**Problem 1**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

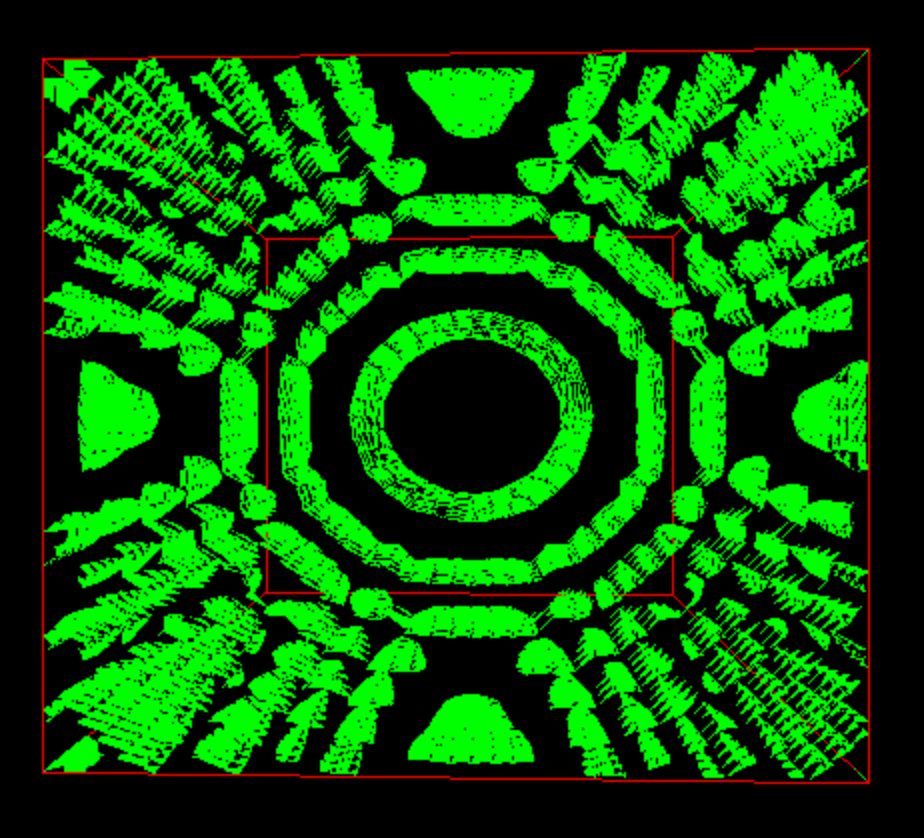
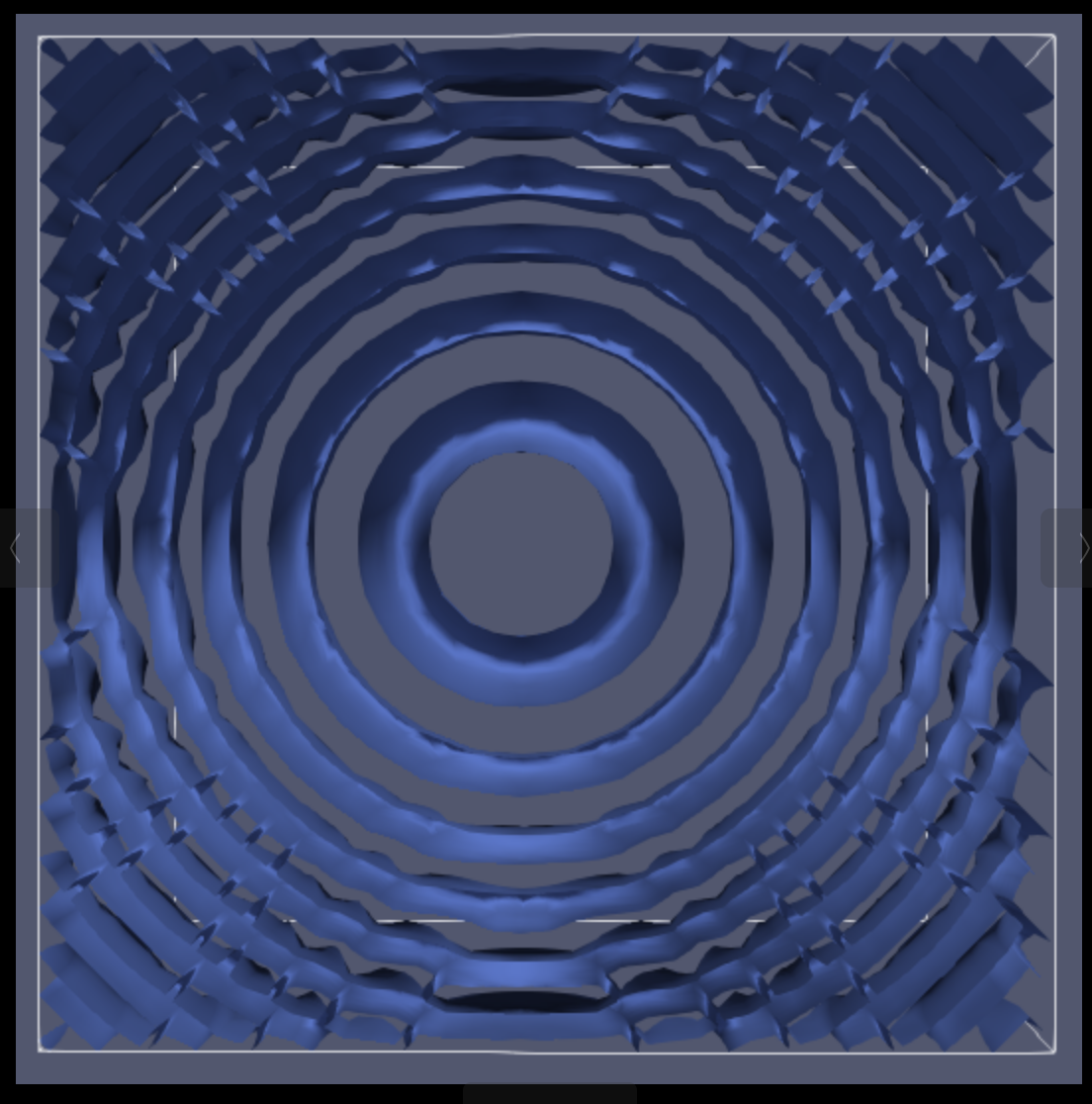
**DataSet:** Marschner-Lobb

**Dimensions:** 41x41x41

**Size:** 67.3 KB

| Scalar Value | No. of vertices | Time to create vertex buffer (Marching Tetrahedra) | Time to create vertex buffer (Domain Searching) |
| --- | --- | --- | --- |
| 40 | 337365 | 10.795 ms | 8.094 ms |
| 80 | 203847 | 7.287 ms | 3.6 ms |
| 120 | 180927 | 5.576 ms | 3.189 ms |
| 160 | 179842 | 6.893 ms | 2.671 ms |
| 200 | 175580 | 6.491 ms | 3.766 ms |
| 240 | 172794 | 6.91 ms | 2.536 ms |
| 255 | 408 | 4.128 ms | 0.05 ms |

**Preprocessing time for Domain Searching (building tree):** 17.75 ms

** **

1. Domain Searching / Marching tetrahedra 2) Paraview

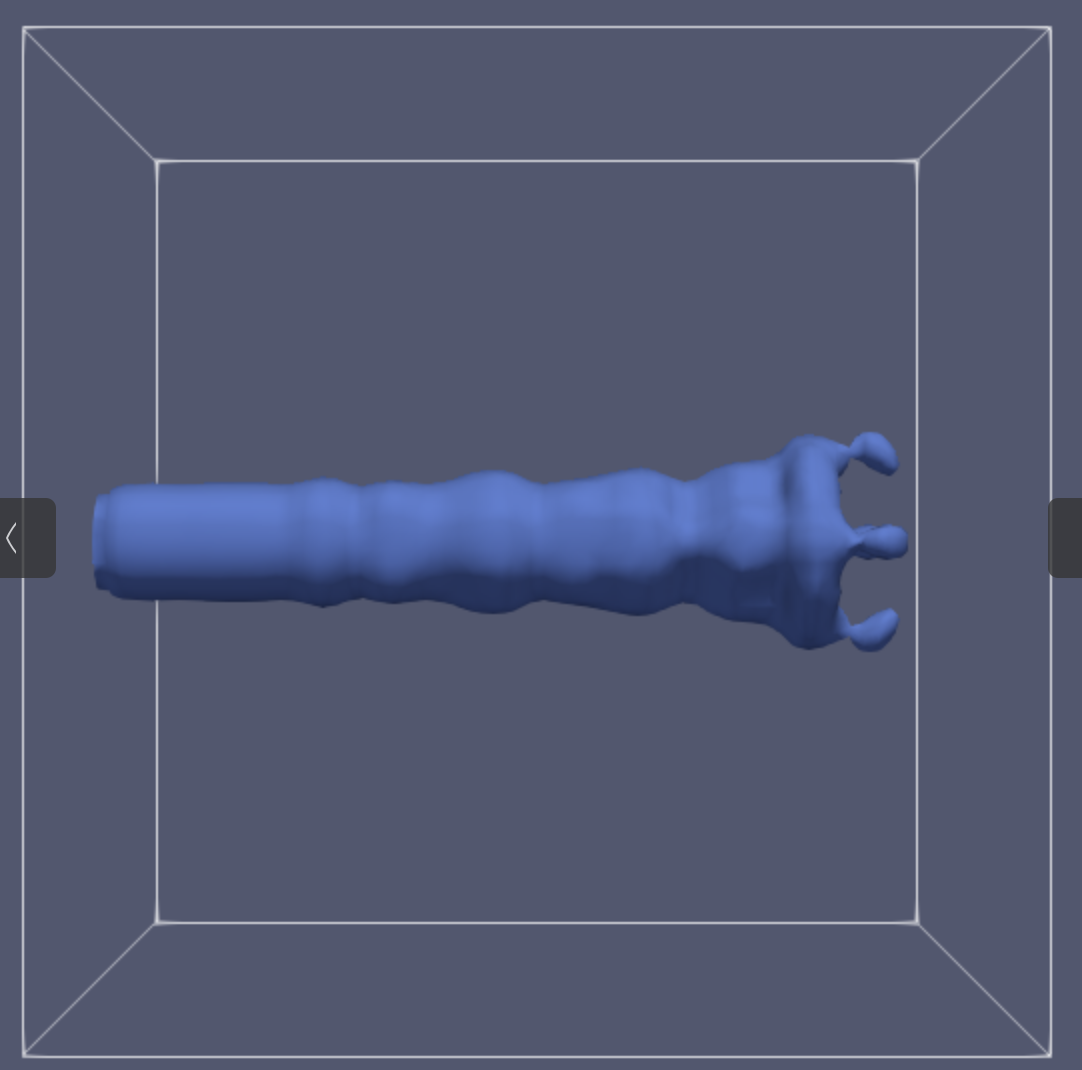
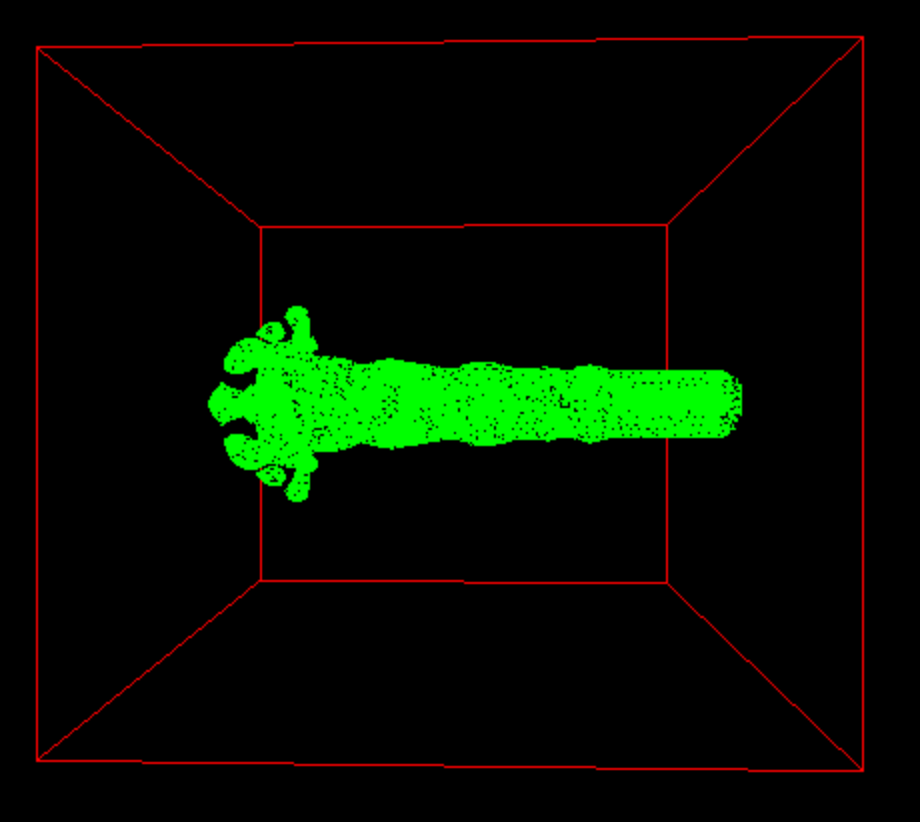
**DataSet:** Fuel

**Dimensions:** 64x64x64

**Size:** 256.0 KB

| Scalar Value | No. of vertices | Time to create vertex buffer (Marching Tetrahedra) | Time to create vertex buffer (Domain Searching) |
| --- | --- | --- | --- |
| 40 | 51876 | 23.859 ms | 1.89 ms |
| 80 | 38928 | 18.739 ms | 0.904 ms |
| 120 | 24648 | 17.503 ms | 0.601 ms |
| 160 | 17340 | 16.387 ms | 0.428 ms |
| 200 | 8520 | 16.826 ms | 0.302 ms |
| 240 | 2928 | 18.123 ms | 0.131 ms |
| 255 | 1152 | 15.93 ms | 0.054 ms |

**Preprocessing time for Domain Searching (building tree):** 97.581 ms



1. Domain Searching / Marching tetrahedra 2) Paraview

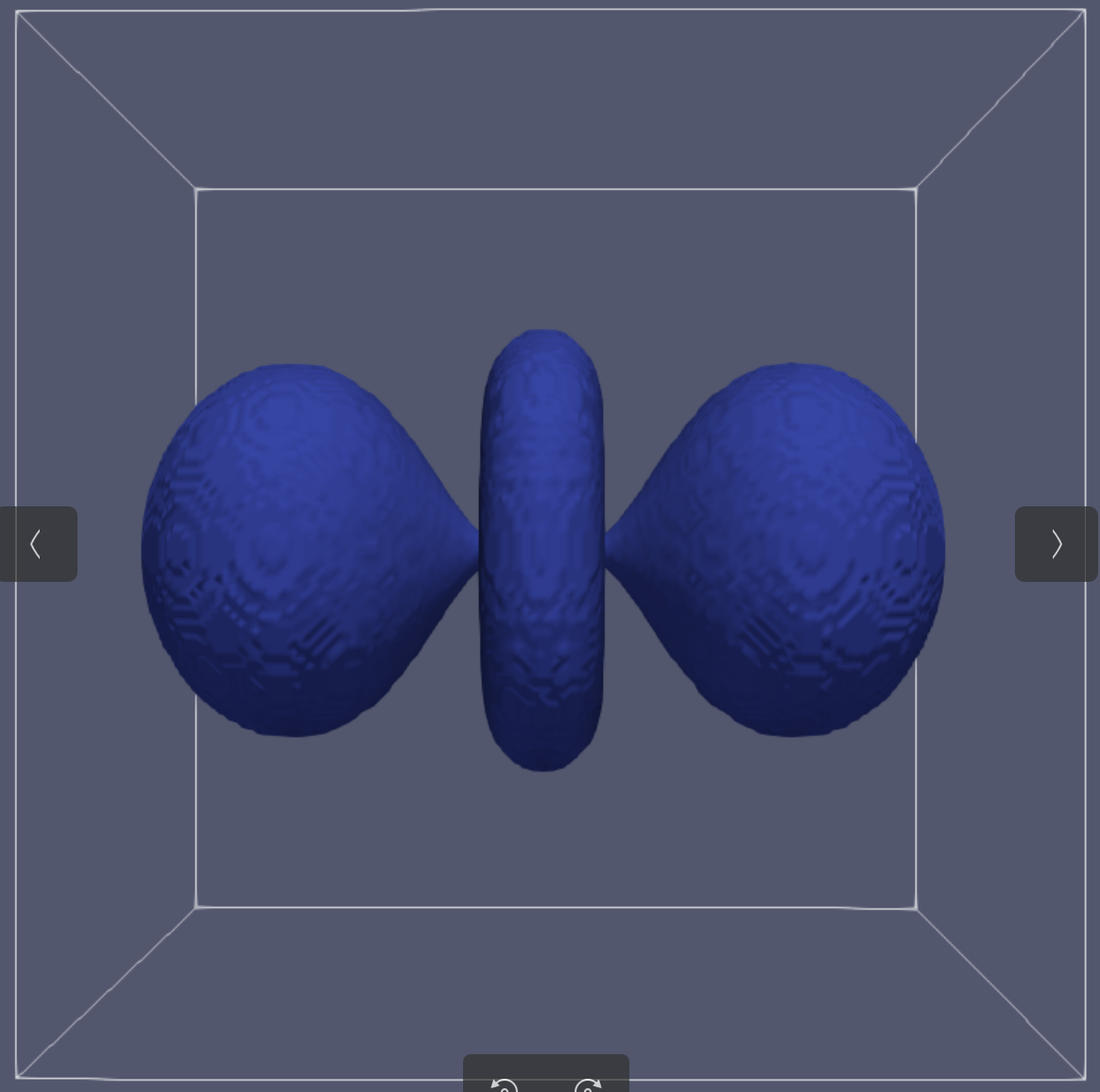
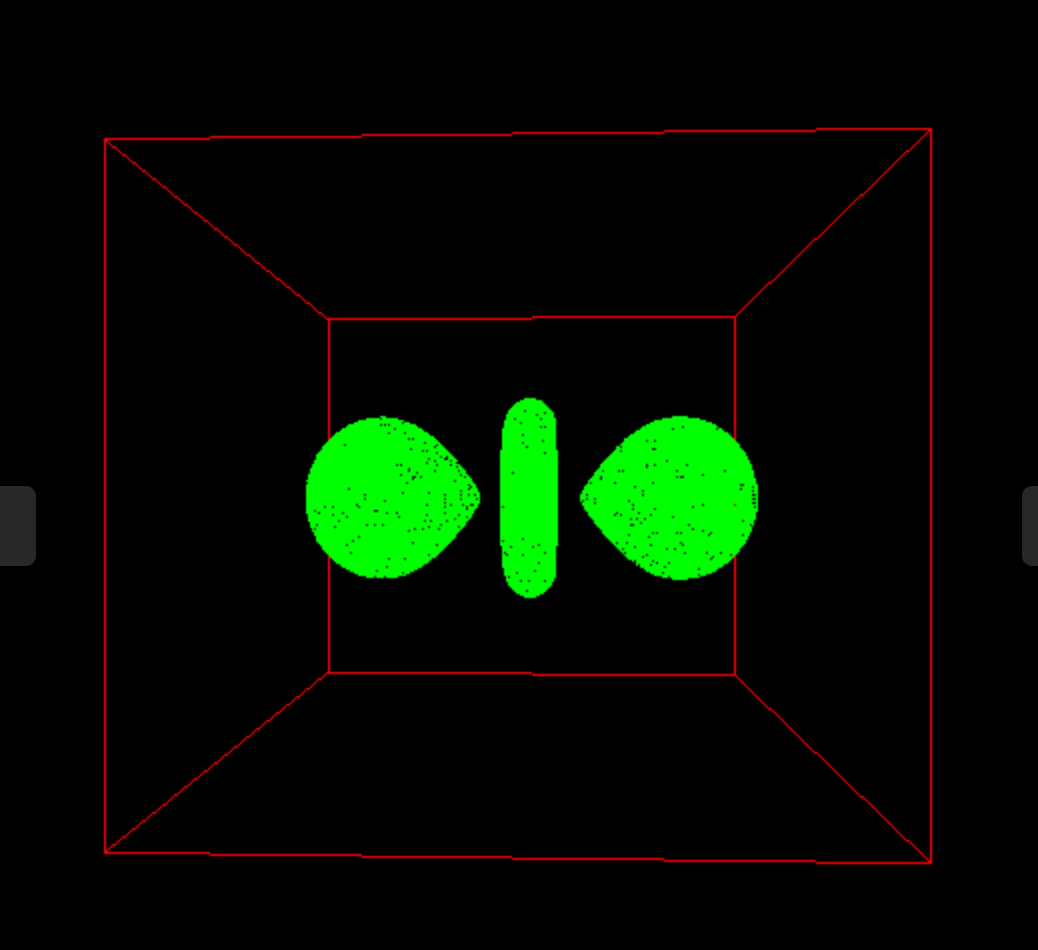
**DataSet:** Hydrogen

**Dimensions:** 128x128x128

**Size:** 2.0MB

| Scalar Value | No. of vertices | Time to create vertex buffer (Marching Tetrahedra) | Time to create vertex buffer (Domain Searching) |
| --- | --- | --- | --- |
| 40 | 135960 | 121.277 ms | 4.109 ms |
| 80 | 1440 | 126.525 ms | 0.143 ms |
| 120 | 432 | 125.877 ms | 0.093 ms |
| 160 | 72 | 124.707 ms | 0.022 ms |
| 200 | 72 | 124.438 ms | 0.047 ms |
| 240 | 72 | 124.822 ms | 0.03 ms |
| 255 | 0 | 130.233 ms | 0.0 ms |

**Preprocessing time for Domain Searching (building tree):** 496.576 ms



1. Domain Searching / Marching tetrahedra

**Observations:**

* As we increase the scalar value, the number of vertices goes down in the output, and it is identical for both Marching Tetrahedra as well as Domain Searching.
* However, the time taken by marching tetrahedra to recalculate the vertices after changing scalar value is much higher than Domain Searching as we have to iterate over all the cubes again and again, in contrast to searching the cubes for plausible candidates for Domain Search.

**Problem 2a**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Local Maximas | Total : 2

Local Minimas | Total : 2

Saddle Points | Total : 6

-------------------------------------

**Local Maximas:**

45.019997893

53.930614285

-------------------------------------

**Local Minimas:**

0.32654431492

0.85126404677

-------------------------------------

**Saddle Points:**

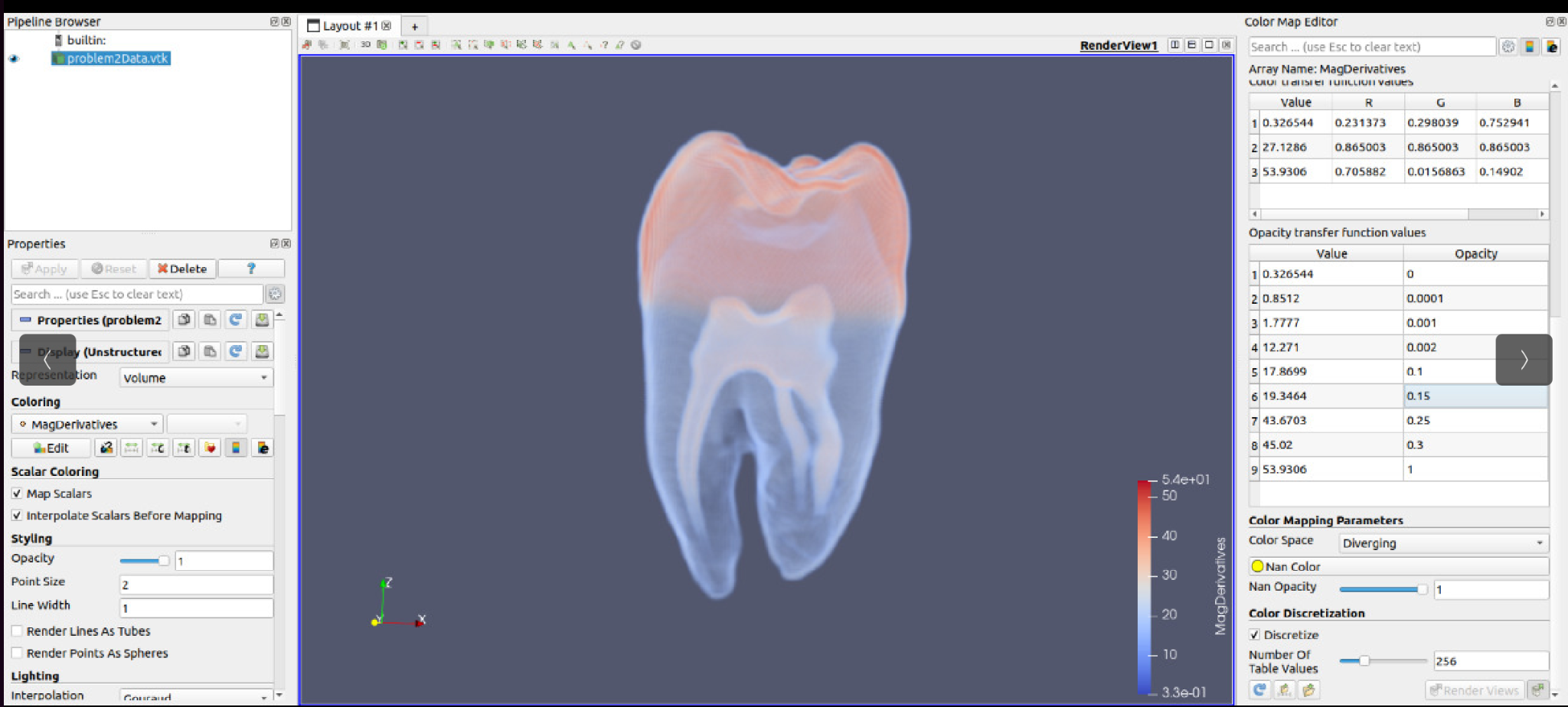
12.271048299

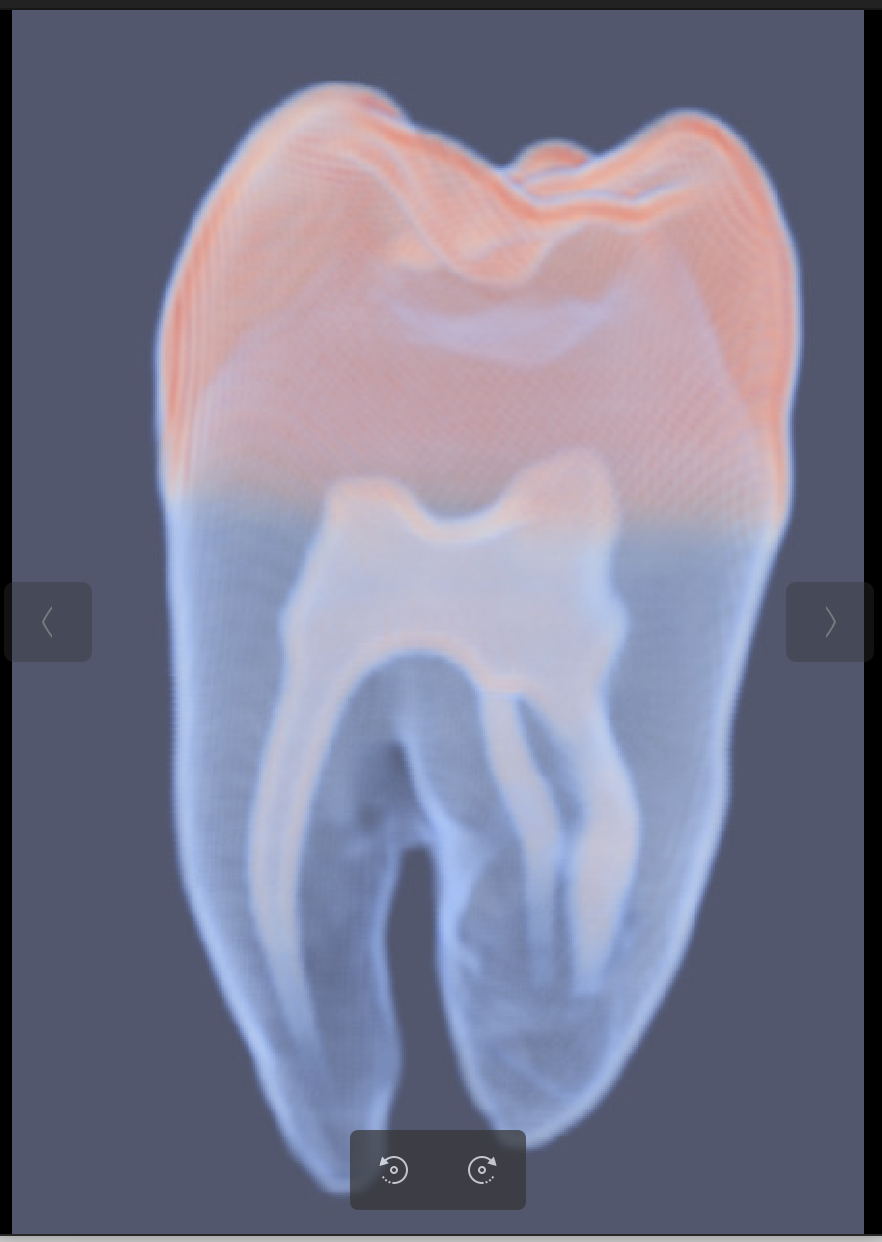
1.7770817416

17.869913783

19.346409047

43.670351333

****

****

**Procedure:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

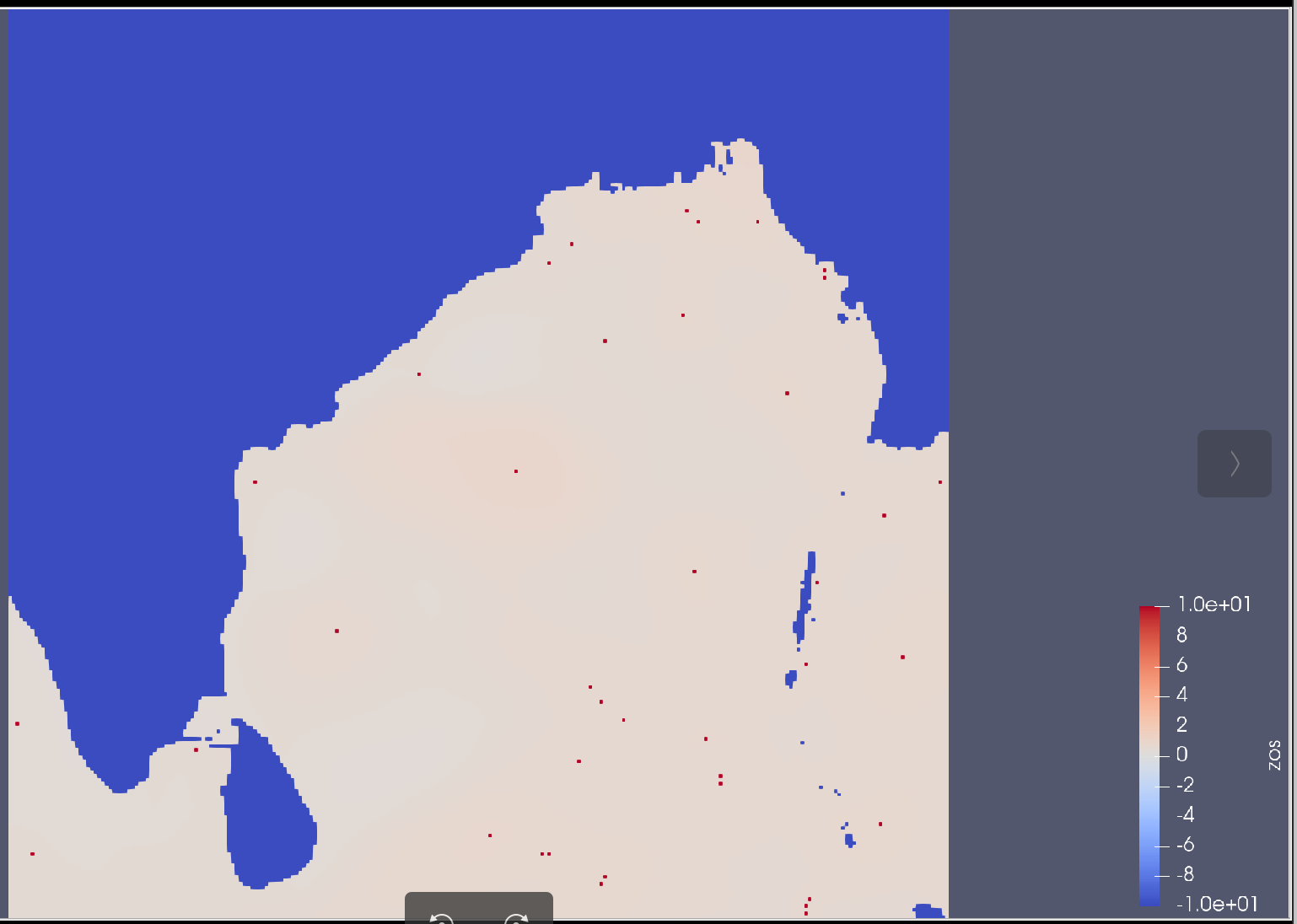
**Calculator of maxima/minima:** For every point, we are checking whether its ttkOffsetScalarField is greater than that of all its neighbors for maxima. Similarly for minima, we check whether the ttkOffsetScalarField is less than all its vertices.

**Calculation of saddle points:**

* First of all I separated the neighbours of the given vertex into two lists, the one which had their values all greater than that of the current vertex (high list), and the other which had all its values less than that of the current vertex (low list).
* Second, for every vertex I am checking whether the set of vertices in the high list are connected to each other via immediate vertices (which may not be in the high list) such that all the intermediate vertices have their values higher than that of the current vertex.
* We do a similar connectivity check for the low vertices.
* If we find that either the vertices in low list or high list are disconnected as per above criterion, then we declare them as saddle point.
* To perform connectivity check, I used bounded depth BFS starting from multiple sources (all the vertices of the high list) and found the vertices reachable via each of the sources. Then using the union find algorithm, I checked whether the set of reachable vertices from all these sources forms a single set or not. If it does, we declare them as connected.

**Problem 2b**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

****

**Procedure:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* First we read the VTK file in a grid.
* To find the local maxima, we considered the neighbours of each vertex. We consider it a candidate for local maxima only if the point and none of its neighbours have values equal to the placeholder.
* The values of these local maximas were set to a high value.
* Then using paraview, the visualization was performed.