2 Week 2

The main tasks for the week were to generate the driver files, mark the initial conditions and finally mark neighboring nodes of the initial alive points. Throughout the whole work, a structured square mesh has been used for simplicity but they can be changed with a simple change in the initial conditions. The code is developed in a way to tackle any unstructured domains. The only change that needs to be done is to change the way in the initial conditions for the neighboring points are set.

2.1 Initial stages of the solver

A solver class named `EikonalSolver` has been initiated. It is currently capable of taking the inputs given and plotting the current state of the nodes i.e. in a graphical manner it will represent how many nodes are Alive($\mathcal{A}$), Far Away($\mathcal{F}$) and in the Narrow Band($\mathcal{N}$) region.

The variables used to define the solver are as follows

- `Mesh2D *mesh`: This stores the address of the mesh which is involved in the solver. The mesh itself stores the co-ordinates, nodes, elements, etc.
- `function<float(float, float)> F, v1, v2`: These functions store the wave speed and medium velocity fields i.e. $F(x, y)$, $v1(x, y)$ & $v2(x, y)$ respectively. Where the medium velocity is given by $v(x, y) = v1(x, y)\hat{i} + v2(x, y)\hat{j}$.
- `priority_queue<Node*, vector<Node*>, Compare> narrowBandHeap`: This is a priority queue which would enable us to implement the heap data structure, which is provided as a part of the Standard Library in C++. The class `Compare` has been designed in order to sort the nodes with respect to their T i.e. their wave arrival time. At each iteration, the head would contain the node in the Narrow Band region with the minimum arrival time.

2.2 Domain of Interest

The domain(Ω) is an (un)structured grid as shown in Fig. 4.

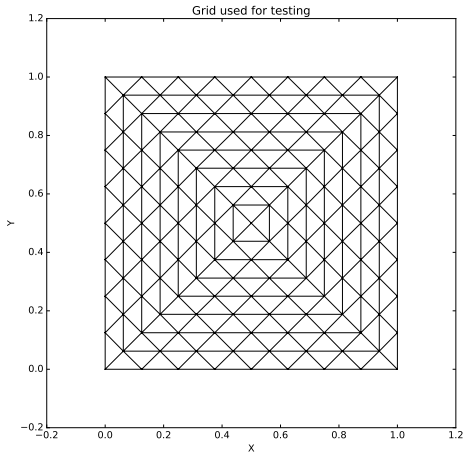

Figure 4: The domain(Ω) on which the code is tested

2.3 Initialization

The whole process of initialization can be broken down into the following steps:

1. The initial conditions are chosen such that the line $x = 0$ would be the initial set of Alive points & the points on the initial wavefront. The code would initially consider all the nodes to be far-away points and the grid would be of the state as shown in Fig. 5.

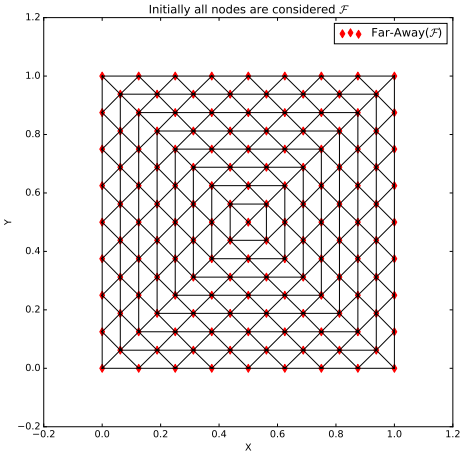

Figure 5: All the nodes are initially marked as $\mathcal{F}$

2. Then, all the points satisfying the condition $x = 0$ would be considered to be Alive and hence, would be removed from the set $\mathcal{N}$, and added to the set $\mathcal{A}$. The state of all the nodes in the grid resembles the state in Fig. 6

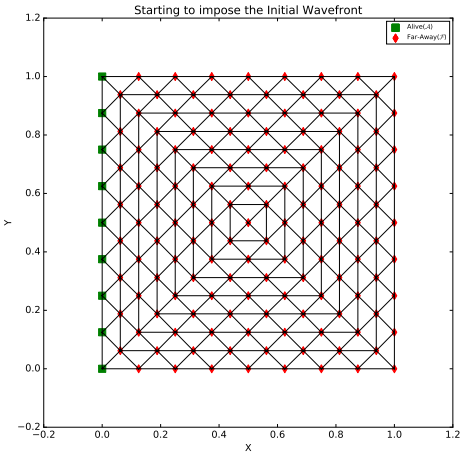

Figure 6: The initial condition and the nodes of initial wavefront are pushed in the set $\mathcal{A}$

3. Finally all the neighboring nodes of the Alive nodes and belonging to the set $\mathcal{N}$, and the state would replicate the state as shown in Fig. 7.

2.4 Conclusion

The initialization has been complete, and the next task would be to trying to implement the Fast Marching Algorithm and the initialized solution. The code used in [?] must be now implemented on unstructured grids.

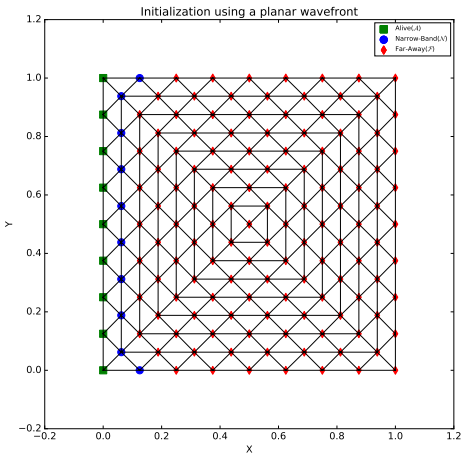

Figure 7: Initialization has been complete and the set $\mathcal{N}$ has been populated successfully.