CS732 : Data Visualization
Technical Report for Datathon-2

Neetha Reddy
International Institute of Information Technology Bangalore
IMT2018050
Neetha.Reddy@iiitb.ac.in

***Abstract*—This technical report consists of a brief overview of methodologies employed in the visualisation of Ionization front instability simulation data set. Further, this report attempts to draw inferences from the results of the above mentioned visualizations.**

I. Introduction

The ionization front instability simulation data set submitted by Mike Norman and Daniel Whalen to the 2008 IEEE Visualization Design Contest is being used for the scope of this paper. The dataset contains *10 scalar fields*:

1) Total particle density (# of particles/cm^3)
2) Gas temperature (degrees Kelvin)
3) H mass abundance
4) H+ mass abundance
5) He mass abundance
6) He+ mass abundance
7) He++ mass abundance
8) H- mass abundance
9) H_2 mass abundance
10) H_2+ mass abundance

and a *velocity field*.

II. Methods and Visualizations

We have visualized all the data using *numpy*, *matplotlib*, *plotly*, *pandas* and *imageio*.

A. *Streamline Visualizations*

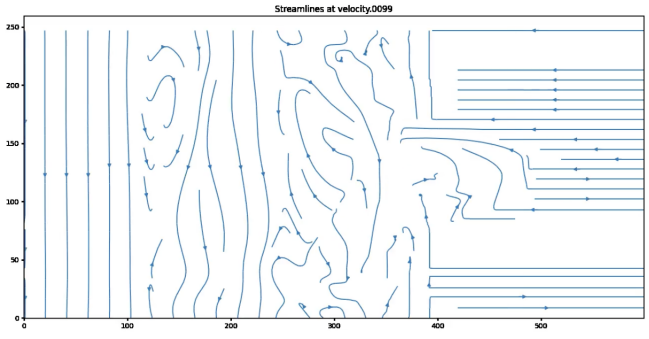

Fig. 1. Streamlines

Streamline visualizations, like quiver plots, are used to display 2D vector fields. The streamplot() function from *matplotlib* was used to generate the plots. As mentioned in the previous assignment, the X-Y plane is chosen with Z = 40 over 11 timestamps across the 200 timestamps in the dataset with a gap of 20 timestamps, i.e. 0000, 0019, 0039,..., 0199. This was done to ensure that necessary information is communicated over the course of the dataset at a much faster speed (sampling of the dataset). Each data file has 600*248*248 number of lines and each of these lines represent a particular 3D point. This means that X indices value most rapidly, followed by Y indices and then Z indices. Therefore, X-Y plane was chosen with Z=40 due to its ease of indexing and large plane size which allows us to draw meaningful inferences (as explained in the Datathon-1 report).

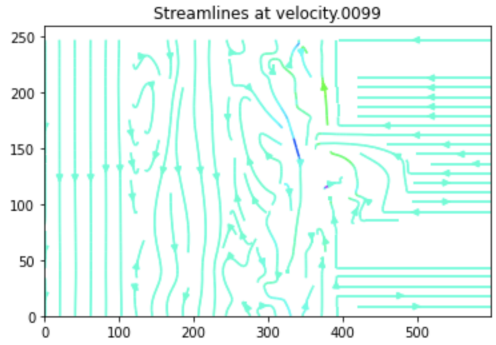

Fig. 2. Color mapped Streamlines of velocity

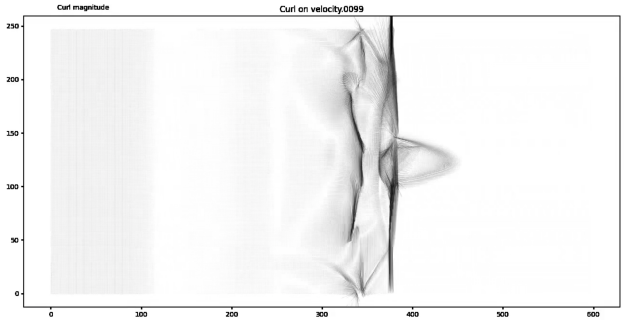

Fig. 3. Quiver plot of curl of velocity

On comparing these streamline plots with the quiver plots, we observe that the information about the magnitude of velocities and their change is lost in quiver plots (Fig. 1). Meanwhile, streamline plots have colour attributes which

can be changed to better denote these changes in velocity magnitudes and arrive at more informative visualizations. The hsv cyclic color map has been used since it better denotes the changing velocity magnitude. Further, the curl magnitude of the velocity vectors had to be calculated for quiver plots whereas we could directly pass the velocities to the *streamplot()* function. (Fig. 2 and Fig. 3 show the streamline plot and quiver plots at timestep 99)

Note:- These visualizations were done in a Kaggle notebook. Due to memory limit exceeded error, the computations for streamlines were animated in batches of 3 and then merged.

B. Iso-surface extraction

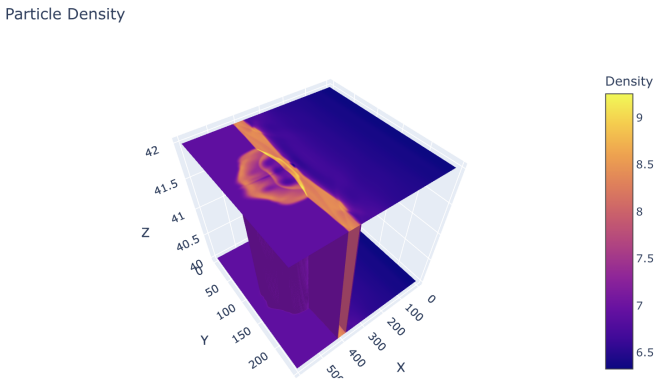

Fig. 4. Log of total particle density.

Iso-surface extraction is done to contour maps to scalar fields. One of the most well known algorithms to extract iso-surfaces is the Marching Cubes algorithm. Here, we have chosen to show the extraction for the log of the particle density field, since it is easier to draw inferences about how the wave front progresses and diffuses. We have chosen the timestamp 119 for all the subsequent iso-surface visualizations, since the chosen plane interval Z = 40 to Z = 42, marks the beginning of clearer disturbances in the ionic wave front (Fig. 3). We have plotted five iso-surfaces at timestep 119 and the perpetually uniform sequential colormap plasma was used for the same. The yellows/oranges denote regions having relatively higher particle densities whereas the darker blues/purples denote regions having relatively lower particle densities. We can clearly see in Fig. 3, how the wave front has diffused its particles from the starting observations(this has been discussed in Datathon-1).

These iso-surface plots have been generated using the *Isosurface* function from the plotly library. Since rendering the entire volume data using the function was taking too much time and computation power, only the points in the plane interval Z = 37 to Z = 43 were taken into consideration and visualized accordingly. We set the surfacecount attribute

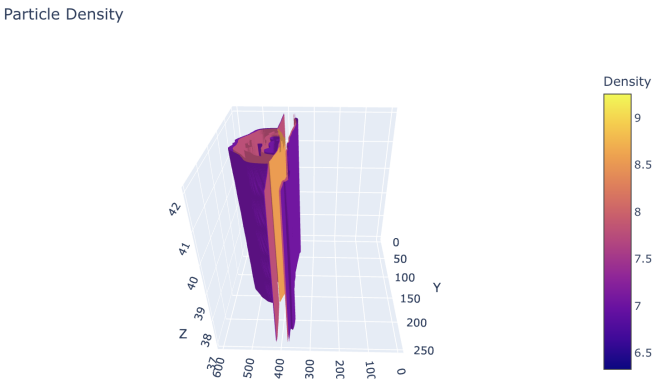

Fig. 5. Iso-surface count = 5, opaque.

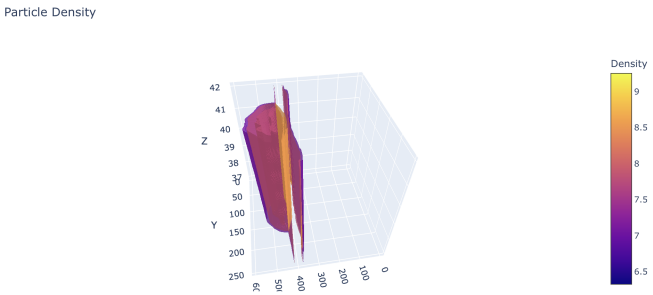

Fig. 6. Iso-surface count = 5, opacity=0.75.

in the *Isosurface* function to be 5, in order to extract five iso-surfaces while the isomin and isomax values were fixed as the mininum and maximum values encountered across the plane considered to try and spread out the generated iso-surfaces. Further, the transperancies of these iso-surfaces have also been experimented upon using the *opacity* attribute. Better results were observed when using 0.8 opacity (usage of the surfacefill attribute may or may not have helped).

C. Oblique Slicing

Till now, we have only visualised 2D planes parallel to a certain axis. However, it might be useful in the future to

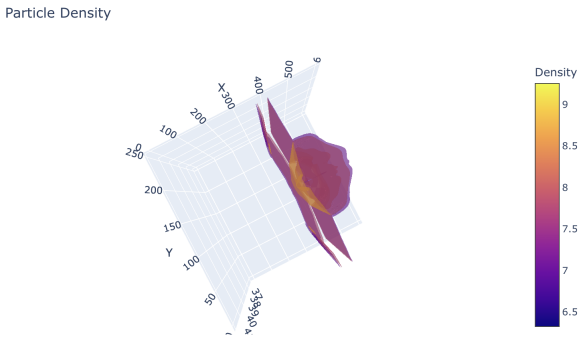

Fig. 7. Iso-surface count = 5, opacity=0.6.

isolate any desired 2D plane from a given volumetric dataset.

Procedure: We begin by assuming the equation of the plane to be $ax + by + cz + d = 0$. We then considered valid pairs of (y, z) (here since there are 248 of each, we considered points with a gap of 3, giving us 84 points each respectively) and for these valid pairs, we found all x-coordinates on the plane. Using the (x,y,z) points thus obtained, we can find the particle density at that point in the volume and then use *plotly* surface object to plot the surface. If the x-coordinate obtained is not an integer, we used a linear interpolation of the coordinate with respect to its floor and ceil and then proceeded to include it in the plotting process. For moving along the normal direction, we are changing the value of the constant d in the plane equation $ax + by + cz + d = 0$.

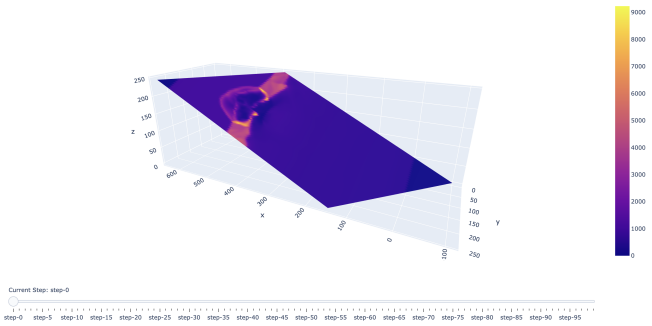

Fig. 8. Oblique slicing, step 0

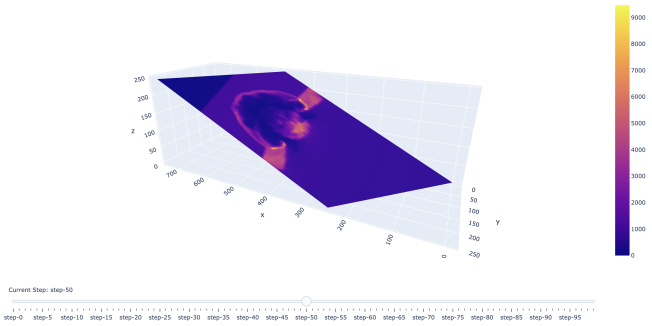

Fig. 9. Oblique slicing, step 50

Fig. 8, Fig. 9 and Fig. 10 depict the oblique slices of the plane $1x + (-1)y + (-2)z + d = 0$ where d ranges from -100 to 100 with 2 as the step size. In Fig. 9, we can observe that on dragging the slider, towards the end, the plane starts turning dark blue from the top. This happens because we are restricting the x-coordinates to be between 0 and 600 so anything that doesn't fall in this range gets equated to zero (this is the logic used in the code).

III. Inferences

Streamline plots vs Quiver plot of curl Answered previously in II A. Streamline Vizualizations

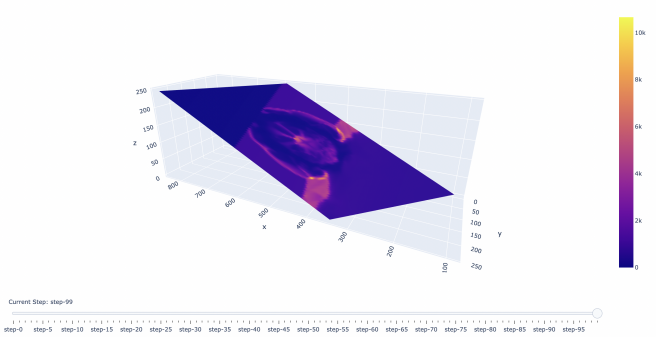

Fig. 10. Oblique slicing, step 99

Iso-surfaces: Did the use of transparency improve the visualization? Did the data have isosurfaces as layers that could exploit the use of transparency for improving visualization?
Yes, the usage of transparency helped improve visualisation since it allowed us to observe the stark differences in the colours between the different iso-surface layers, which we are attributing to particle diffusion. The existence of iso-surfaces as layers allowed us to exploit the transparencies of these layers in order to make the visualization more informative.

IV. References

1) IEEE Visualization 2008 Design Contest.
2) 3D Isosurface Plots in Python
3) Slice in volumetric data, via Plotly
4) Sliders in Python