Math 228B
Homework 7
Ahmad Zareei

Introduction

In this problem set, we will talk about fast marching method and solving Eikonal equation. Basically, we have a cost function which shows the accumulated cost of getting to a point, $c(x, y)$, and the goal is to find cost for reaching to each point of the domain. We will solve the Eikonal equation in the following form in order to find the cost of reaching to each point

$$|\nabla T| = c(x, y), \qquad T = 0 \text{ on } \Gamma \tag{1}$$

where $c(x, y)$ is the cost function. To better understand this formula consider 1-D region whose boundary is moving with speed $1/c(x, y)$. We know that differential displacement dx is equal to speed times the time elapsed, i.e.

$$dx = \frac{1}{c\,(x, y)} dT \quad \Rightarrow \quad c\,(x, y) = \frac{dT}{dx} \tag{2}$$

where T is the cost (i.e. time) of reaching to a point, and $c\,(x, y)$ is one over the speed of propagation. If we move faster (less cost), we could reach a point faster (less time) and vice versa.

We will start with $T = 0$ at some specific point, and then find the cost of reaching to all other points in the domain. Having this cost function over the whole domain, we can find the shortest pass from any point to the starting point that we picked at first. We just need to follow the direction of gradient at each point in order to find the shortest path.

Problem Definition

In this problem set we will solve the equation

$$|\nabla T| = c(x, y), \qquad T = 0 \text{ on } x_p \tag{3}$$

where $c(x, y)$ is the cost function, $T(x, y)$ is cost of reaching to a point and x_p is the starting point. For simplicity, we solve this equation on $[0, 1] \times [0, 1]$ domain and we will set x_p to be at $(0, 0)$. We will first solve for $c(x, y) = 1$ and then we will change this $c(x, y)$ to see the influence of this cost function on the solution. We will plot the contours of T to show the curves of same cost $T(x, y)$. At the end, we will find the shortest path from point $(0.75, 0.75)$ to the origin by moving along the direction of gradient of $T(x, y)$.

Implementation

The goal is to find the cost value at each node. Basically, we first set all values of nodes to some large number. Then we set the origin to zero and mark it as visited. Then we update its neighbours values and and set them as unvisited. From now on, we pick the the unvisited node with smallest cost value, update its neighbours and set the node as visited and mark the neghbours as unvisited. We will go forward untill no node is remained. The only thing remained is how we will update the cost value of neighbours. The way that we should update the neighbours is the following

We first compute the cost value from our point to the neighbour, if it is less than the cost of the neighbour node, we set the value of the neighbour to the cost that we found. If the cost is same as the value of the neighbour (numerically with some error), then we use all visited nodes with connection to this neighbour node to compute its cost. The case, that cost found is larger than value of neighbour node, we do nothing. Way of finding cost is one of the following

1. Suppose we are updating a node's neighbour say $T_{i,j}$, and the visited node's cost is T_v, then
$$\frac{T_{ij} - T_v}{dx_i} = c_{ij} \tag{4}$$
where dx_i could be either dx or dy.

2. Suppose the value is coming from two visited neighbours with costs T_{v1} and T_{v2}, we solve for T_{ij} in
$$\left(\frac{T_{ij} - T_{v1}}{dx_1}\right)^2 + \left(\frac{T_{ij} - T_{v2}}{dx_2}\right)^2 = c_{ij}^2 \tag{5}$$

This will go forward, untill no node is left and all nodes are marked as visited. This method will be of order $O(n \log n)$, if we use heap sort for finding smallest unvisited node. However, In this home work, we are not using the heap sorting method and we are doing the computation of order $O(n^2)$. We will start from the bottom row and sweep forward, until values of nodes reach their final values.

For finding the shortest path, we will find the ∇T at the closest node of our current posintion of path. Then we will use this value of derivative to update the next position.

Problem

1. Part One:Program up the fast marching method. On a box, $[0,1] \times [0,1]$, with 100 mesh points in each direction, find the cheapest cost from $(0,0)$ to all mesh points in the domain. Some comments:
   (a) Use a cost function $C(x,y) = 1$.
   (b) You don't have to worry about optimally programming the search for the smallest of the accepted points. Just use anything, no matter how slow, for that step.

2. Repeat, but this time let $C(x, y) = 1$ for $x \leq 1/2$, and $C(x, y) = 2$ for $x \leq 1/2$.
3. Draw the shortest path from $(0.75, 0.75)$ to $(0, 0)$
4. put rectangular obstacles in the domain, where the cost function is one million, and one everywhere else. Find the cheapest cost to reach all points from $(0, 0)$ to any point in the domain (and not in an obstacle).

Solution

We implemented the method and the results are as follows:

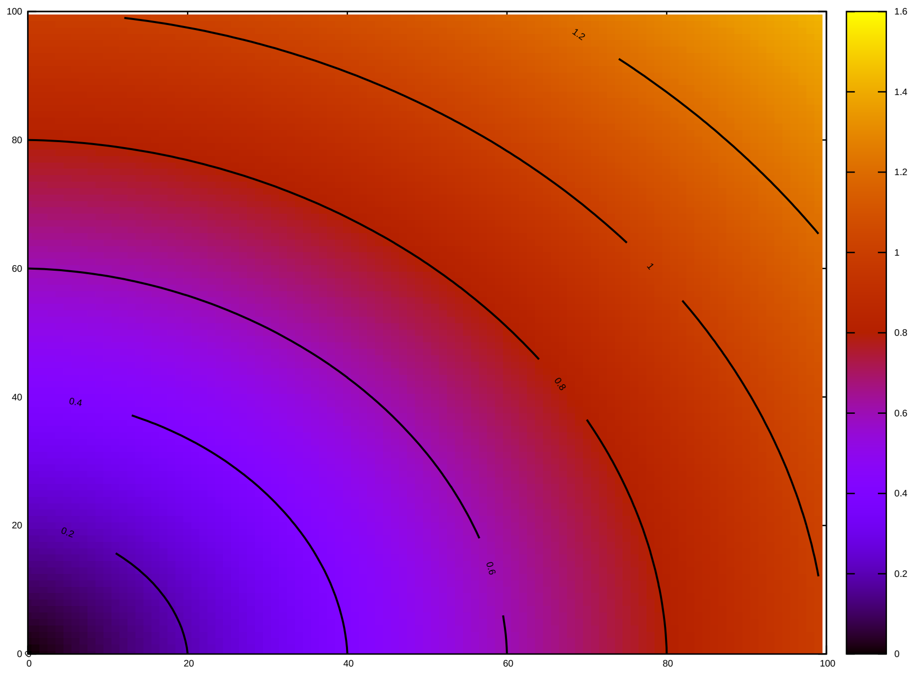

Figure 1: Iso contours of cost function, using $c(x, y) = 1$

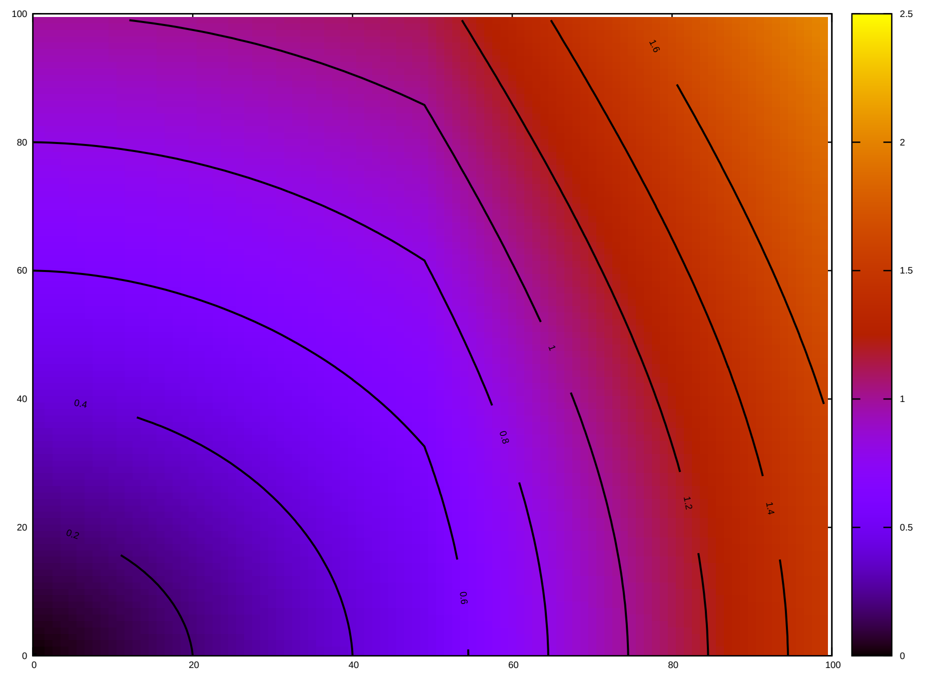

Figure 2: Iso contours of cost function, using $c(x,y) = 1$ for $x \leq 0.5$ and $c(x,y) = 2$ for $x \geq 0.5$

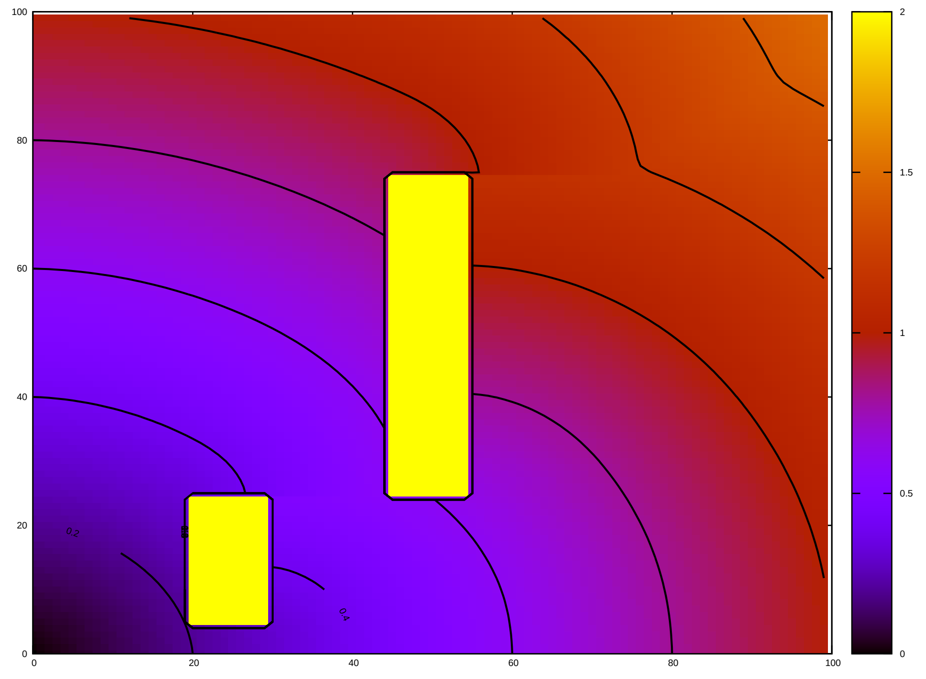

Figure 3: Iso contours of cost function, using $c(x,y) = 1$ and some obstacles

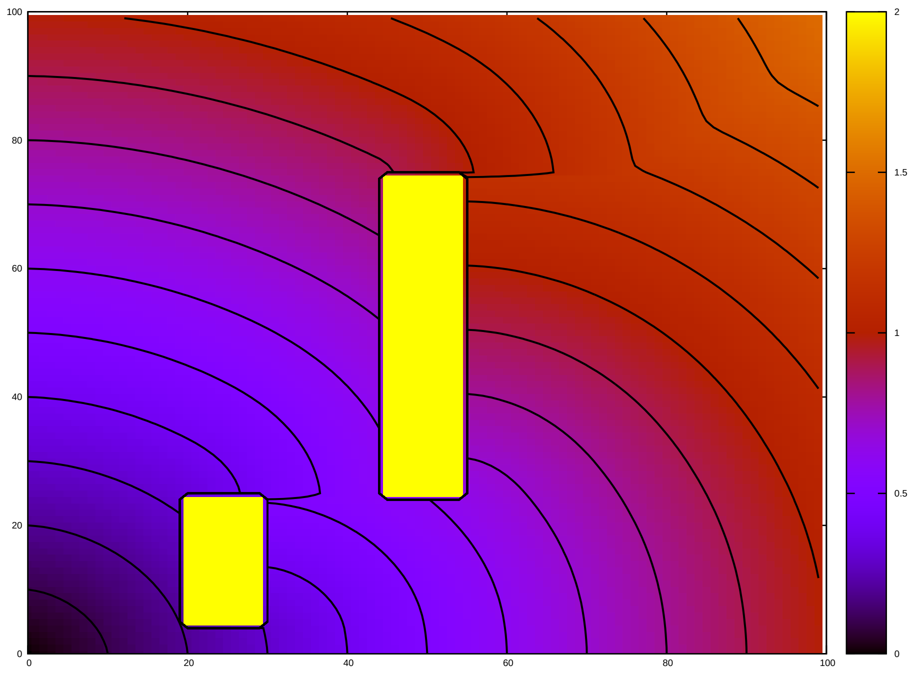

Figure 4: Iso contours of cost function, using $c\left(x,y\right)=1$ and some obstacles

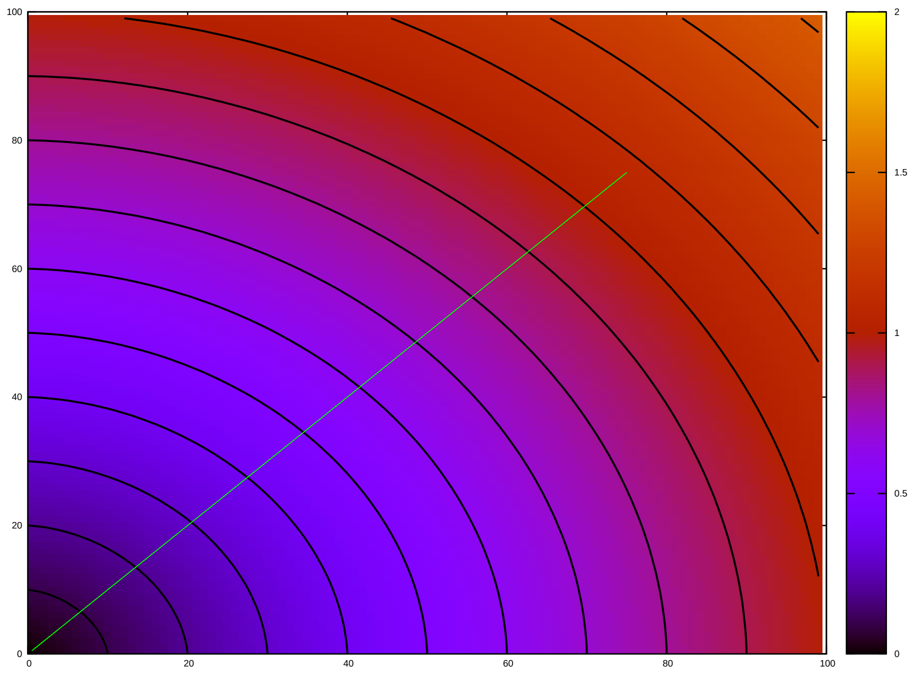

Figure 5: Iso contours of cost function and path, using $c\left(x,y\right)=1$

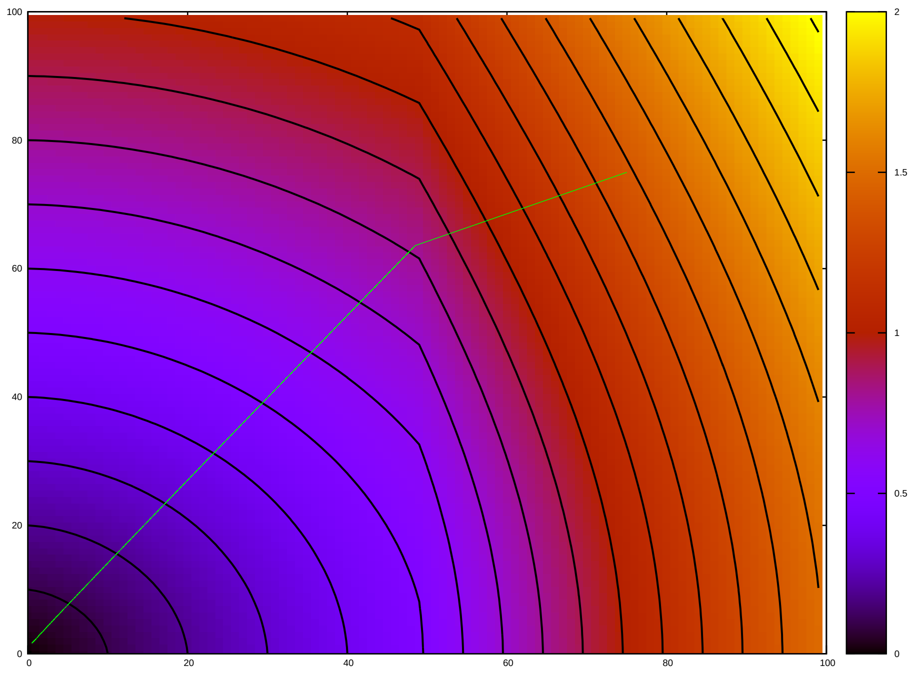

Figure 6: Iso contours of cost function and the shortest path, using $c(x, y) = 1$ for $x \leq 0.5$ and $c(x, y) = 2$ for $x \geq 0.5$

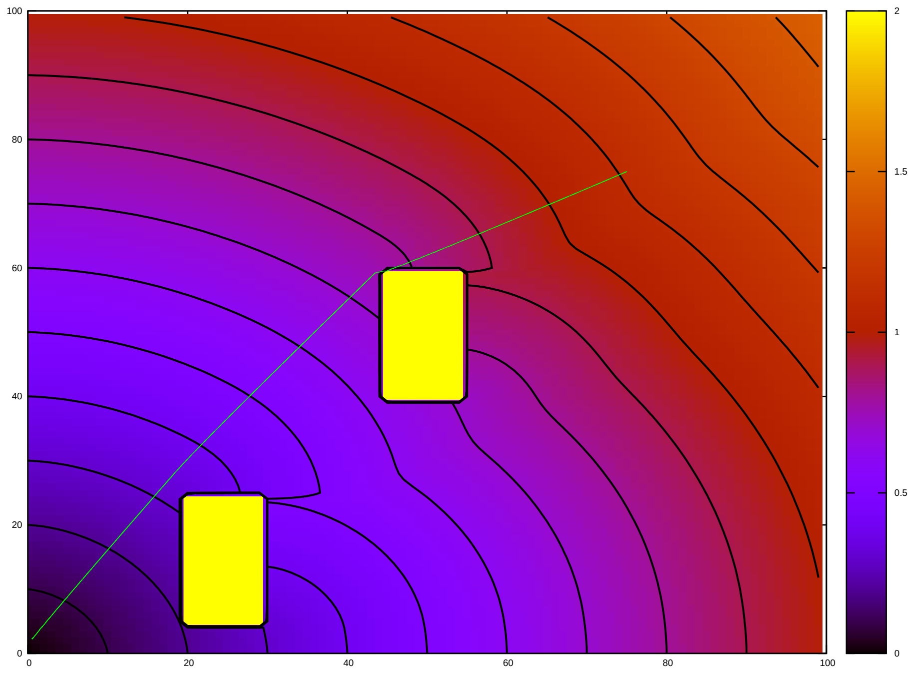

Figure 7: Iso contours of cost function and the shortest path, using $c(x, y) = 1$ and some obstacles

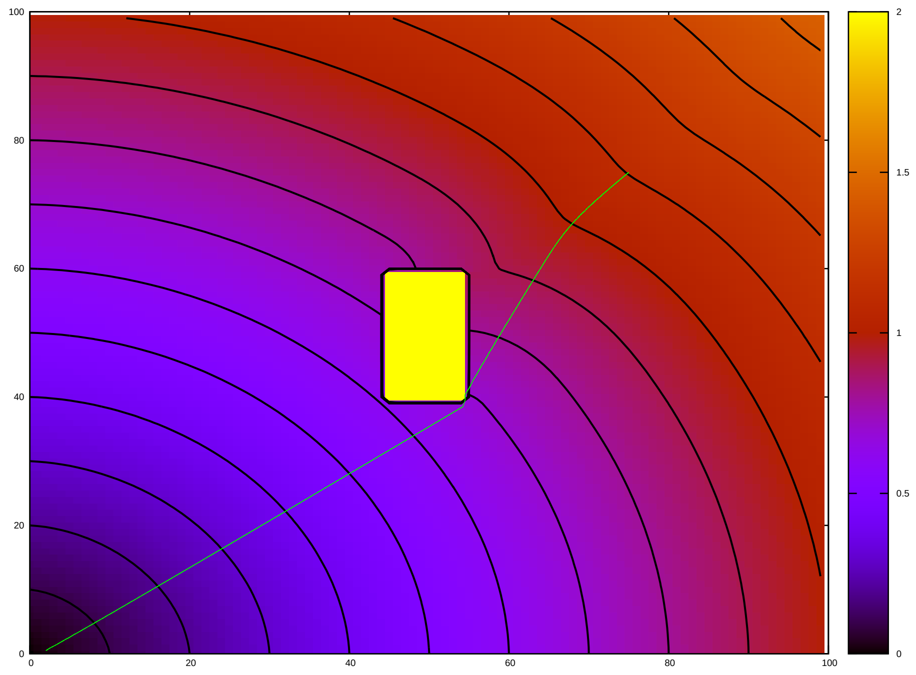

Figure 8: Iso contours of cost function and the shortest path, using $c\,(x, y) = 1$ and an obstacle

Conclusion

The fast marching method for solving Eikonal equation was implemented. We did not implement the heap sort for computing the smallest unvisited node. The shortest paths are also found. This was solved for different conditions of the domain, with varying $c(x, y)$ and with obstacles.