International Institute of Information Technology, Bangalore

Assignment 1 - 2D Scalar and Vector Field Visualization

Satvik Ramaprasad | IMT2016008

September 2, 2019

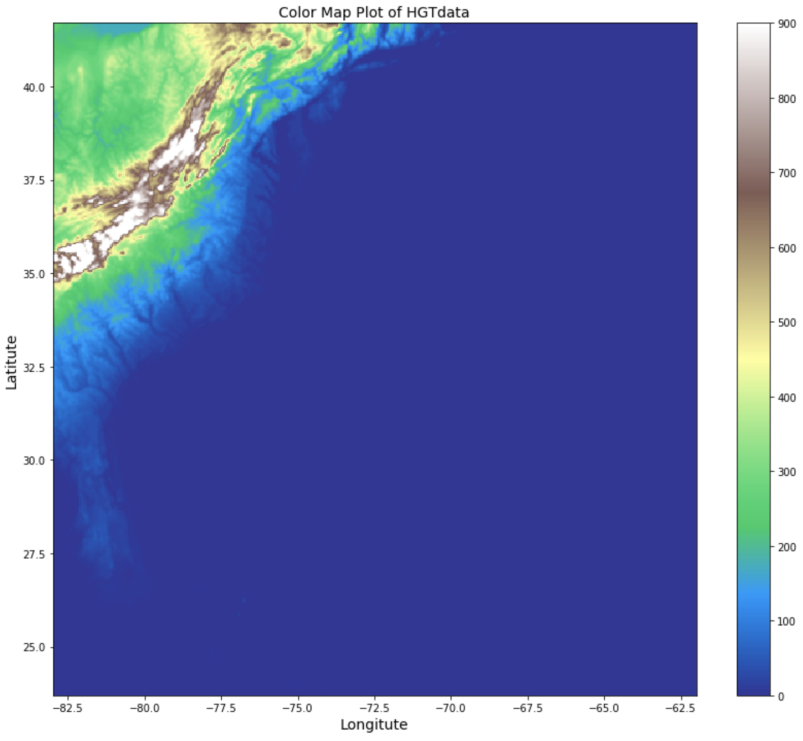

Figure 0.1: Topography Color Map Plot

Contents

1 Problem Statement

Description of Assignment

Dataset: Structured 3D dataset at `http://sciviscontest-staging.ieeevis.org/2004/data.html`

Consider scalar fields of cloud moisture mixing ratio, pressure, and temperature; and 2D vector field of (u,v), i.e. X and Y wind speeds. The volumetric data is in x, y, z coordinates, in 500x500x100 sized grids. We are interested in only studying the x-y planes (corresponding to longitude-latitude data). Hence, the first task is to aggregate data about the z-axis. Uniformly aggregate "n" layers in the z-axis, such that n is a manageable number for running an interactive application. Experiment values of n to be 5, 10, and 20. The aggregation operation can be averaging. Visualize the scalar fields and vector field for each (aggregated) x-y layer, looping through values of z. The scalar fields can be visualized using colour mapping, and height/elevation mapping, and vector field using the quiver/hedgehog plot. Use color and geometry visual encodings judiciously. For color mapping, visualize using 1-colour, 2-colour, 3-colour, and rainbow colour spectrum.

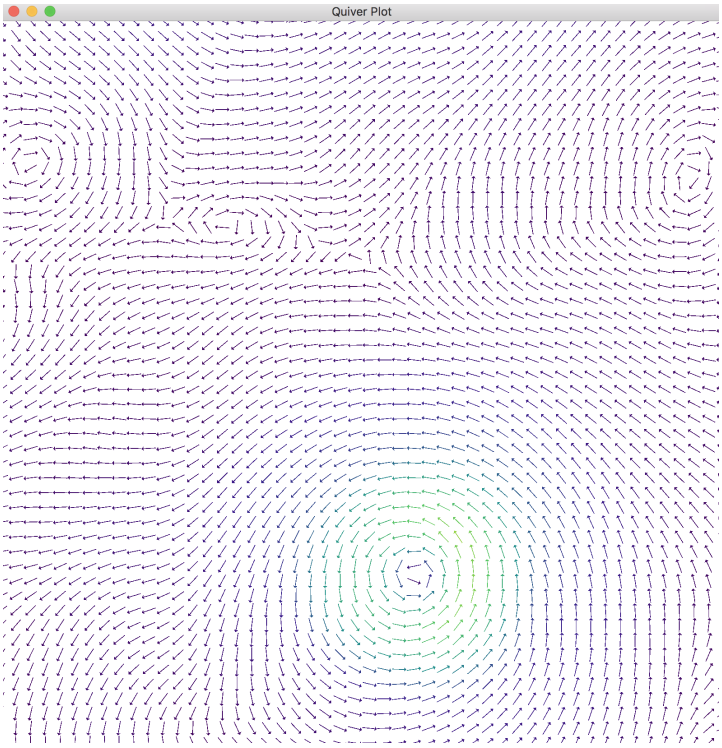

Figure 1.1: Wind Quiver Plot (Uf01+Vf01) height - 10 time - 1

2 Approach Taken

I decided to implement the algorithms for the visualizations in OpenGL 3.3 in addition to visualizing the data in Matplotlib (python). Note that things like legend, axis etc was not shown in the opengl implementations due to the unnecessary overhead involved.

2.1 Data Extraction

The dataset consisted of several variables including temperature, pressure and wind. The data was stored in a brick of floats. Except HGT Data, other variables were of the dimensions 500x500x100.

- x (Longitude) coordinate runs from 83 W to 62 W (approx)
- y (Latitude) coordinate runs from 23.7 N to 41.7 N (approx)
- z (Vertical) coordinate runs from .035 KM to 19.835 KM (100 equally spaced levels with delta=.2 KM)

Data extraction was slightly tricky as the data was stored in Big Endian format and in coloumn major instead of row major. In Python, numpy was used to extract and manipulate the data. In C++, the data was extracted by using native functions. Data Aggregation over different time steps was done in numpy and then exported for use in C++.

2.2 OpenGL

The following visualizations were implemented

1. ColorMap
2. ElevationMap
3. ContourPlot
4. QuiverPlot

In addition to these visualizations, *Animation* and *MultiPlot* was created to easily combine different visualizations. They will be explained in detail in section 3.

2.3 Matplotlib

Matplotlib was used to explore the data and try different visualizations and color mappings. This will be explained in detail in section 4.

3 OpenGL

3.1 Instructions

Building

To build the program, simply run `make` in the directory with the `Makefile`. To simplify demonstration, the `Makefile` also launches the program from the `bin` folder. Run `make clean` to cleanup.

Usage and Controls

Use *space* to loop through the visualizations. Use mouse control to explore the visualization.

3.2 Data structures

Several data structures were created to make manipulation of data easy. The data structures are as follows.

3.2.1 Data3D and Data2D

This data structure have the following features

- Read data into memory, given the data specification (little Endian and big Endian, data size)
- Easily manipulate individual data points
- Get data slice (Extract a Data2D slice from Data3D)
- Down Sample Data by averaging (Convert a 500x500 -> 100x100)

3.2.2 Color

This data structures have the following features

- Manipulation of Color Easily
- Ease of interpolation between colors

3.2.3 Point

This data structures have the following features

- Manipulation of Position Easily
- Ease of interpolation between points (color and position)

3.2.4 ColorMap

This data structures have the following features

- Insert Color Steps and store efficiently
- Query Color Value efficiently and interpolate between color values

3.2.5 Grid

This data structures have the following features. It is the primary datastructure used to finally render the data.

- Ease of manipulation of grid points (Position and Color)
- Render efficiently using shaders

3.3 Color Mapping

Color Map's implementation is straightforward. For each grid point, the color map data structure discussed in section 3.2.4 was used to compute the color. The Grid datastructure as discussed in section 3.2.5 will render efficiently.

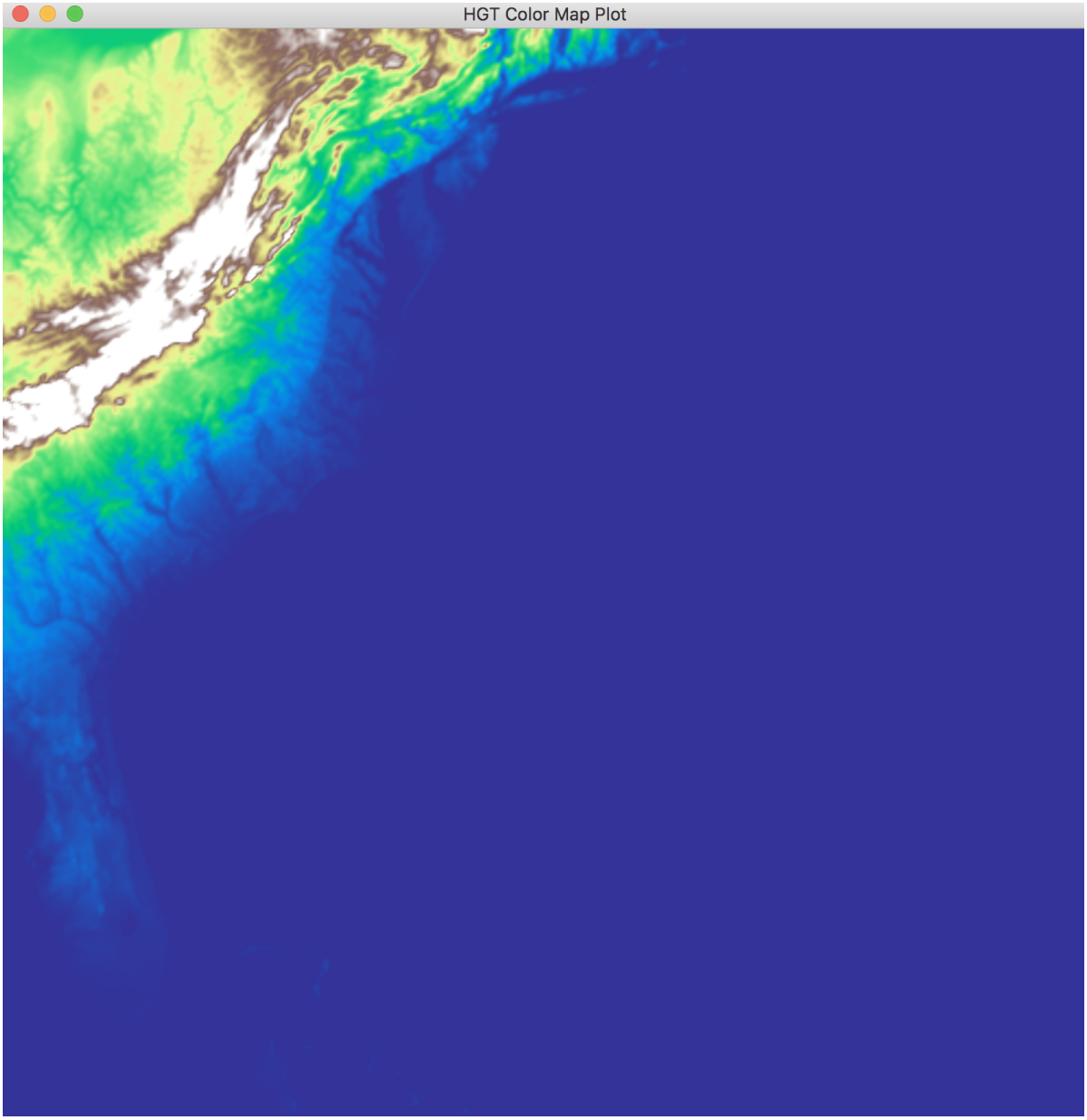

Figure 3.1: HGT Color Map Plot

3.4 Elevation Mapping

Elevation Mapping adds on to color mapping. In addition to color, the depth of the grid points also are changed to give depth. Figure 3.2 illustrates a visualization of pressure data using elevation mapping. Note that the eye of the hurricane can clearly be seen as a valley.

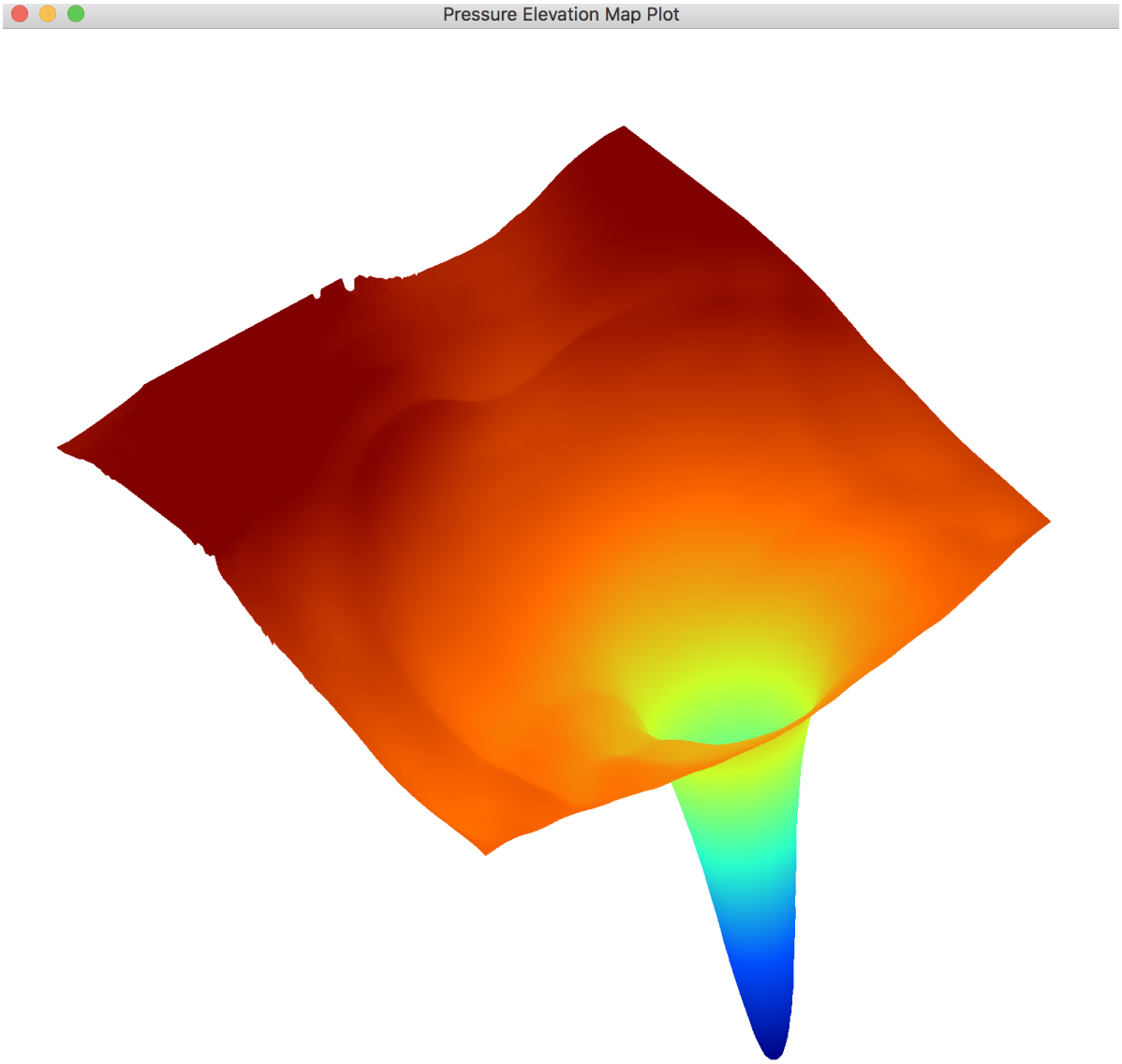

Figure 3.2: Pressure Elevation Mapping (Pf01). Height - 80, time - 1

3.5 Contour Mapping

Contour mapping was implemented using marching squares algorithm. Figure 3.3 illustrates a visualization of HGT data using contour mapping.

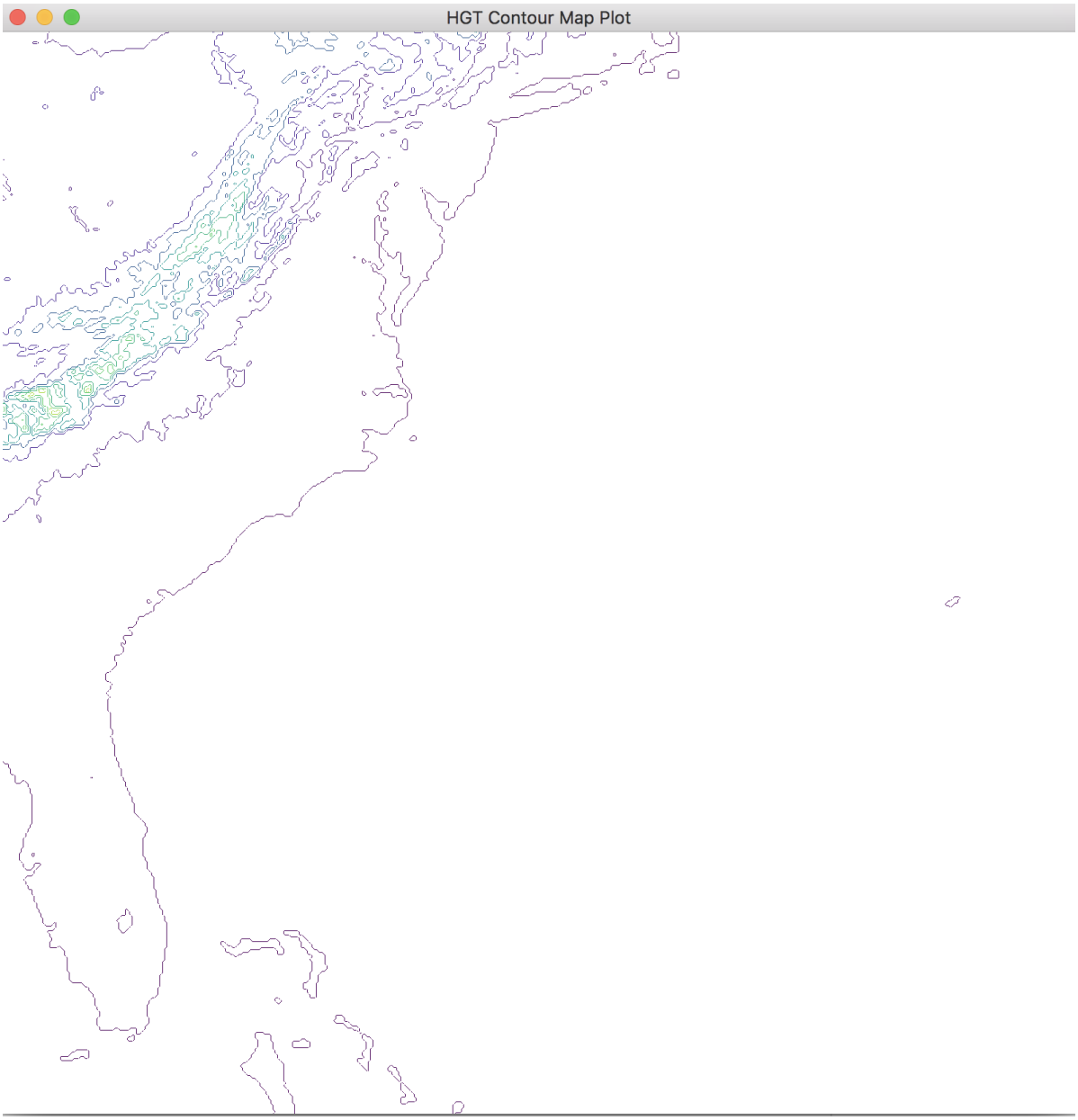

Figure 3.3: Contour Mapping of HGT data.

3.6 Quiver Plot

Quiver Plot was implemented by normalizing the u-v directions and using color to indicate velocity. Figure 3.4 illustrates a visualization of wind data using quiver plotting. Note that while there seems to be many hurricanes in the picture, only one of them has high velocity indicating high wind speeds.

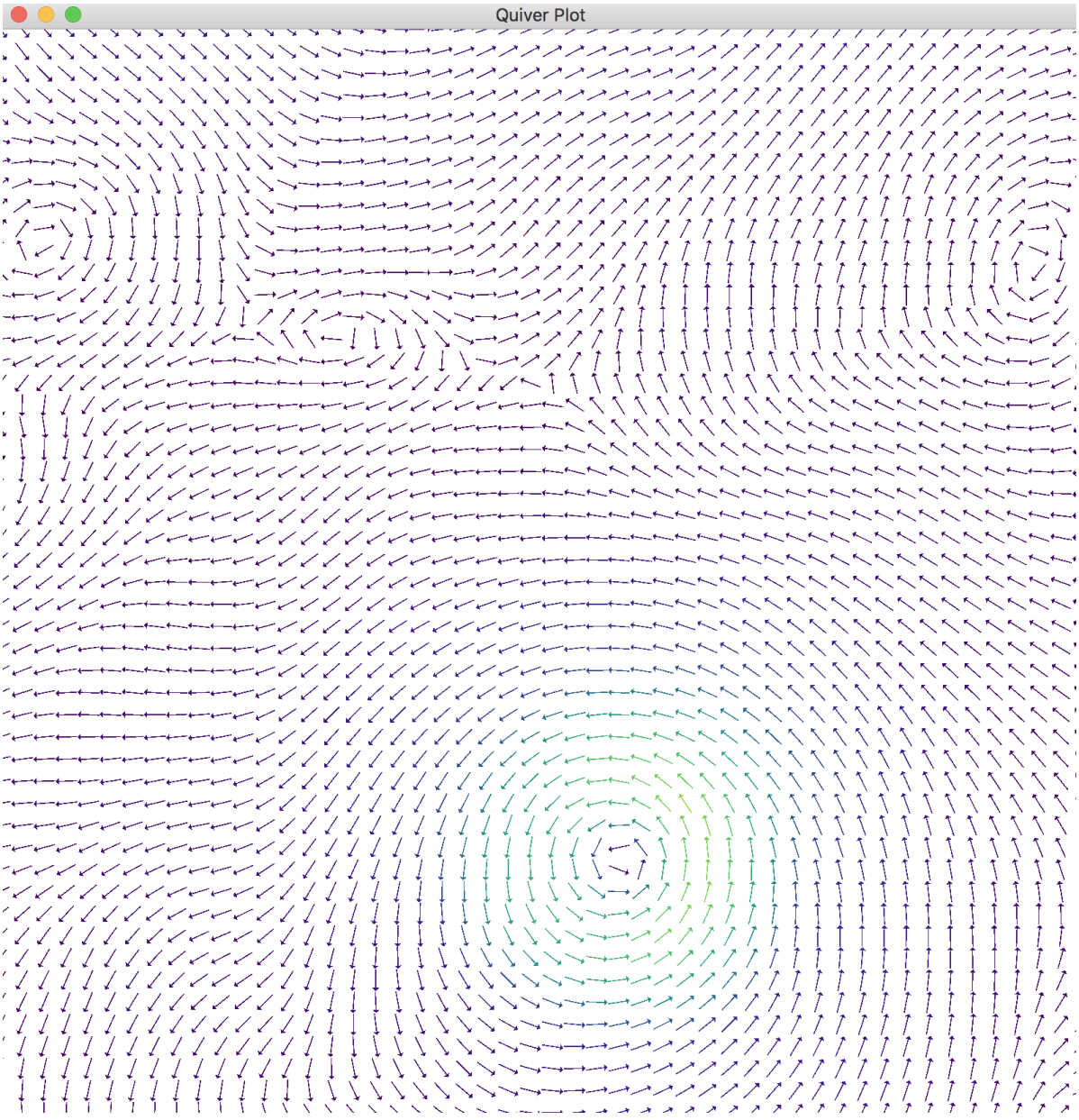

Figure 3.4: Wind Quiver Plot (Uf01 + Vf01), height = 1, time = 1

3.7 Animation

Any of the previous visualizations can be combined to provide an animation. This is done by adding multiple frames and the animation class loops through the frames. Note that it is not possible to show animation in the report. Please check `images/opengl/animation.mp4` for a preview.

3.8 Multi Plot

Any of the previous visualizations including animations can be combined one over the other to be visualized simultaneously. Opacity of the visualizations can be varied to get best results. Figure 3.5 illustrates a visualizations of HGT Data, Wind Data and Rain Data.

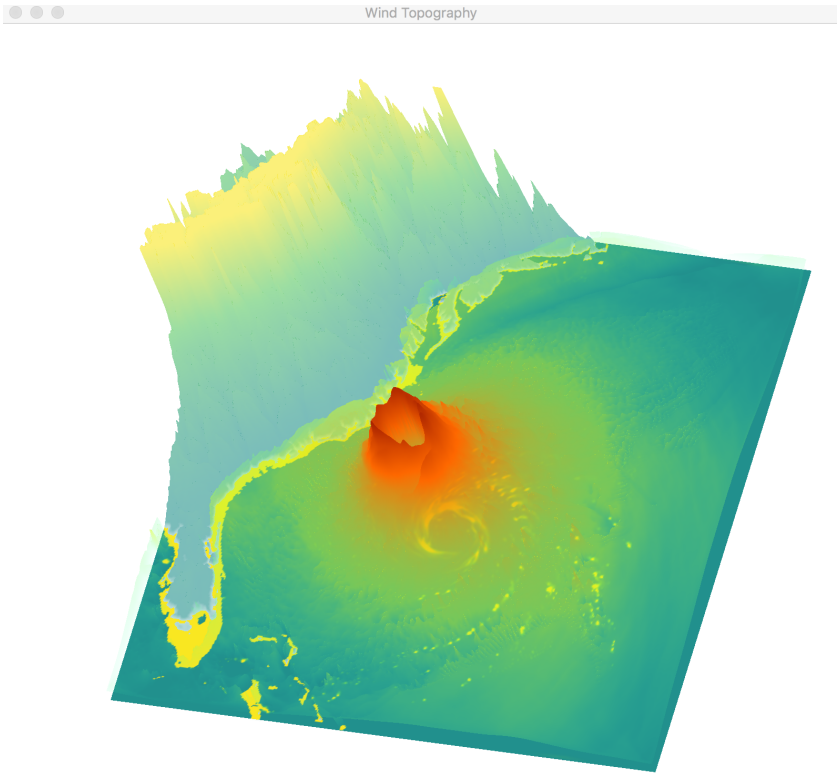

Figure 3.5: Wind Quiver Plot (Uf01 + Vf01), height = 1, time = 1

4 Matplotlib

All visualizations done in matplotlib can be seen in `matplotlib/visualizations.html`. The following are the highlights of the visualizations.

4.1 Color Maps

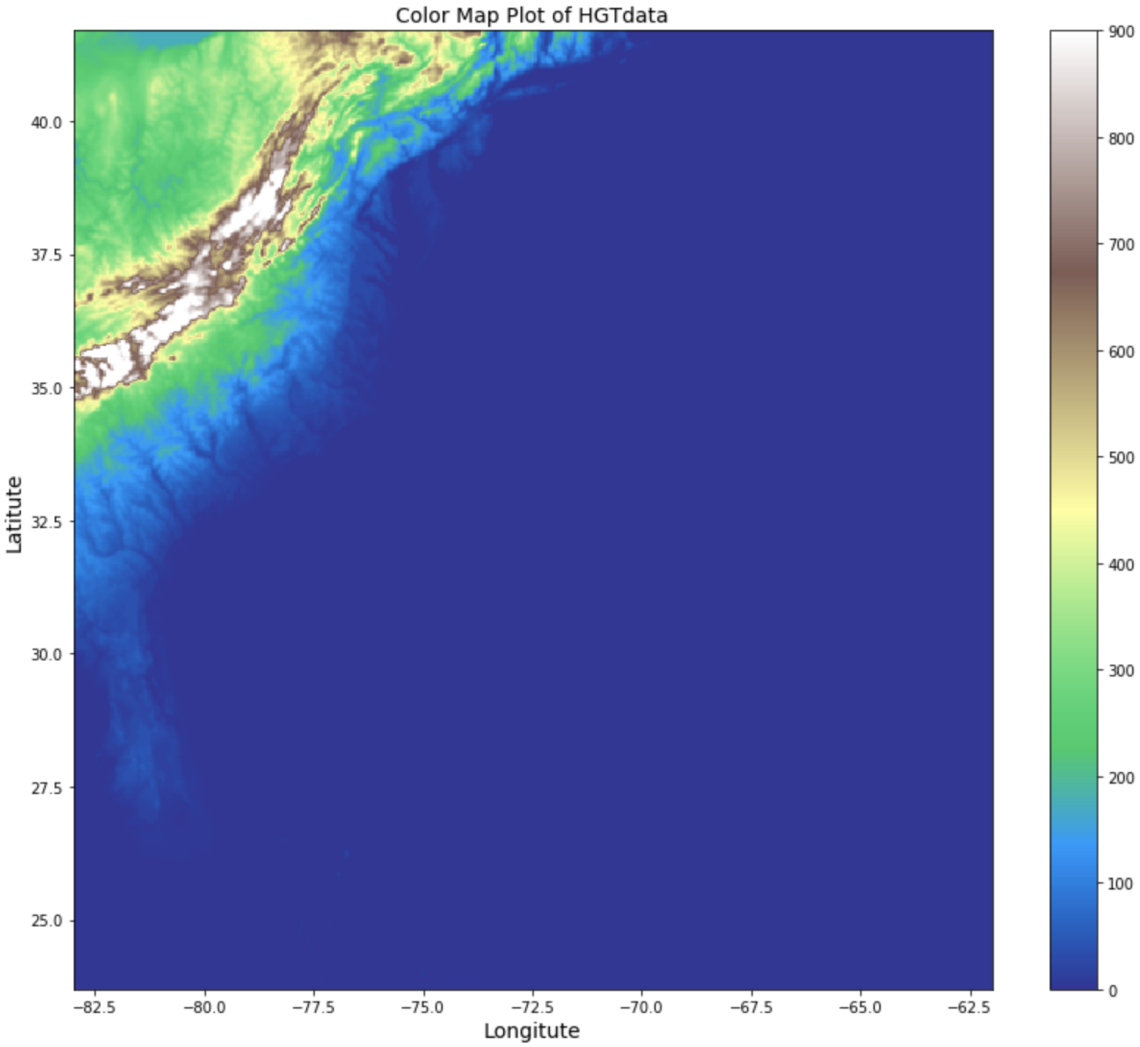

Figure 4.1: HGT Color Map

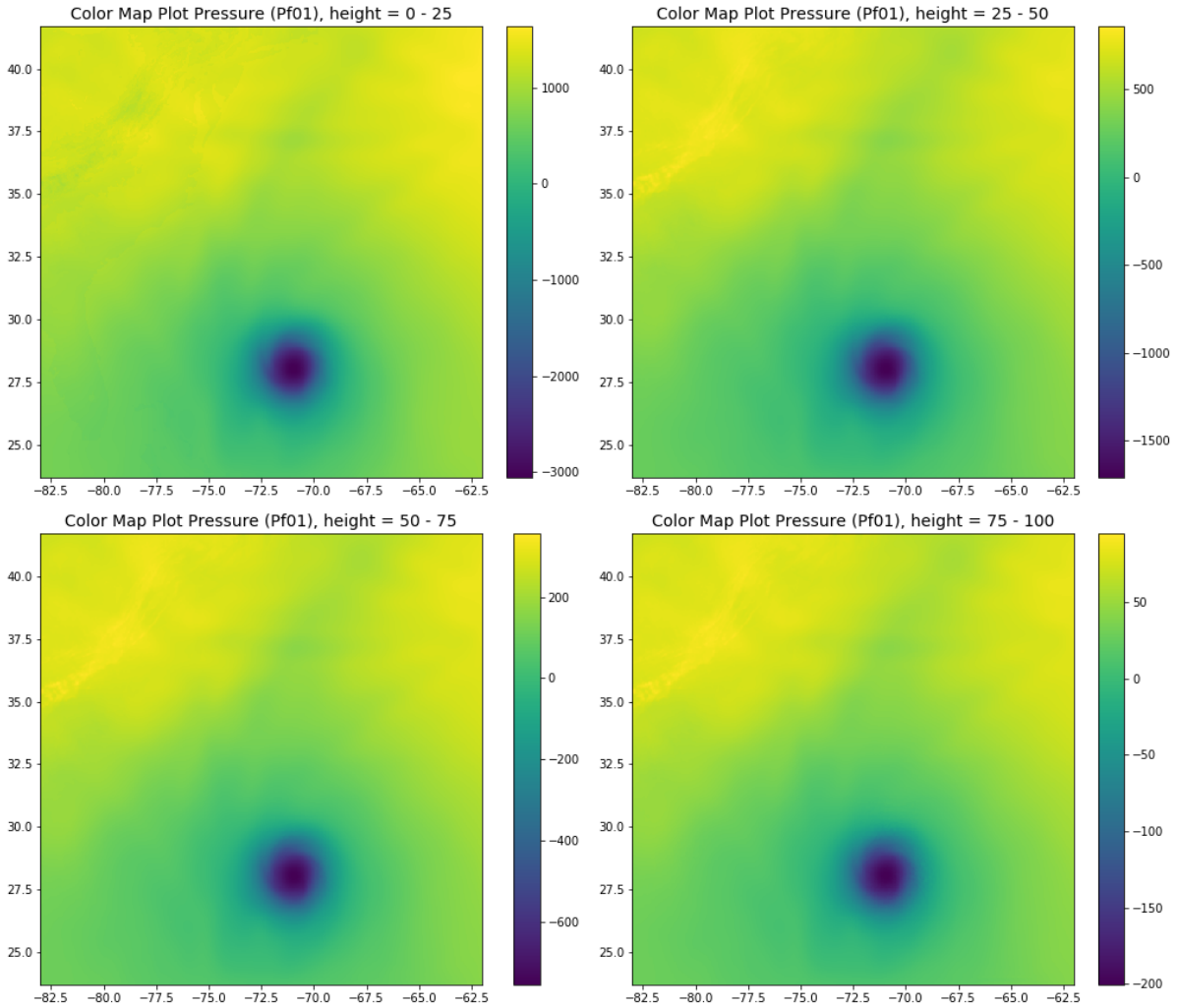

Figure 4.2: Pressure Color Map (Pf01)

4.2 Contour Maps

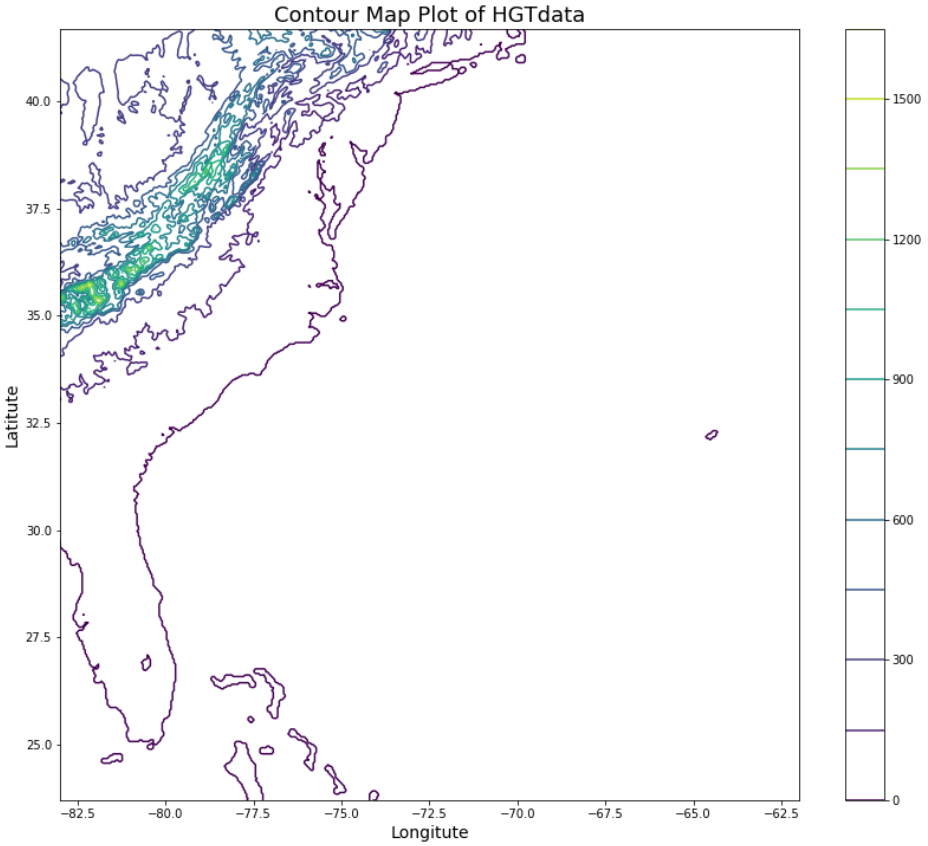

Figure 4.3: Contour Map of HGT Data

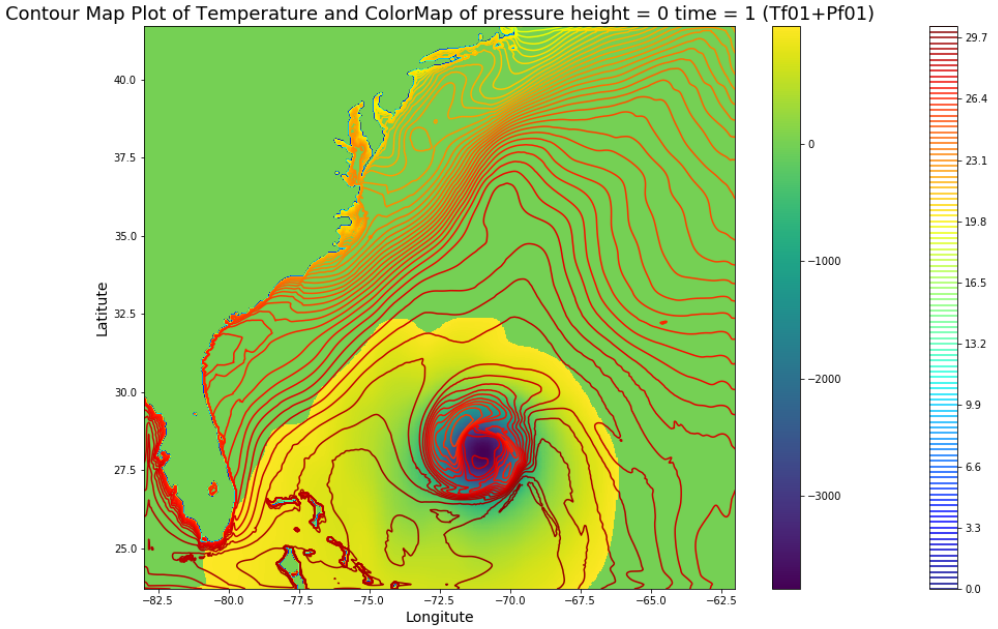

Figure 4.4: Color Map of pressure and contour map of temperature (Pf01 + Tf01)

4.3 Elevation Maps

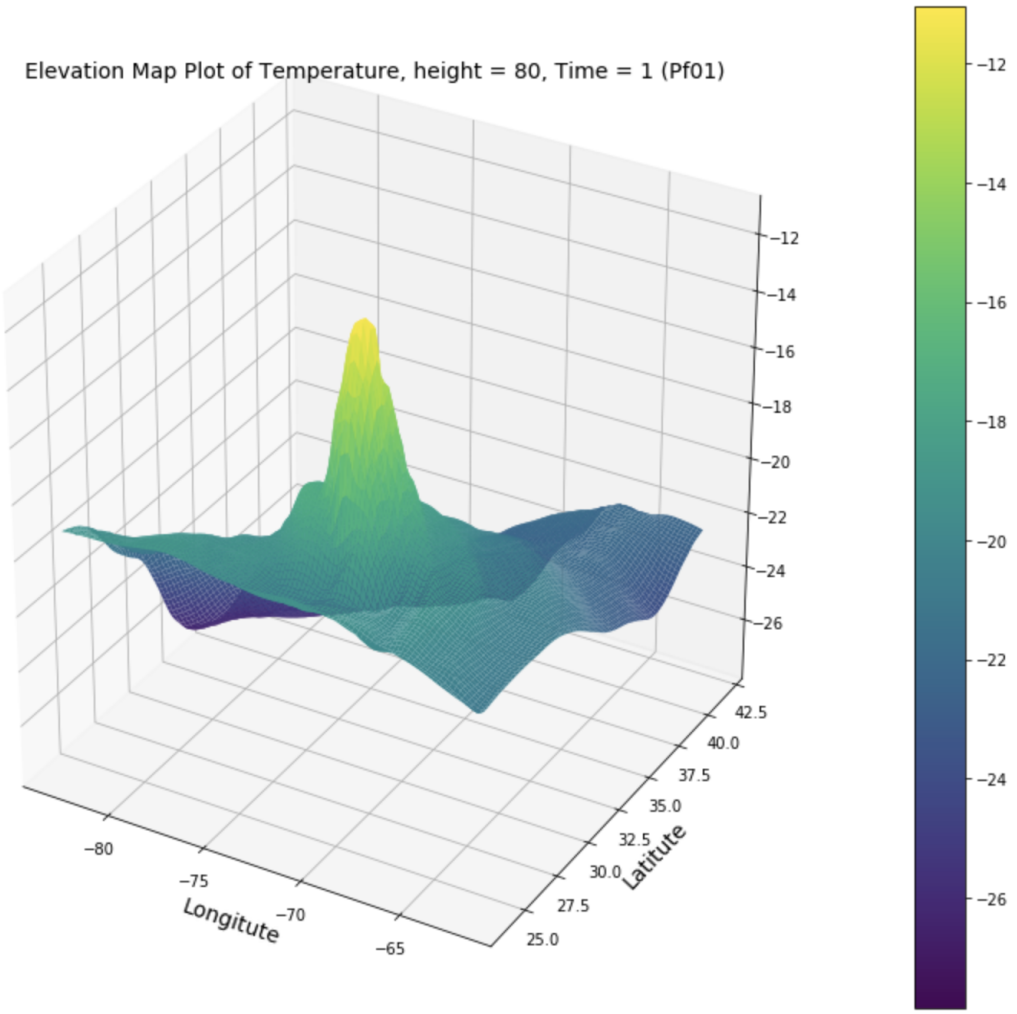

Figure 4.5: Temperature Elevation Plot

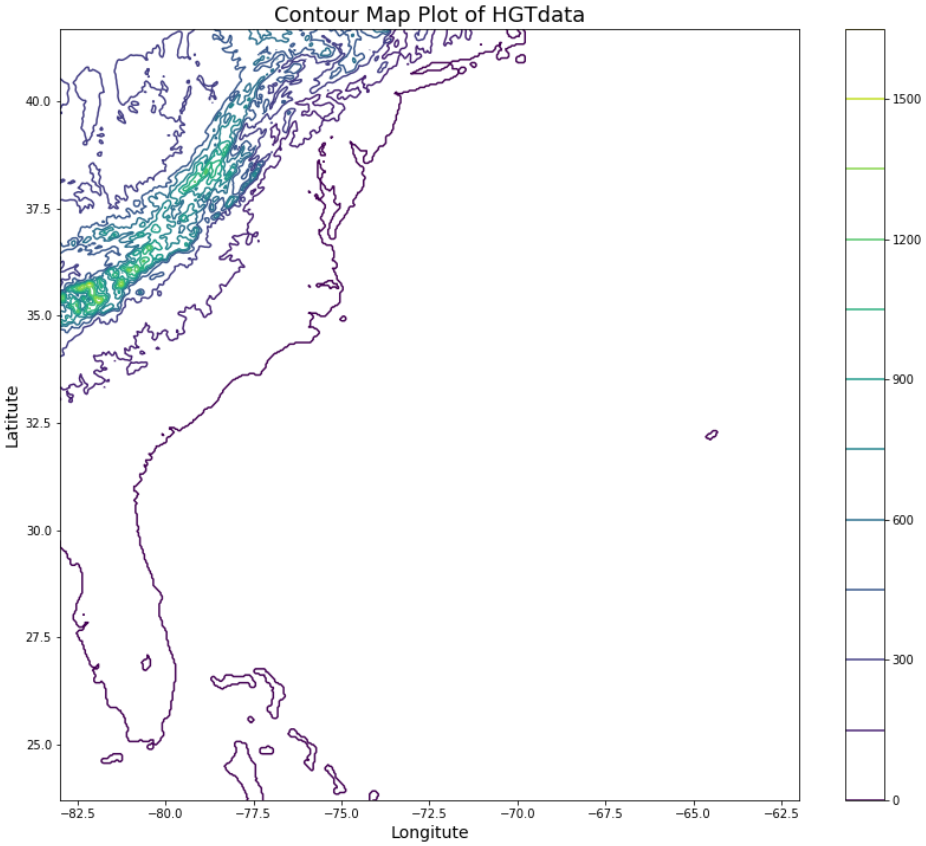

Figure 4.6: HGT Elevation Plot

4.4 Quiver Plots

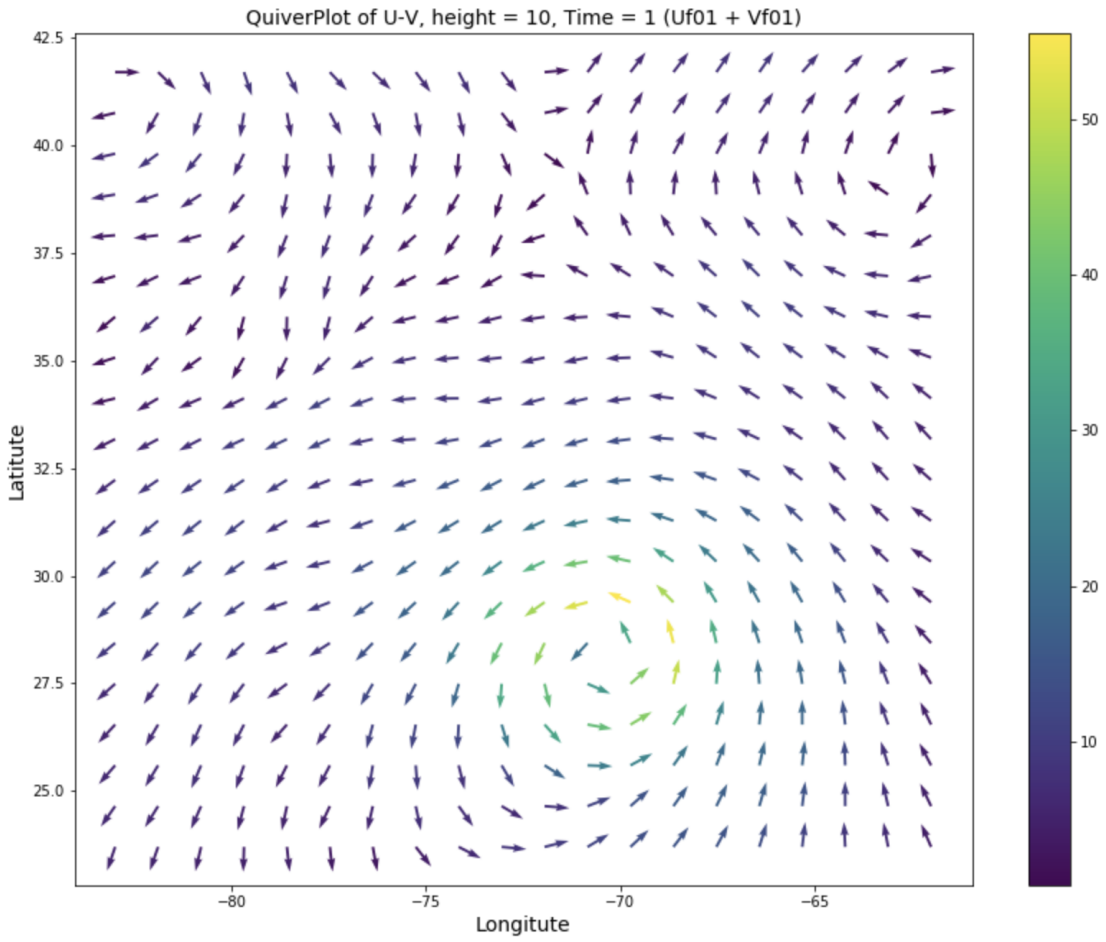

Figure 4.7: Wind Quiver Plot

4.5 Streamlines

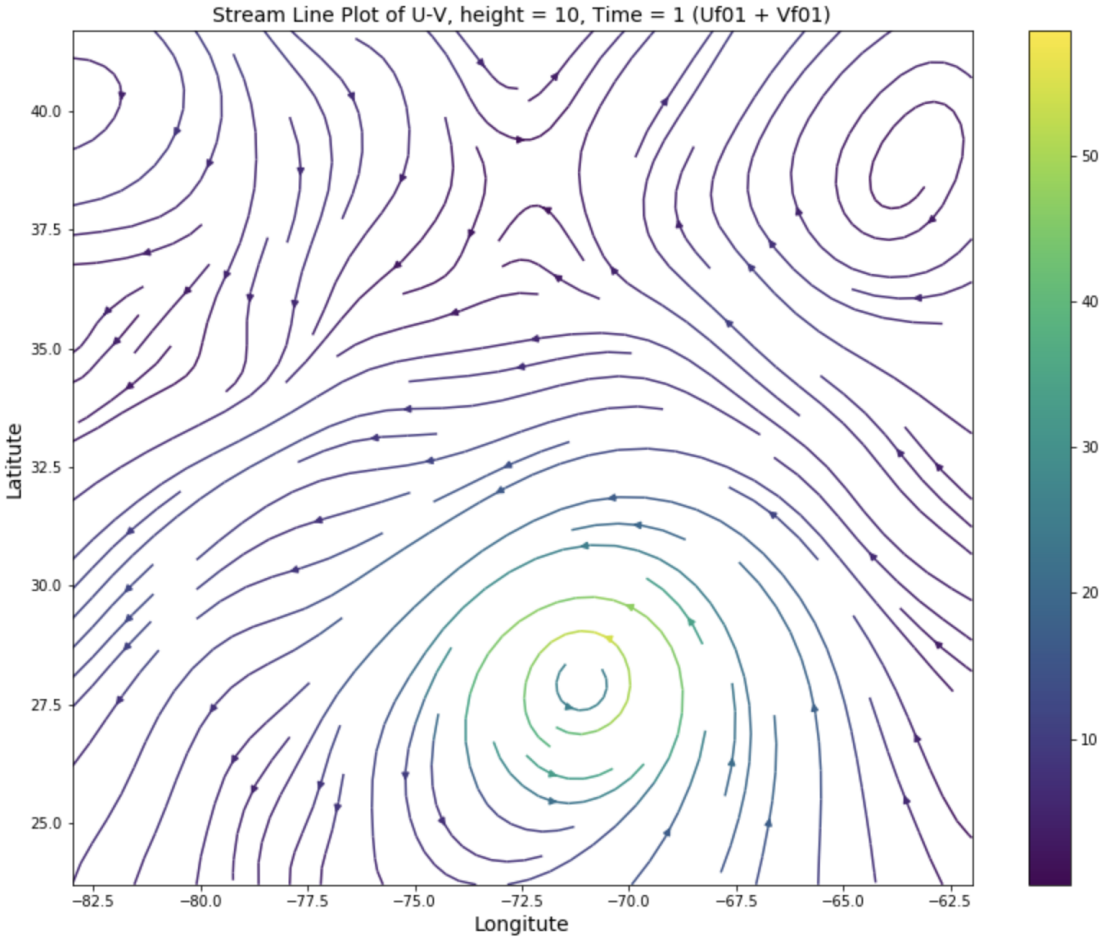

Figure 4.8: Wind StreamLine Plot

5 Questions

1. What are your inferences on the value of n? Did large values of n lead to loss of information? If yes, which variable(s) was/were affected?

   Ans: In my experiments, I noticed that the large values of n smoothened out some variations with height which is expected as we are aggregating by averaging. Another issue I noticed was that the range of values for some variables changed drastically with height. For example temperature at sea level > 20 but at higher levels, the temperature was sub zero. Therefore depending on the n values, the averaging lead to loss of information. I noticed this primarily with temperature, pressure and wind values.

2. What inferences can you make on the choice of your colour spectra? Did you find any artifacts introduced by the rainbow colour spectrum?

   Ans: In my experiments, I conclude that it is generally good to use 1 or 2 Color Color Maps. Sometimes 3 Color Colormaps are also good. ColorMaps with higher number of colors such as the rainbow color spectrum generally has many issues. Firstly there will be abrupt change in color brightness (as perceived by the eye) which will lead to false conclusion that there are boundaries in the data. Secondly the color variation between the spectrum is not very smooth and change very rapidly between the color boundaries. Lastly to the human eye, red appears closer to blue than yellow and green, however the rainbow colormap has the order reversed.

Figure 5.1: Rainbow Vs Parula Color Map

6 Summary

I have successfully implemented color maps, contour maps (marching squares algorithms) , quiver plots and elevation maps in OpenGL. Using OpenGL I was able to render reasonably complex animations efficiently. I was able to explore the data and try different color maps in matplotlib.

References

[1] Learnopengl: Basic lighting. https://learnopengl.com/Lighting/Basic-Lighting.

[2] Learnopengl: Matrix transformations, coordinate systems. https://learnopengl.com/Getting-started/Coordinate-Systems.

[3] Matplotlib. https://matplotlib.org/users/interactive.html.

[4] Matplotlib documentation. https://devdocs.io/matplotlib~3.1.

[5] Virtual trackball - brocku. https://www.cosc.brocku.ca/Offerings/3P98/course/OpenGL/glut-3.7/progs/examples/trackball.c.

[6] Virtual trackball - khronos. https://www.khronos.org/opengl/wiki/Object_Mouse_Trackball.