Problem 1:

1. CPU-based approach on Isosurfaces using marching tetrahedra is implemented with the domain search technique.
2. In CPU based approach, each small cube is considered one by one and then is divided into 6 tetrahedra. In each tetrahedron, there are 4 edges. For a given scalar value, the intersection has been computed and triangles have been generated based on that interception.
3. For the domain search technique, the oct tree has been implemented which will store the minimum and maximum scalar value over the range, and for searching isosurface of a particular scalar value, the whole range can be skipped if the scalar value does not fall between min and max of scalar values of the range.
4. The time complexity to build the tree is O(NlogN), where N = number of cubes and logn is the height of the tree, for searching and time complexity for searching the k cubes that contain isosurface is O(klogN).

Here are three datasets for which code is computed and result is submitted:

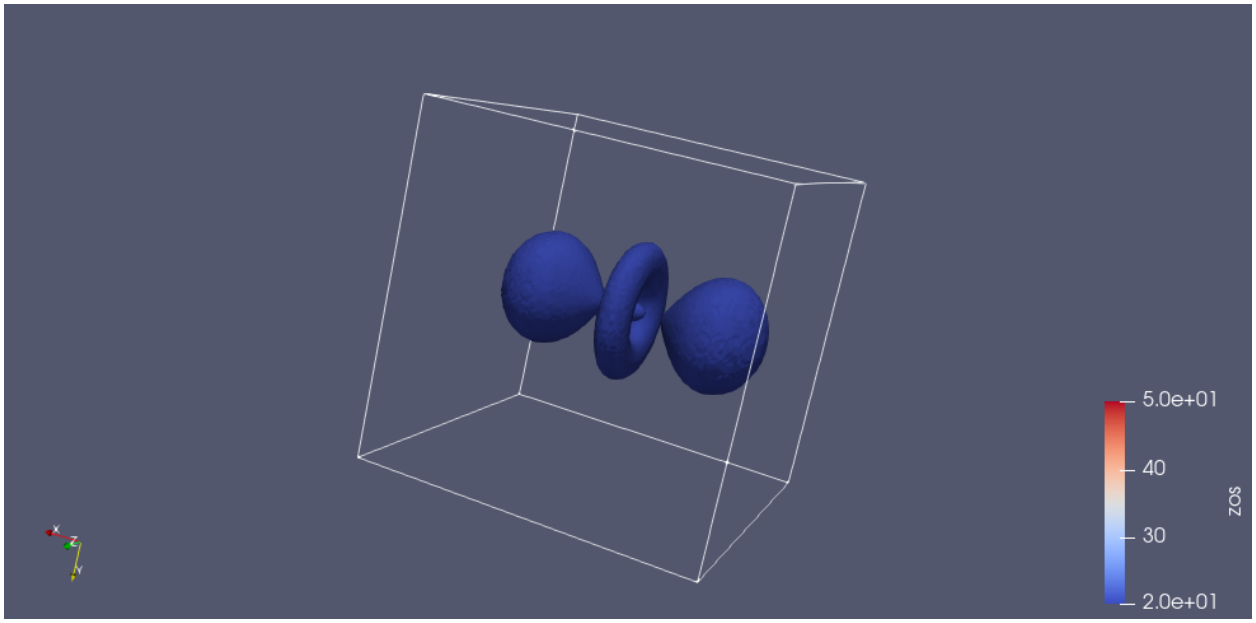
1. Hydrogen atom:

   Size: 128 * 128 * 128

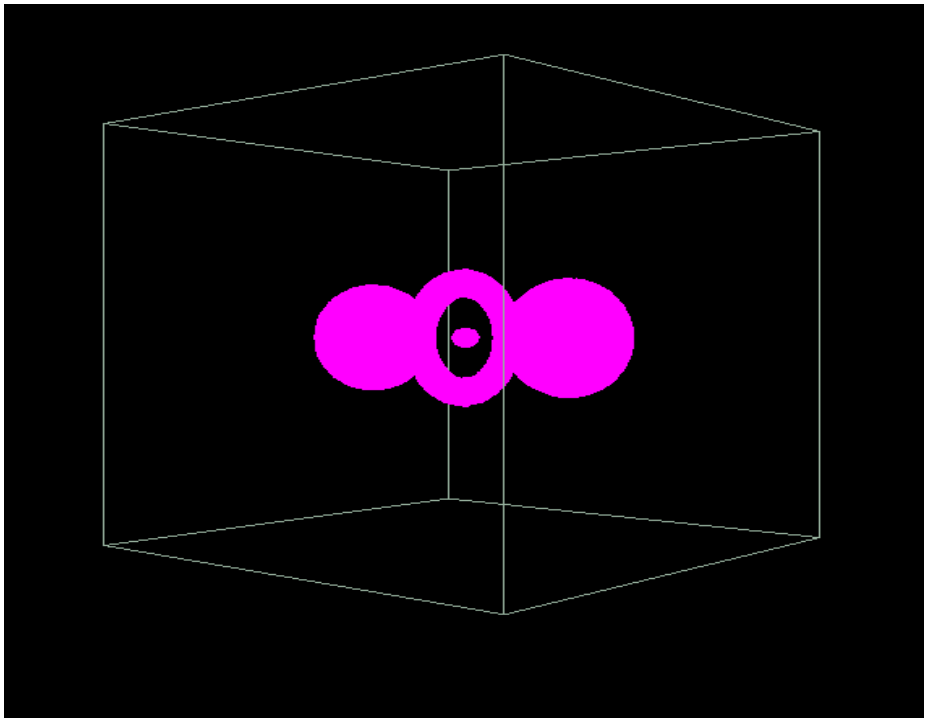
   Total vertices: 2097152

   Isovalue = 20

   Paraview generated output:

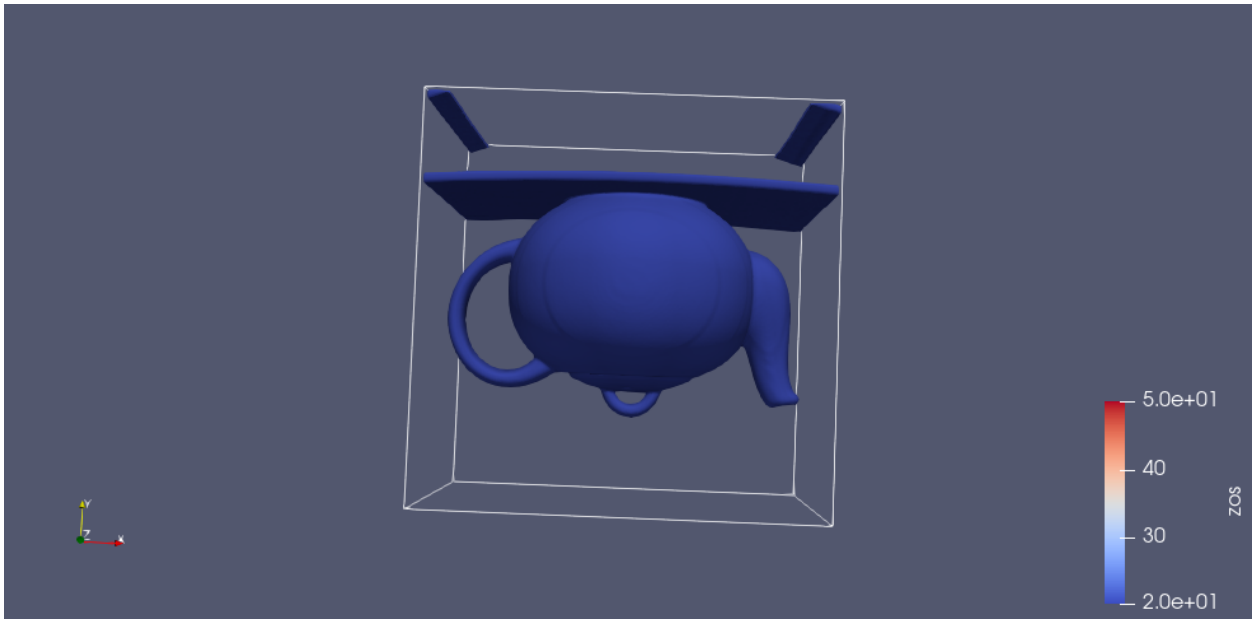


   Code generate output:

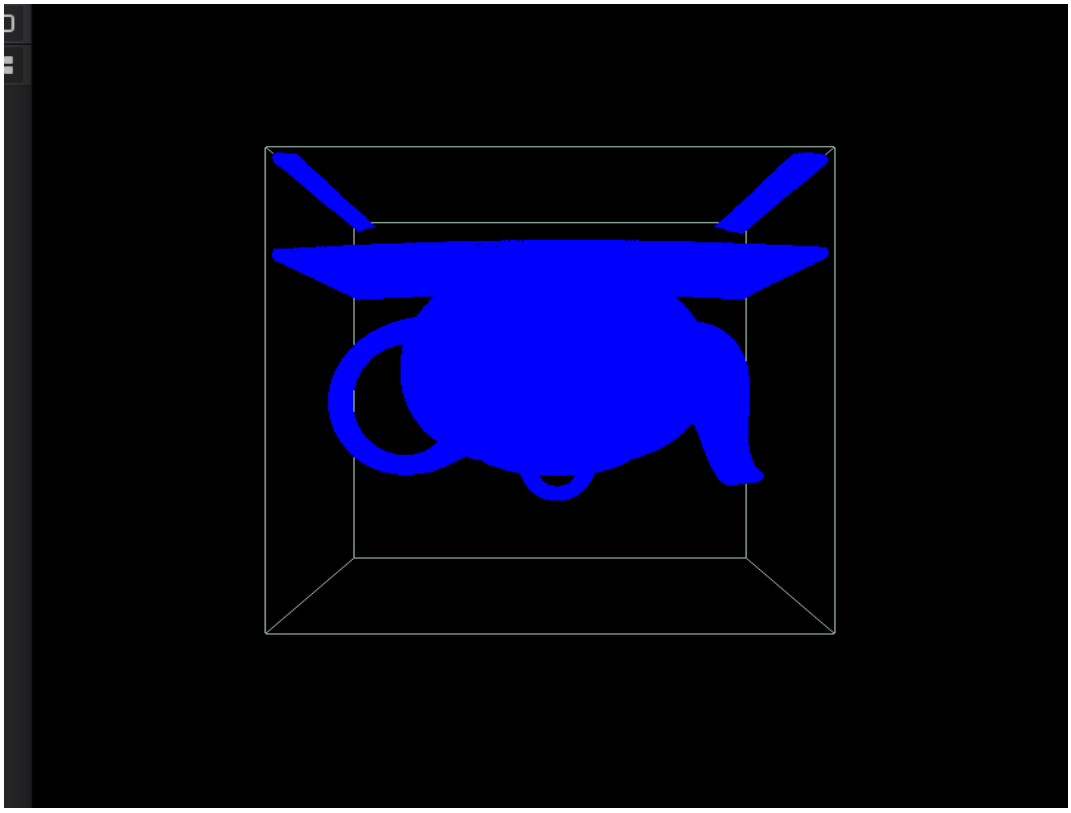


2. Boston teapot:
   Size: 256 * 256 * 178
   Total vertices: 11665408
   Isovalue = 20

   Paraview generated output:



   Code generated output:



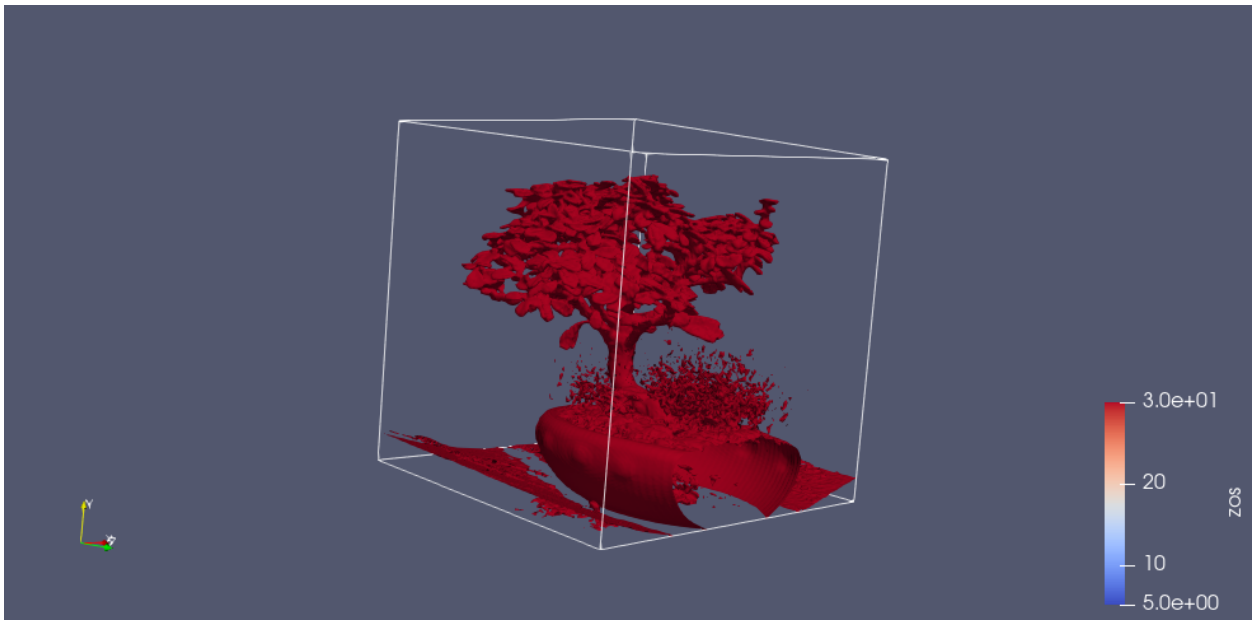
3. Bonsai:

   Size: 256 * 256 * 256

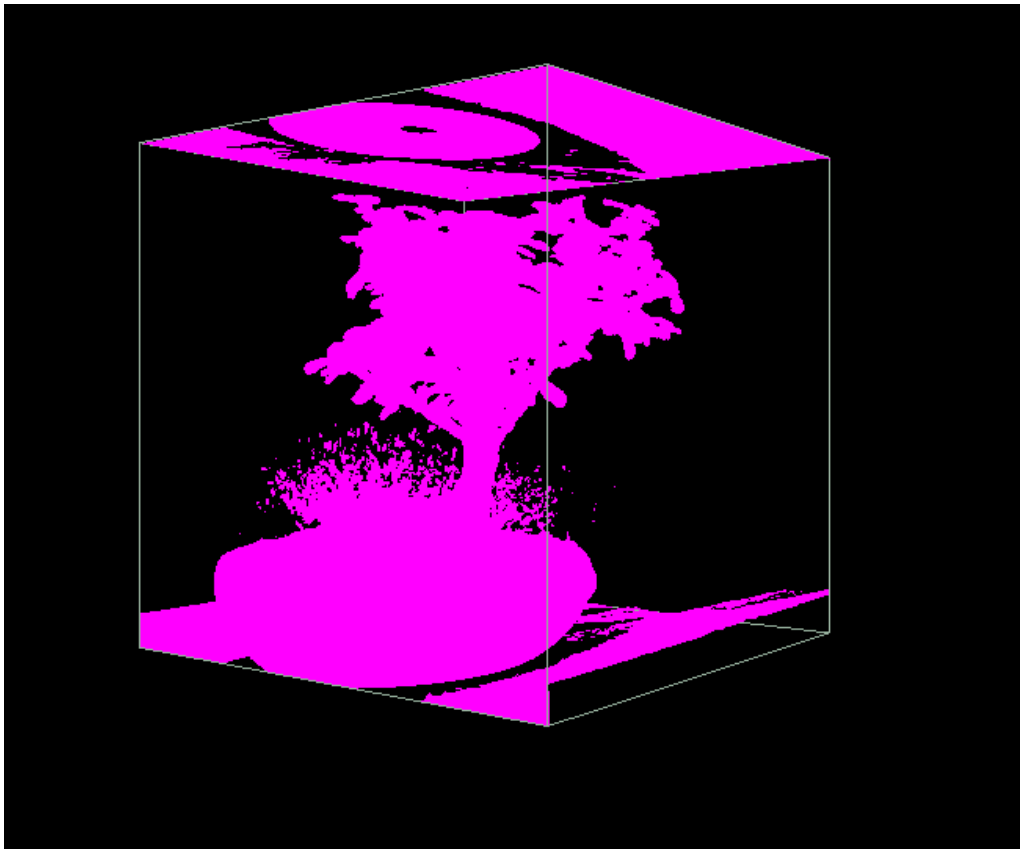
   Total vertices: 16777216

   Isovalue = 20

Paraview generated output:



Code generated output for isoval : 40



- As the size of the dataset increases both the space to store and the time for running the code increases.
- The quality remains the same for all cases, and the FPS was 46.57 for all cases.

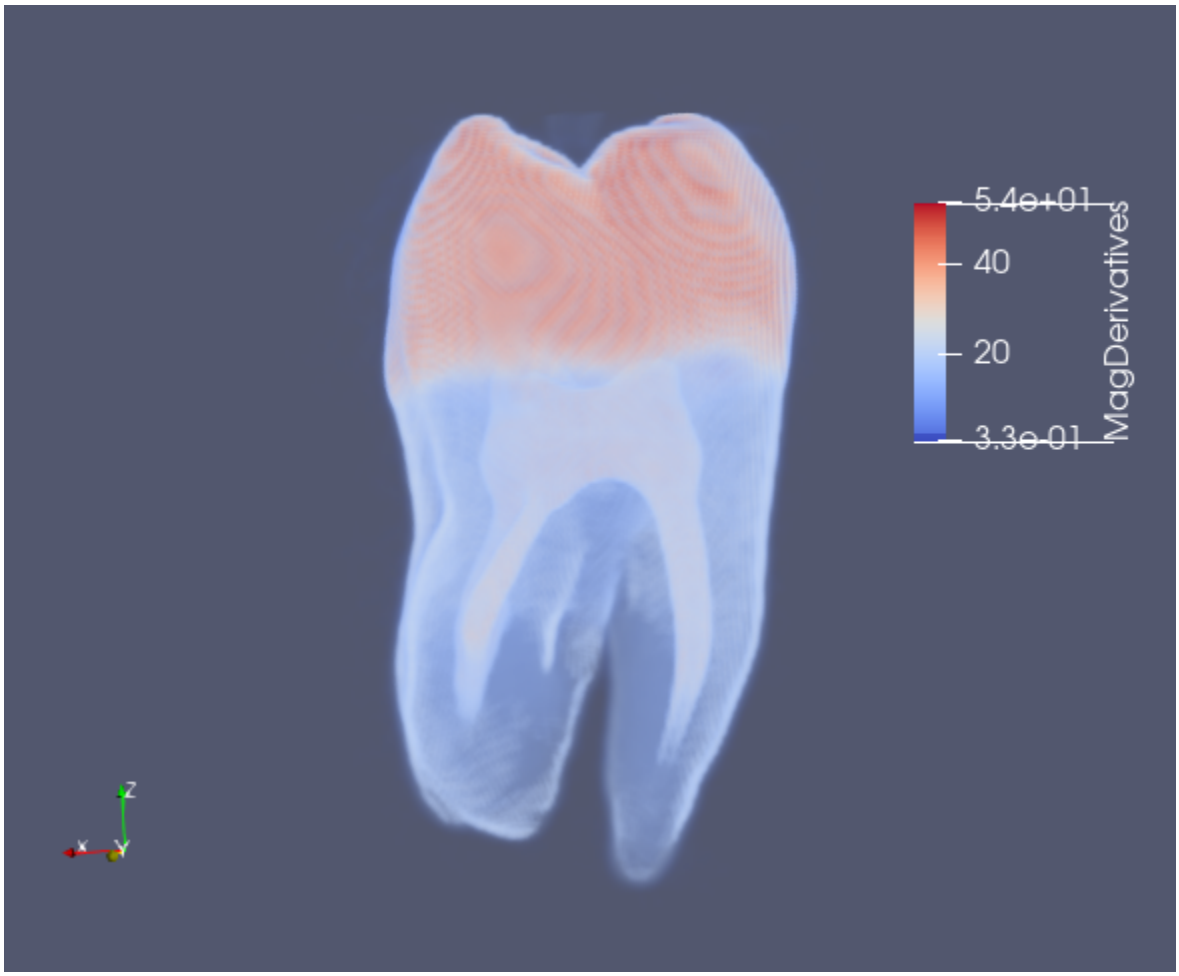
2a.

1. For critical points, the following method has been implemented:
    a. **Maxima:**
        If all neighbor points are having values lesser than the point, it will be local maxima.
    b. **Minima:**
        If all neighbor points are having values greater than the point, it will be local minima.
    c. **Regular Points:**
        If all neighborhood of the point contains some points with a scalar value greater than it and some points with some value lesser than it, and both the greater and lesser neighborhood are connected, then it is a regular point
    d. **Saddle points:**
        The points which are neither maxima nor minima nor regular points are saddle points.
2. Computation:
    a. For each point, all its neighbor points are stored beforehand in an adjacency list.
    b. Then for each point, if it is maxima or minima it is reported and if its direct lesser and greater neighbors are connected it is a regular point, hence these points are skipped.
    c. if its direct lesser and greater neighbors are not connected it may be a saddle point, neighbors and neighbors of neighbors are considered in this case to see if the greater neighborhood is connected or not.
    Even if some points are not direct neighbors of the current evaluating point, if their scalar values are greater than the current point, they can be included to find if the greater neighborhood is connected or not. This is the same for the lesser neighborhood.
    d. If any of the greater or lesser neighborhoods have more than one component, the point is reported as a saddle point.
    e. For checking connectivity, the BFS algorithm is used, if all the points of a greater \ lesser neighborhood are discovered by a BFS from any point of the set, it is said to be connected.

3. Points found:
    Below is the list of points reported as critical points by above mentioned

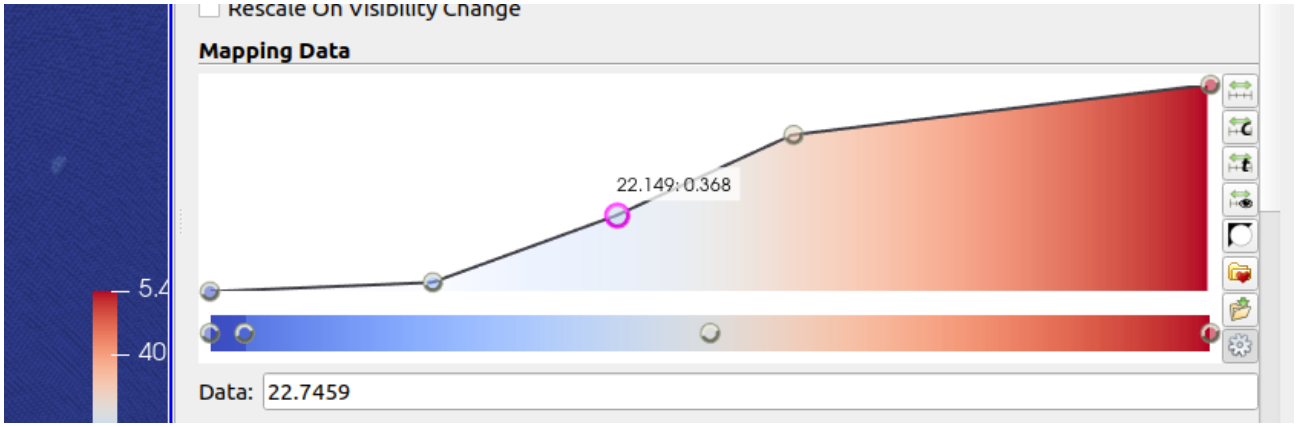
	Considering the border of the dataset	Not considering the border of the dataset

Maxima	[1012142, 1477361]	1012142
Minima	[513, 905517]	905517
Saddle points	171461 247747 257562 394210 653711 655157 886137 895450 952084 1086940 1379746 1503719	-



Transfer function:

The following color map has been used while visualizing the scalar values of the points



Array Name: MagDerivatives

3	2.18077	0.25098	0.329412	0.780392
4	2.18077	0.254902	0.333333	0.784314
5	2.18077	0.345098	0.458824	0.882353
6	12.0833	0.454902	0.592157	0.960784
7	27.1286	0.865003	0.865003	0.865003
8	53.9306	0.705882	0.0156863	0.14902

Opacity transfer function values

	Value	Opacity
1	0.326544	0
2	12.271	0.0441176
3	22.149	0.367647
4	31.5994	0.757353
5	53.9306	1

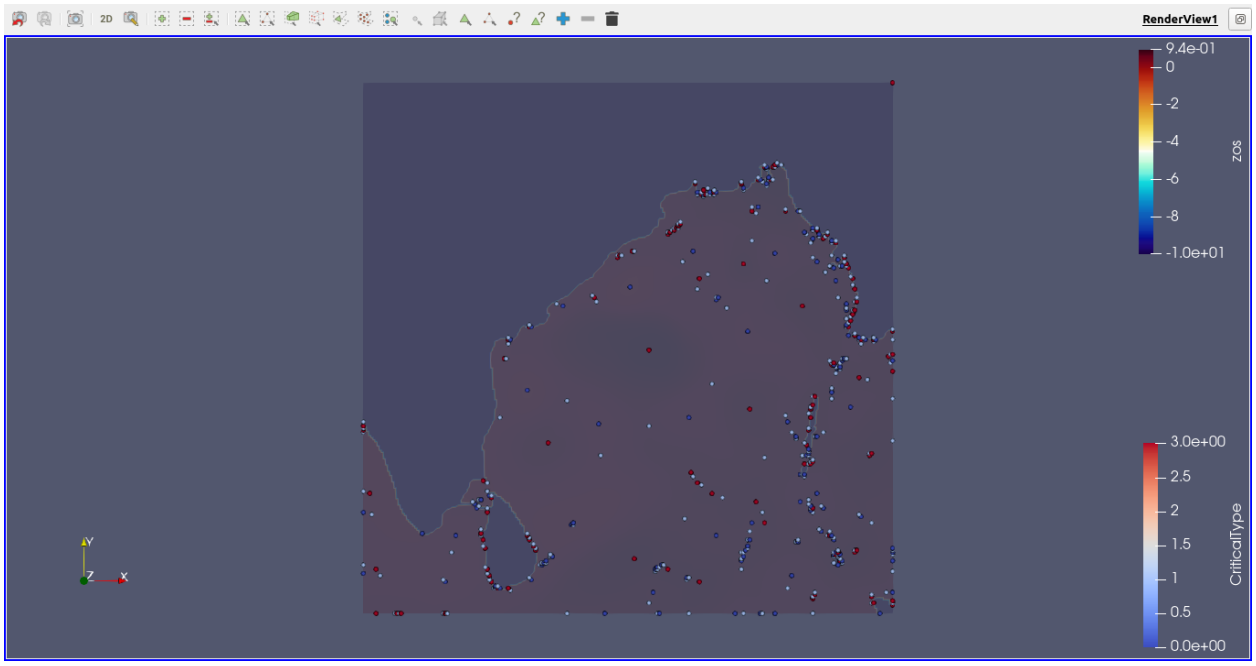
The opacity transfer function is used for direct volume rendering, while the first critical value corresponds to minima and the last critical value corresponds to maxima, and intermediate 3 values are of three saddle points.

2b.

Maxima:

If all neighbor points are having values lesser than the point, it will be local maxima.

For compare: red dots are maxima:

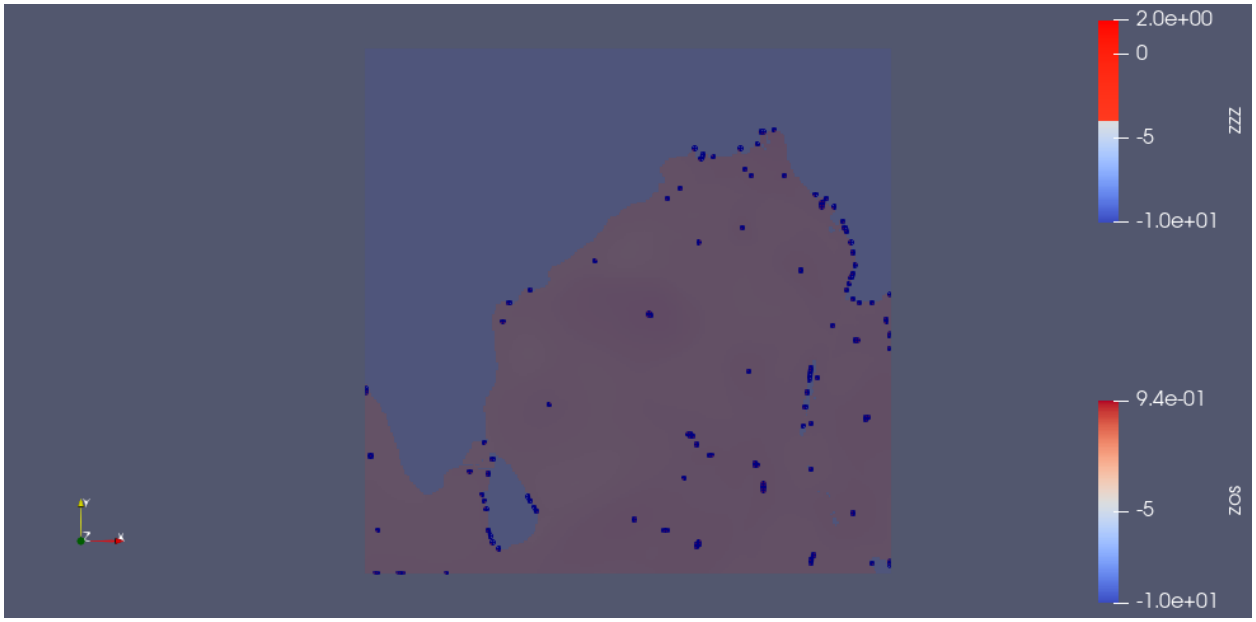


Taking border points into consideration:

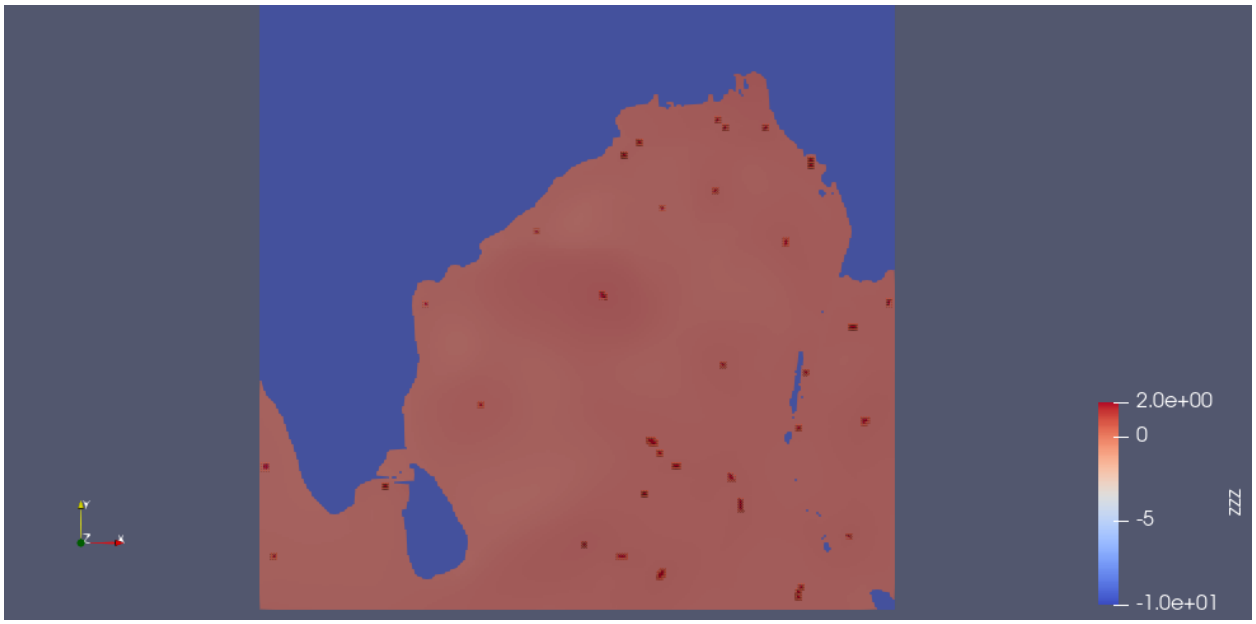
The maxima is greater than its neighbors. The boundary points are not ignored.

Total points reported: 125

Blue dots are the maxima.



Without taking border points:
Points that lie on the boundary of sea or dataset boundary, are ignored.
Total points reported = 60
Red dots are the maxima.



Strictly greater maxima:
If all the maximas are strictly greater than its neighbors, Without taking border points,
Points that lie on the boundary of sea or dataset boundary, are ignored.
Total points reported = 15
Point coordinates:

[92.8333262 5.5833331 0.   ]
[75.4999998 6.7499993 0.   ]
[85.7499957 7.1666658 0.   ]
[94.4999922 7.4166657 0.   ]
[88.2499947 10.1666646 0.   ]
[82.3333304 11.7499973 0.   ]
[90.3333272 13.0833301 0.   ]
[84.166663 17.499995 0.   ]
[88.333328 18.2499947 0.   ]
[90.0833273 18.8333278 0.   ]
[93.2499927 19.8333274 0.   ]
[87.0833285 19.999994   0.   ]
[87.5833283 20.4166605 0.   ]
[90.4166605 20.9166603 0.   ]
[91.7499933 20.9166603 0.   ]

White dots are the maxima.