Assignment 11

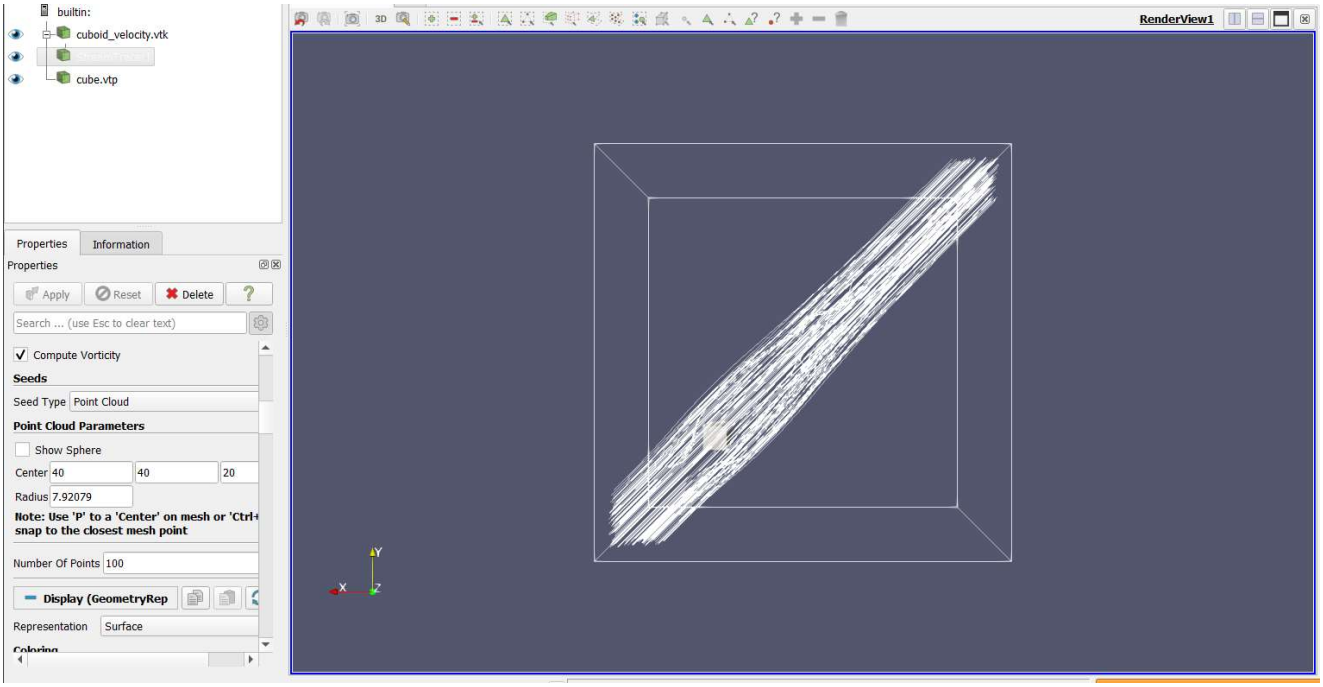
Group Members

Abdul Wahab Madni, Hamza Badar

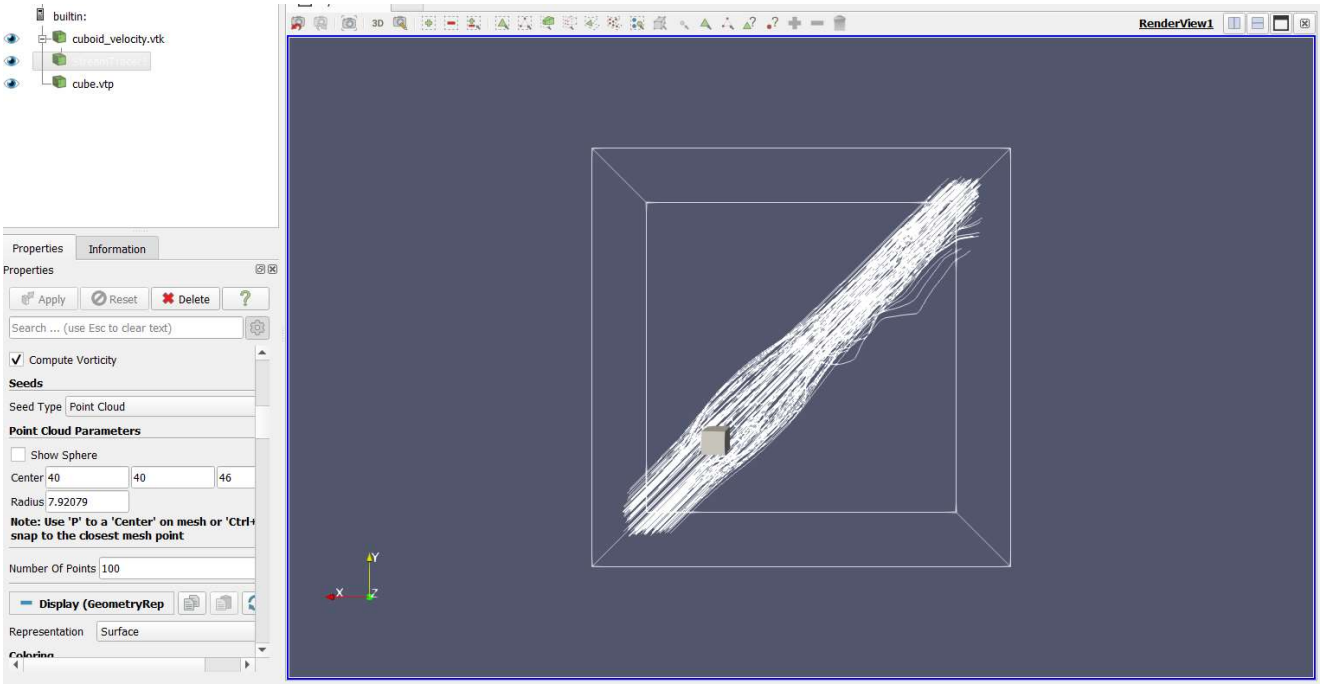
Ex.1: Vector Field Visualization in ParaView

a)

Seeding in front of the obstacle

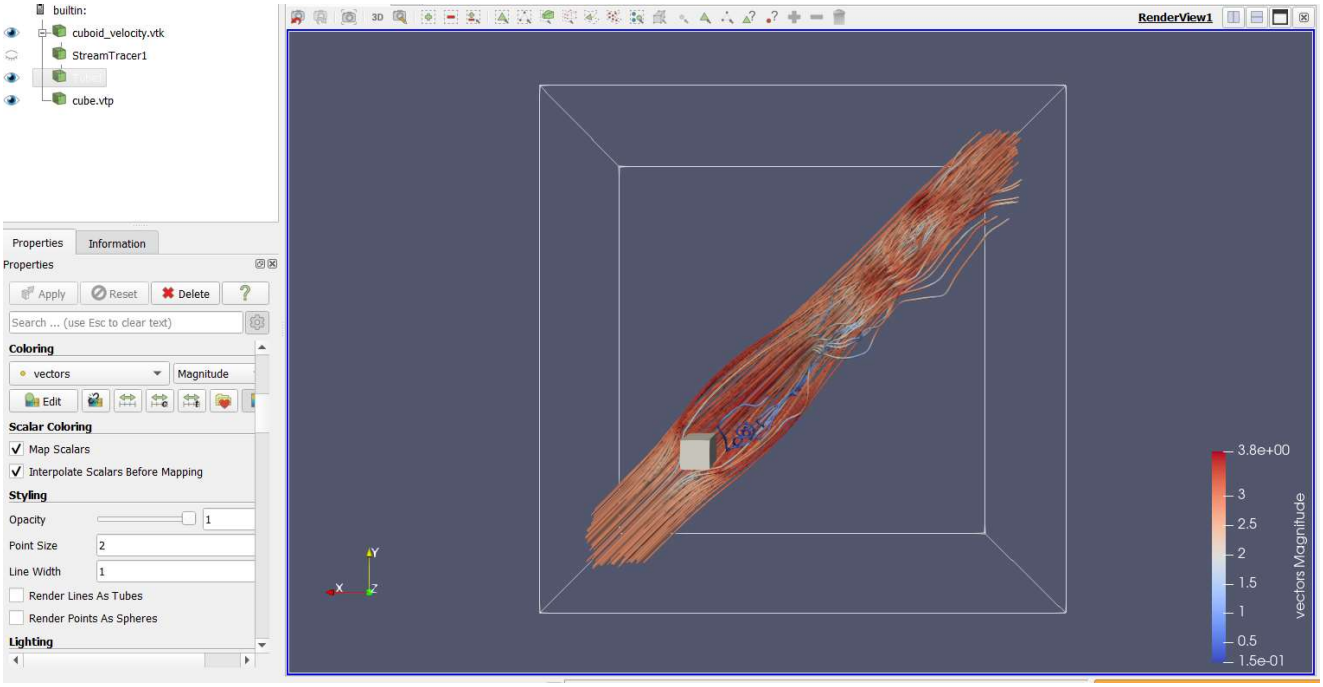


Seeding from behind the obstacle



Regarding the reason to prefer one seeding strategy over the other, it depends on the specific flow and the information you want to visualize. Seeding in front of the obstacle can help you visualize how the flow interacts with the obstacle and how the velocity vectors change as they encounter the cube but cube is not very clearly visible. Seeding behind the obstacle can provide insights into the wake region and how the flow behaves after passing the obstacle as the cube is very clearly visible.

b)



The flow speed regions can be easily observed with the respective color from the color map given in the bottom right corner of the above picture. The region right after the obstacle shows the increase in flow speed in the majority of the streamlines but very few of them show the low flow speed as well indicating the color blue.

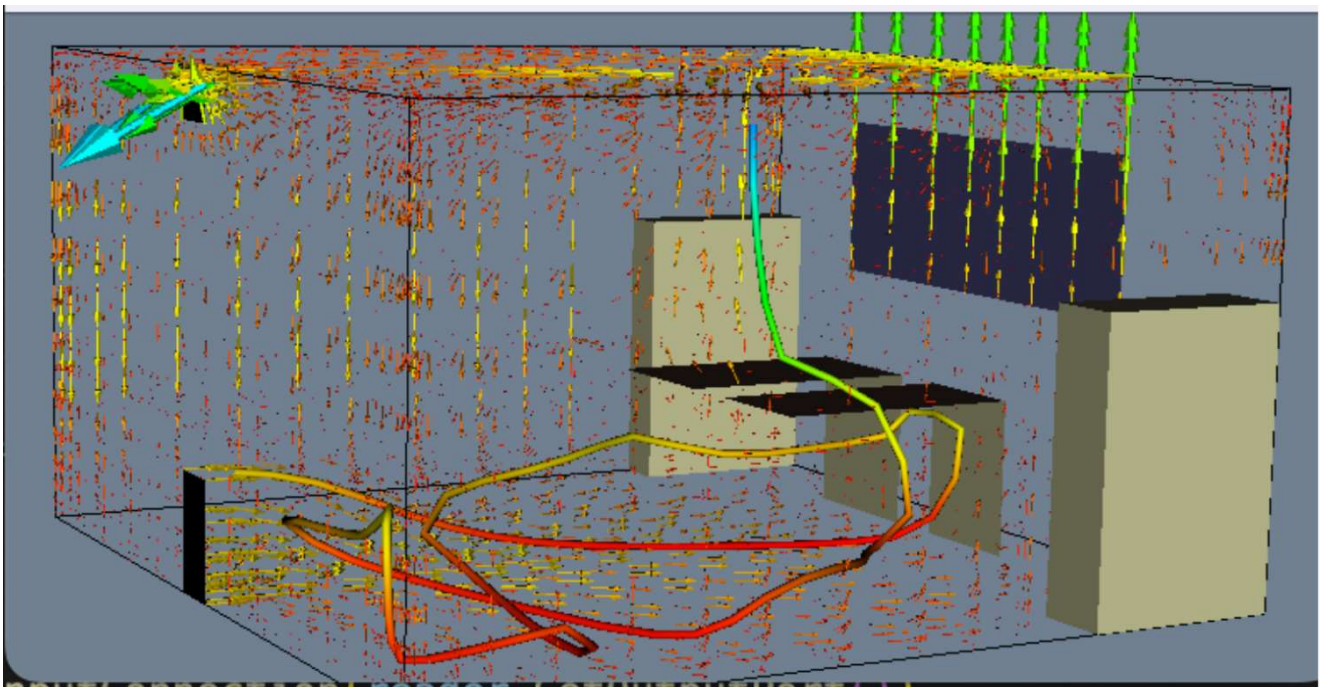
c)

Streamline integration is terminating very few streamlines on maximun streamline length. And it seems like the very few streamlines are terminating because of the maximum number of integration steps are reached. This can be changed by the "Maximum Steps" parameter in the StreamTracer properties. Once the maximum number of steps is reached, the integration is halted.

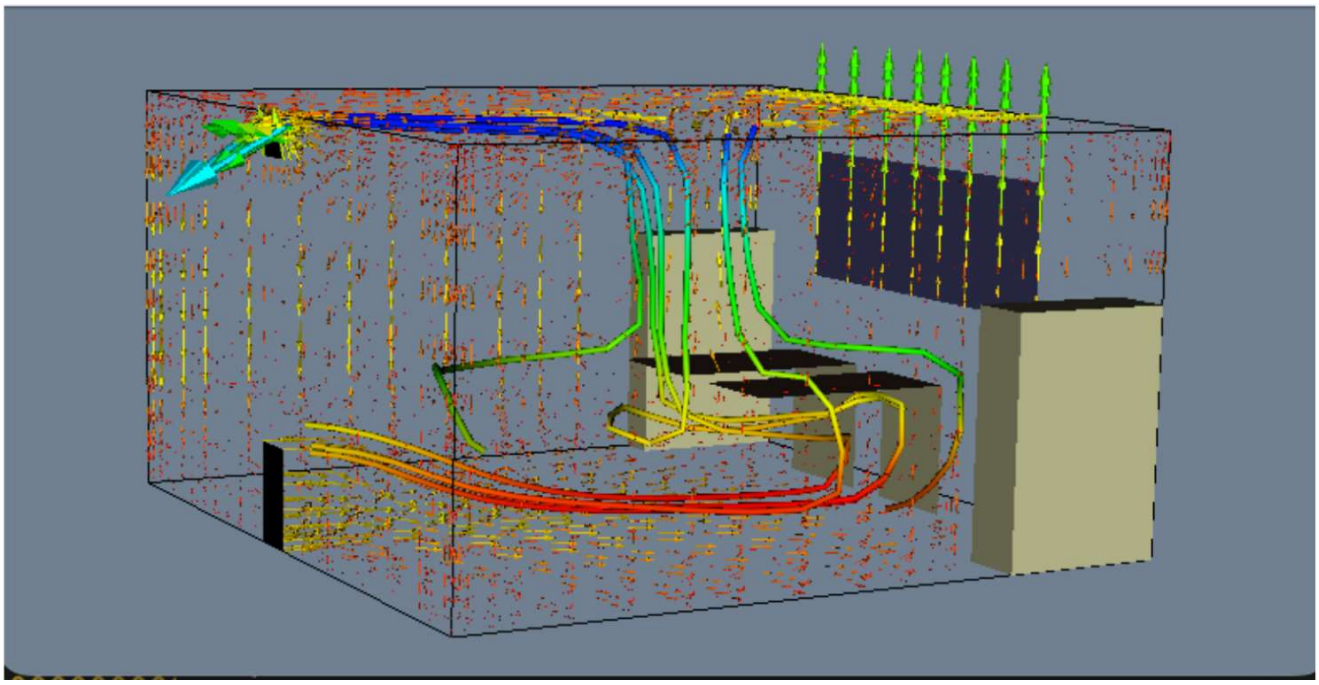
When we substantially increase the maximum streamline length parameter, the reason for termination value drops gradually.

Ex.2: Vector Field Visualization in VTK

a)



b)



c)

By changing InitialIntegrationStep from smaller to larger values (0.1, 0.4, 0.6, 1) incase of smaller values streamlines were smoother and more detailed visualization and especially in regions with rapidly changing flow patterns. In case of large values e.g. “1” resulted in fewer samples along the streamlines. Thus leading to less smoother visualization.

d)

Compared to the fourth order Runge-Kutta scheme, Runge-Kutta45 shows very less effect of InitialIntegrationStep, there was almost no big difference between the first two initial step values 0.1, 0.4 and the last two initial step values. And the reason for that is: The fourth-order Runge-Kutta scheme uses a constant step size throughout the integration process. In this case, the InitialIntegrationStep determines the initial step size for all the streamlines. On the other hand, the embedded scheme Runge-Kutta45 is an adaptive-step integration method. It dynamically adjusts the step size based on the local conditions of the vector field, aiming to maintain a desired level of accuracy. This adaptive behavior allows the scheme to adaptively refine the step size in regions where the flow is complex or rapidly changing and increase the step size in regions with smoother flow behavior.

Ex.3: Numerical Integration

a)

Step 1: Start at the point (-2.3, -3.4) and calculate the slope of the vector field at this point.

Step 2: Use the slope obtained in the previous step to estimate the position of the next point. Since the step size is 1, the estimated point will be (-2.3 + 1 * slope_x, -3.4 + 1 * slope_y).

Step 3: Repeat steps 1 and 2 two more times, using the newly estimated point as the starting point for each iteration.

b)

Step 1: Start at the point (-2.3, -3.4) and calculate the slope of the vector field at this point.

Step 2: Use the slope obtained in the previous step to estimate the position of the midpoint between the starting point and the next point. The estimated midpoint will be (-2.3 + 0.5 * slope_x, -3.4 + 0.5 * slope_y).

Step 3: Calculate the slope of the vector field at the estimated midpoint. Use this new slope to estimate the position of the next point. The estimated point will be (-2.3 + 1 * new_slope_x, -3.4 + 1 * new_slope_y).

c)

Runge-Kutta methods provide more accurate results compared to the Euler method. The Runge-Kutta method involves evaluating the slope of the function at multiple intermediate points, resulting in a better approximation of the true solution. It reduces the truncation error associated with the Euler method and provides a more reliable and accurate numerical integration especially when dealing with complex or stiff systems of equations.

d)

Numerical integration of streak lines is generally more difficult compared to path lines. Streak lines involve tracing the paths of particles released at different times, and the positions of these particles are influenced by the velocity field at each moment. This requires solving a set of coupled differential equations for each particle, which can be computationally intensive and complex.

In contrast, path lines represent the trajectories of individual particles over time, where the particles are released at the same initial time. The computation of path lines only involves integrating the velocity field along the trajectories, which is relatively simpler compared to streak lines.

Ex.5: Streamline Opacity Optimization

a)

Each of the four terms in the energy E serves an important purpose in the line selection algorithm, and removing any of them could lead to undesirable results.

- Sum of the squared difference between the rendered and target image is weighted by p. Setting p to zero would result in renderings with very few or no visible lines, and the rendered image will not be compared to the target image. which would be not informative.
- Sum of squared differences between current and previous opacities are weighted by q. if q is set to zero, there will be no penalty for large changes in opacity from one frame to next, This would result in unstable visualizations.
- Setting r to zero would remove the term that encourages the selection of lines that are important for the current visualization. This could result in the selection of many unimportant lines. Sum of the squared differences between current and initial opacities are weighted by r.
- Setting s to zero, there will be no incentive to reduce the overall opacity of the line segments, This could result in cluttered and occluded visualizations. Sum of the squared opacities weighted by s.

b)

Because p controls the overall opacity level and does not affect the selection of lines. while other three factors, q, r, and s, are responsible for selecting and fading out lines based on their importance, curvature, and coherence with previous frames, respectively. By varying these factors, users can adjust the balance between these different criteria and bring out structures of interest more clearly. However, changing p would only affect the overall opacity level and not the selection of lines, which is why it is sufficient to fix it at 1 during tuning.

c)

The variables that arise in Equations (1)-(4) that are independent from the choice of viewpoint are the opacity values αi for each line segment i. This is because the line selection algorithm is based on the geometry and curvature of the lines, and not on their position or orientation in space.

d)

Frame coherent refers to the consistency of the displayed lines when there is a slight change in the viewpoint or camera position. Achieving frame coherence ensures that there are no sudden changes in the visualization when the viewpoint is moved. frame coherent is achieved by using the previous solution as an initial guess for the next solution, and smoothly blending the current opacities to the latest solution to prevent popping artifacts.

e)

The authors fixed the number of line segments at a relatively modest number. because the computed opacities are blended smoothly along polylines for rendering, thus making it possible to have a small k value without sacrificing the visual quality. A higher number of line segments would not necessarily result in better visual quality and would increase the size of the optimization problem, making it less computationally efficient. The authors observed that k can be rather small, and they used k=8 for all examples.