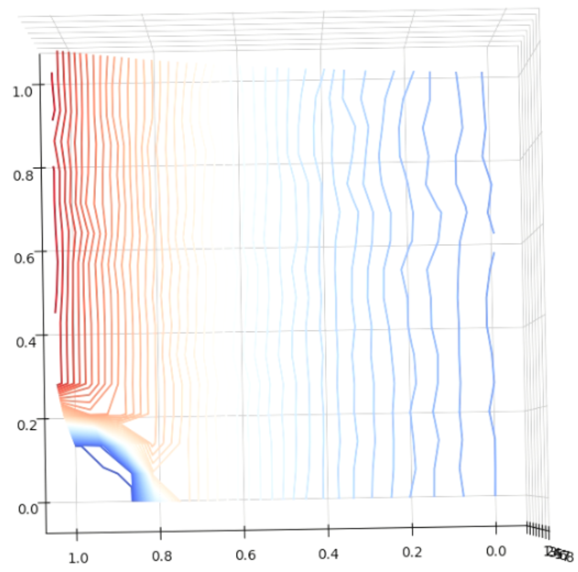
Note: This class requires the h5py python module. `pip install h5py`

4.2 Matplotlib

After successfully loading the data using the above class, I had to get a better understanding of the data structure. For that, I did some basic visualization using `Matplotlib`

```
dataSlice = dataset.data[0]['dens']
points_3d = np.squeeze(dataSlice,axis=1)
x = np.linspace(0, 1, 16)
y = np.linspace(0, 1, 16)
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(x,y,points_3d[0], 50, cmap=cm.coolwarm)
plt.show()
```





Here are some screenshots of what the contour plot looks like. In the right image, you can see how the density is higher on a curve which represents the shock wave at the 0th timeframe.

4.3 Using VTK C++

I decided to do this experiment using `C++` and VTK. To use VTK with `g++` I had to build the VTK package from their GitHub repository which is hosted at https://github.com/Kitware/VTK.[1] After successfully building the packages using `cmake`, I had the VTK package, and was able to compile the hello world program of Cylinder.[2] But running the executable always raised different errors which I was not able to debug. The error which was not solved is:

```
vtkXOpenGLRenderWindow (0x263af40): Cannot create GLX context.  Aborting
```

I did post my issue on StackOverflow, and was able to compile VTK by following the instructions posted in the comments by other users. You can see the post here https://stackoverflow.com/q/78354372/22647897

4.4 Using VTK Python

Since VTK offers a python package which can be installed using `pip`, which is more convenient. I installed it using the command:

```
pip install vtk
```

Although there were some errors at first, I fixed those with some help from the internet and the AI tool named Perplexity. [3] [4]

After Python VTK started working correctly, I wrote a script to convert the HDF5 data to a NumPy array and then render those points in a volume. You can find this script inside the `src` folder as `main.py` file.

After successful execution of the script, these are the renders which I got for the density parameter:

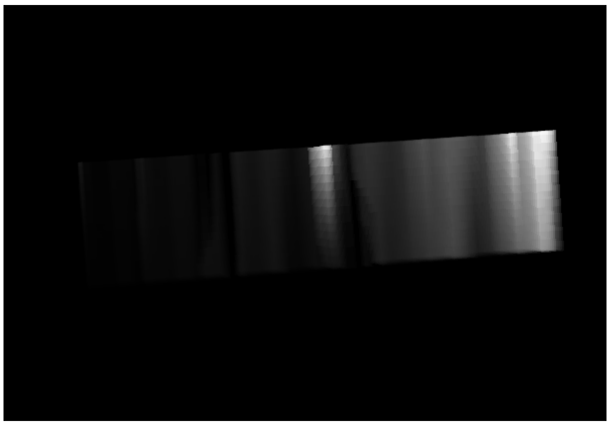
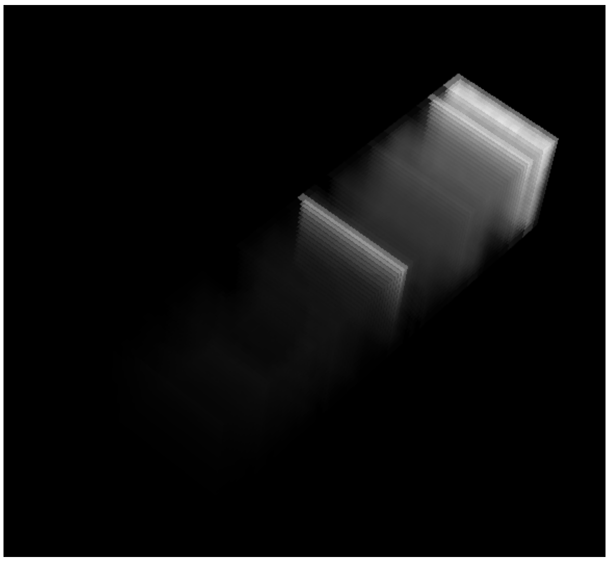



Figure 1: Python VTK Renders

As you can see in the above renders, the density is high at one end and you can also see the shockwave (the bright part in the middle). However it is not as clear to understand anything from this image since it lacks any information regarding the domain of the space and the range of values mapped from white to black color.

4.5 Selection of Method for Visualizing

After looking at the results from VTK, I decided to use the Visit software for rendering of the data since it is much more convenient and efficient for visualizing this type of data. Using VTK with Python we can get good-quality results but it will require a lot of time. Programming VTK from scratch to render the HDF5 dataset is a very complex task.

5 Methodology

I am using the Visit that is installed on the classroom machines, and the version I am using is:
- Visit 3.3.2
- git version 38c0859c41

5.1 Loading the data in Visit

To load the dataset into Visit, I have to use the following command since the files in the dataset do not have an extension and Visit is not able to auto-detect it's format.

```
visit -o "dataset/sedov_hdf5_plt_cnt_*", FLASH
```

This command tells Visit to open all the files that match the pattern `dataset/sedov_hdf5_plt_cnt_*` using the plugin `FLASH` which is required to load the HDF5 datasets.

5.2 Creating the plots

5.2.1 Density

To create the plot of the density field, I am using the pseudocolor function and then setting the scale as logarithmic, since there are many more changes happening in the range of 0 to 1 compared to the whole range of values that is 0 to 70.

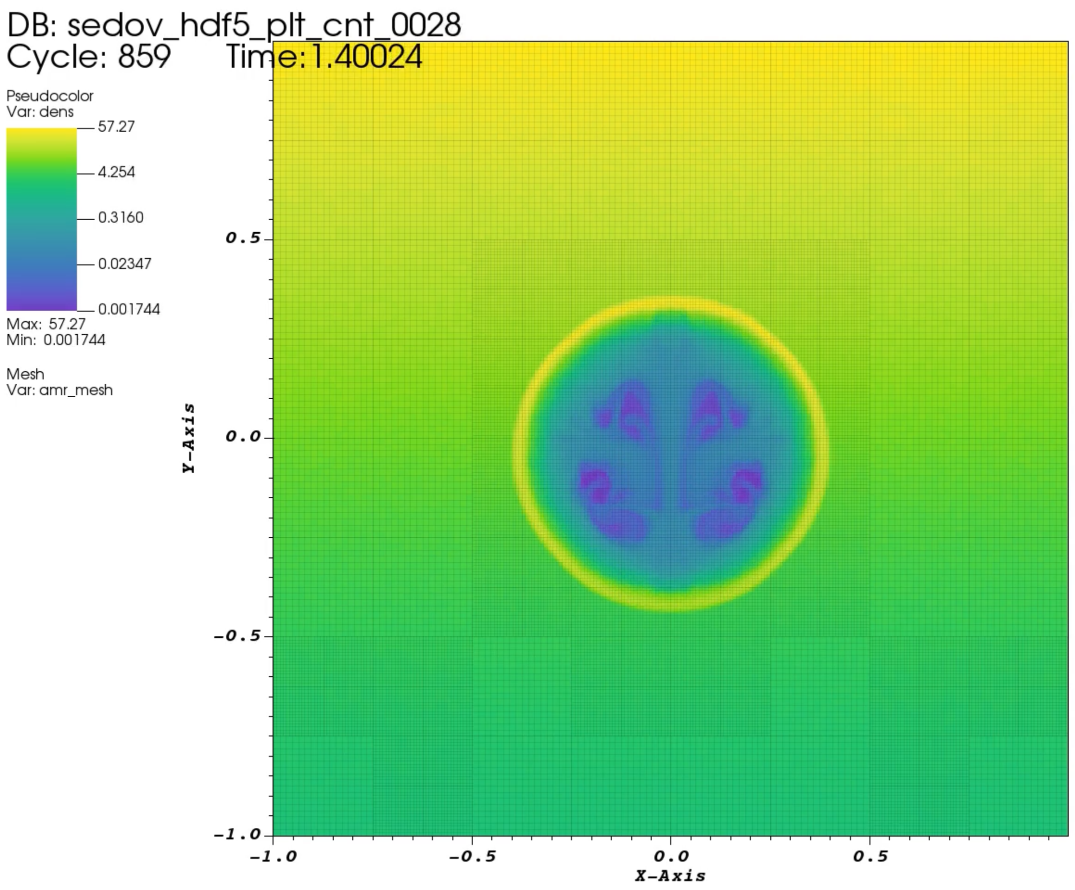
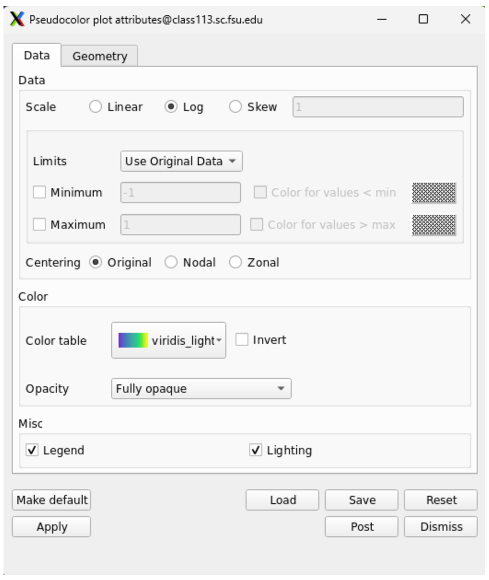


Figure 2: Density plot

In the above plot, you can also see a mesh that is overlayed on top of the pseudocolor plot. This mesh shows the structure of data density inside the HDF5 dataset. Since HDF5 is a dynamic dataset, it can change the resolution of data as per the requirement. In this case, the data near the center has more resolution compared to the edges.


Pseudocolor plot attributes@class113.sc.fsu.edu

Data | Geometry

Data

Scale: Linear [✓ Log] Skew 1

Limits: Use Original Data

Minimum -1 — Color for values < min

Maximum 1 — Color for values > max

Centering: [✓ Original] Nodal Zonal

Color

Color table: viridis_light — Invert

Opacity: Fully opaque

Misc

[✓ Legend] [✓ Lighting]

Figure 3: Density Plot Attributes

5.2.2 Pressure

Using the same technique I made the pseudocolor plot the Pressure field, the only difference is that it uses a linear scale.

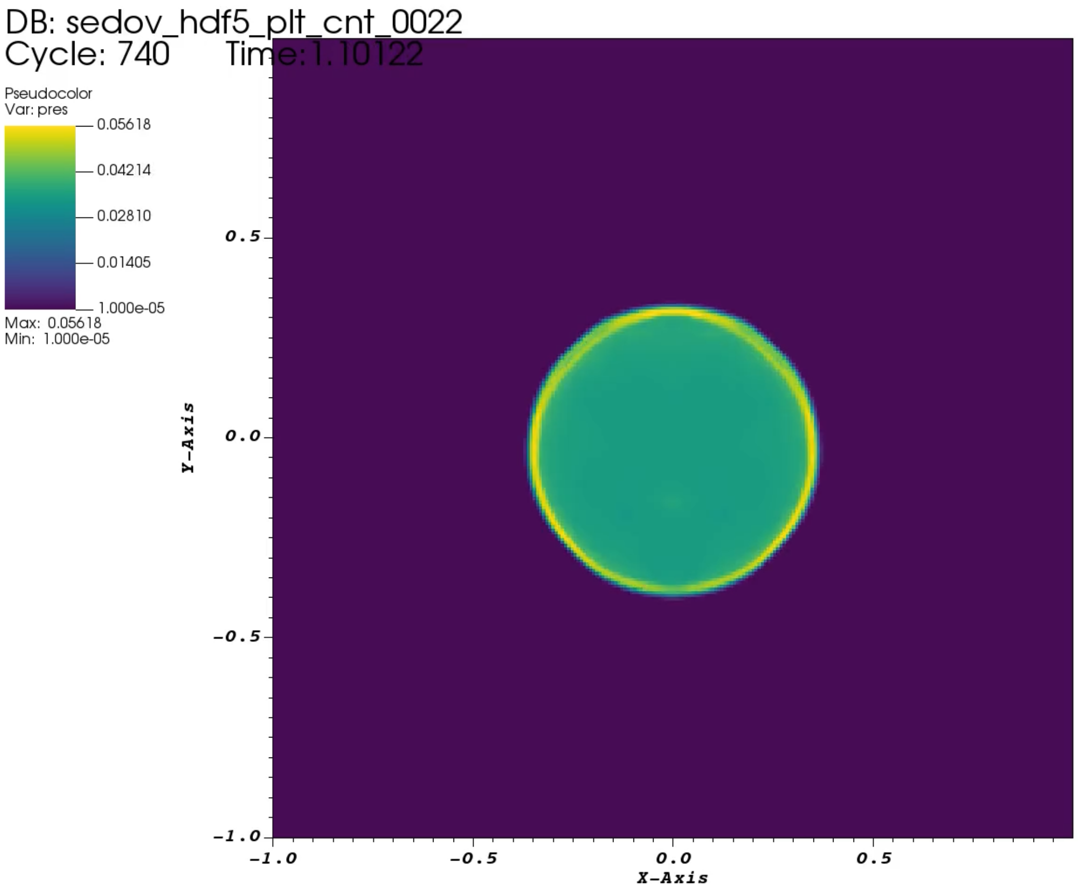
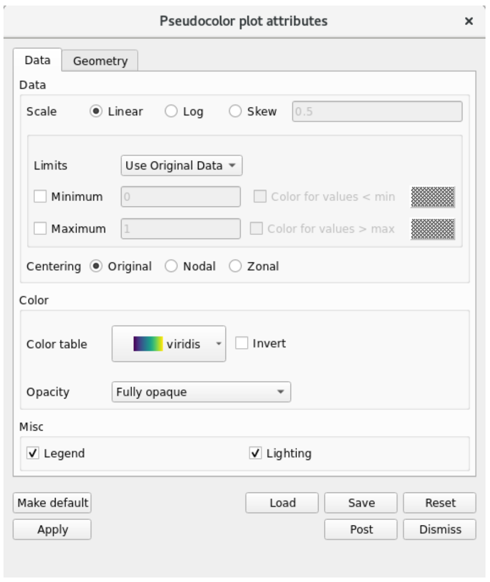


Figure 4: Attributes used


Pseudocolor plot attributes

Data | Geometry

Data

Scale [✓ Linear] Log Skew 0.5

Limits: Use Original Data

Minimum 0 Color for values < min

Maximum 1 Color for values > max

Centering [✓ Original] Nodal Zonal

Color

Color table: viridis ☐ Invert

Opacity: Fully opaque

Misc

☑ Legend ☑ Lighting

Figure 5: Pressure Plot Attributes

5.2.3 Temperature

For temperature, I have made two plots, one using the logarithmic scale and the other using the linear scale. In the logarithmic scale, we can see a significant amount of details about how the temperature is changing, but detecting the hotspot i.e. the small areas with the highest temperature is difficult. That is why I have also plotted a linear scale graph.

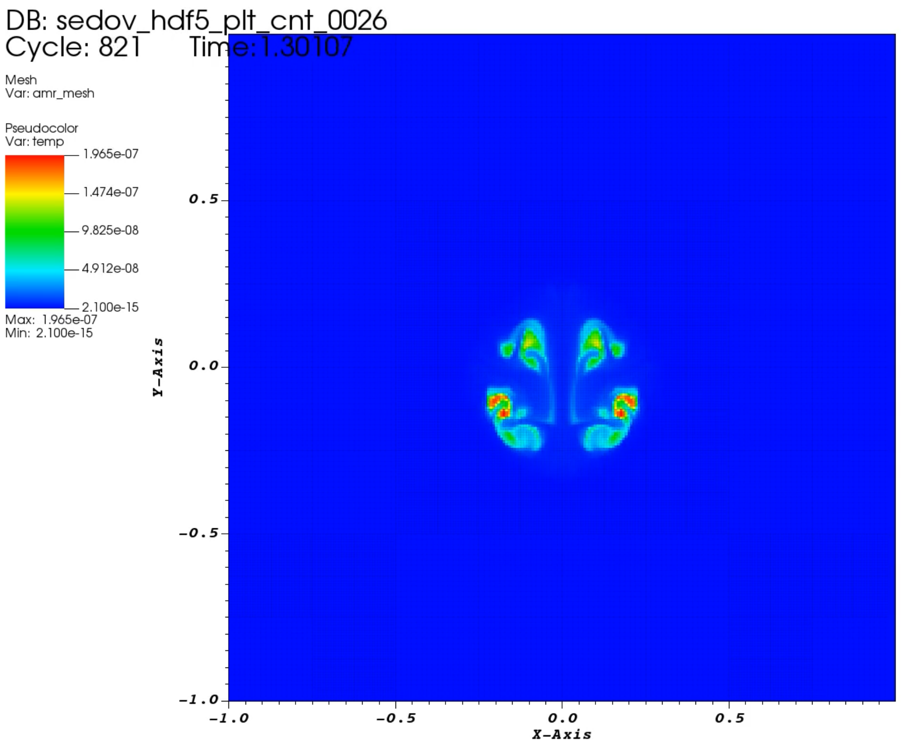


Figure 6: Temperature Linear Plot

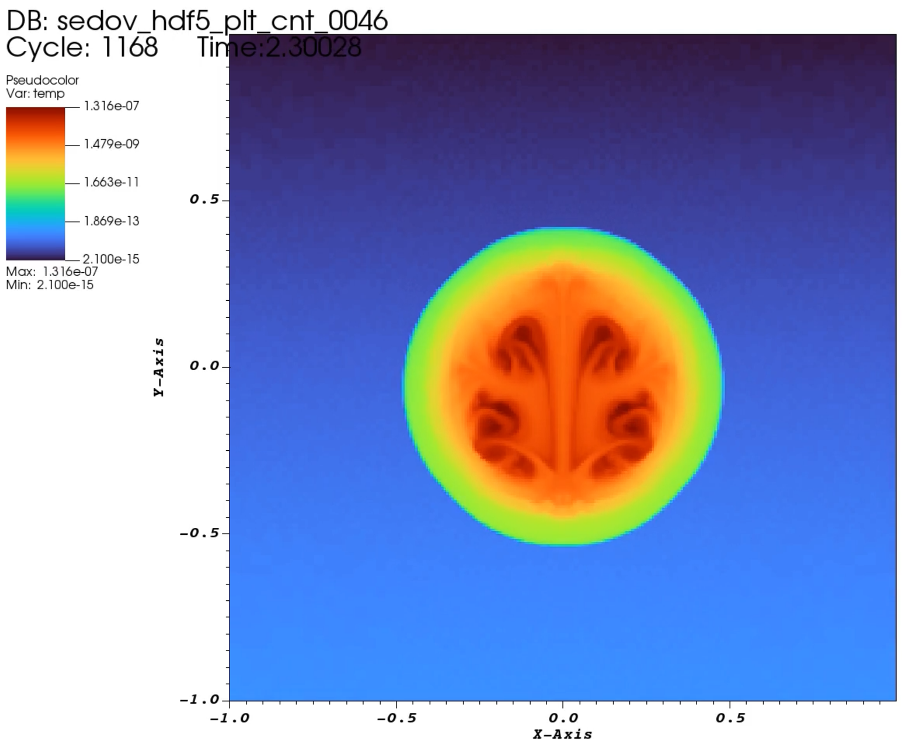


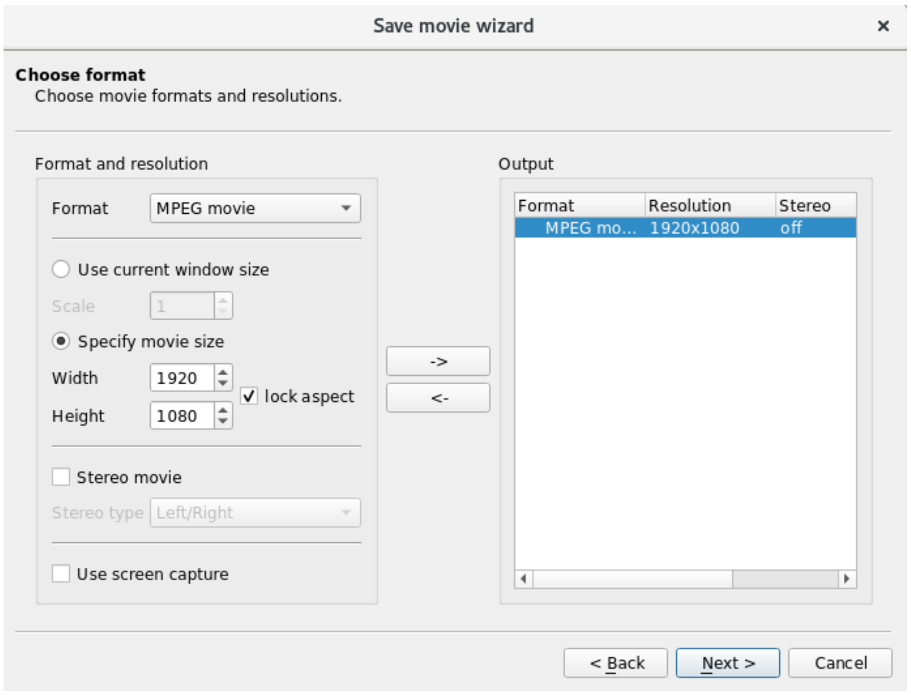
Figure 7: Temperature Logarithmic Plot

5.3 Creating the movies

Our data has 100 timeslices, which means everything we are plotting is changing with respect to time, to visualize this we can make a movie. Every frame in this movie will represent a time slice in the dataset.

5.3.1 Generating the frames

To generate all the frames for the movie I used the *save movie* option in Visit. This allows me to generate images for each timeslice. To generate the frames I used the resolution of 1920 × 1080 which is the standard resolution of modern monitors with a 16:9 aspect ratio.



Choose format
Choose movie formats and resolutions.

Format and resolution

Format: MPEG movie

( ) Use current window size
Scale: 1
(●) Specify movie size
Width: 1920
Height: 1080
[✓] lock aspect

[ ] Stereo movie
Stereo type: Left/Right

[ ] Use screen capture

Output

Format	Resolution	Stereo
MPEG mo...	1920x1080	off

Figure 8: Movie Wizard

Note: I have selected the format as MPEG movie but still the output from the Visit was images. I'm not sure what caused that.

5.3.2 Generating the movie

The images generated by the Visit were sorted and had the appropriate filenames. To make a movie from them I used `ffmpeg`. I used the following command to generate movies.

```
ffmpeg -framerate 15 -pattern_type glob -i 'reflec_pres/movie-0/*.png' -c:a
    copy -shortest -c:v libx264 -pix_fmt yuv420p pres_out.mp4
```

Here I have specified the frame rate at 15 frames per second and then specified the path to images as a regex pattern so that ffmpeg can load all of the images. Then I have specified to use H.264 encoding for the video, which is supported by most of the major video players. Also we are specifying that audio should be copied, which in this case no audio. And in the end I have specified the output file name that is `pres_out.mp4`

References

[1] *VTK - Getting Started* , *https://docs.vtk.org/en/latest/getting_started/index.html*. 2023.

[2] Lynn. *How to install VTK-9.1 on Ubuntu 18.04* , *https://freedium.cfd/https://medium.com/@yulin_li/how-to-install-vtk-9-1-on-ubuntu-18-04-a4530ce790da*. 10 July, 2022.

[3] David Gobbi. *Seg faults on Render() call* , *https://discourse.vtk.org/t/seg-faults-on-render-call/9803/1*. Nov, 2022.

[4] Perplexity. *ModuleNotFoundError: No module named 'vtkmodules* , *https://www.perplexity.ai/search/ModuleNotFoundE-No-module-n8tjIL8fTA2jZRmVtBmS5A*. 23 Apr, 2024.

[5] Chat-GPT. *Convert PNG to MP4* , *https://chat.openai.com/share/7a6b7868-bc7a-4c0d-be84-e3717b6d4ae8*. 24 Apr, 2024.