Notes on implementing the Immersed Boundary Method to solve the Eikonal Equation in 2D

Balaje K & Sanath Keshav

June 9, 2016

Contents

Chapter 1

Eikonal equation in 2D

The Eikonal equation in 2D is given by the following PDE

$$|\nabla u| = 1 \tag{1.1}$$

where $u = u(x, y)$ in the 2D plane. Consider first a 2D rectangular domain, on which we first illustrate a Finite Difference Scheme to solve the Eikonal Equation. We start with the following boundary value problem.

$$\begin{aligned} |\nabla u| &= 1 \quad \text{in } \Omega := (-1,1) \times (-1,1) && (1.2) \\ u &= 0 \quad \text{on } \Gamma && (1.3) \end{aligned}$$

where Γ denotes the boundary of the rectangle under consideration.

First we take the PDE (1.3) and add a transient term u_t to it. This makes it a time dependent problem. The idea is to solve this problem till it reaches the steady state. The corresponding initial boundary value problem is

$$\begin{aligned} u_t + |\nabla u| &= 1 \quad \text{in } (-1,1) \times (-1,1) \times (0,T] && (1.4) \\ u(x,y,0) &= 0 \quad x, y \in \Omega && (1.5) \\ u(x,y,t) &= 0 \quad x, y \in \Gamma && (1.6) \end{aligned}$$

A finite difference mesh on a rectangle is shown in the following Figure 1.1. The grid spacing along x-axis is given by h_x and along y-axis is given by h_y. A finite difference scheme due to Rouy and Tourin [4] is formulated below.

Let

$$h_x = \frac{2}{N_x} \tag{1.7}$$

$$h_y = \frac{2}{N_y} \tag{1.8}$$

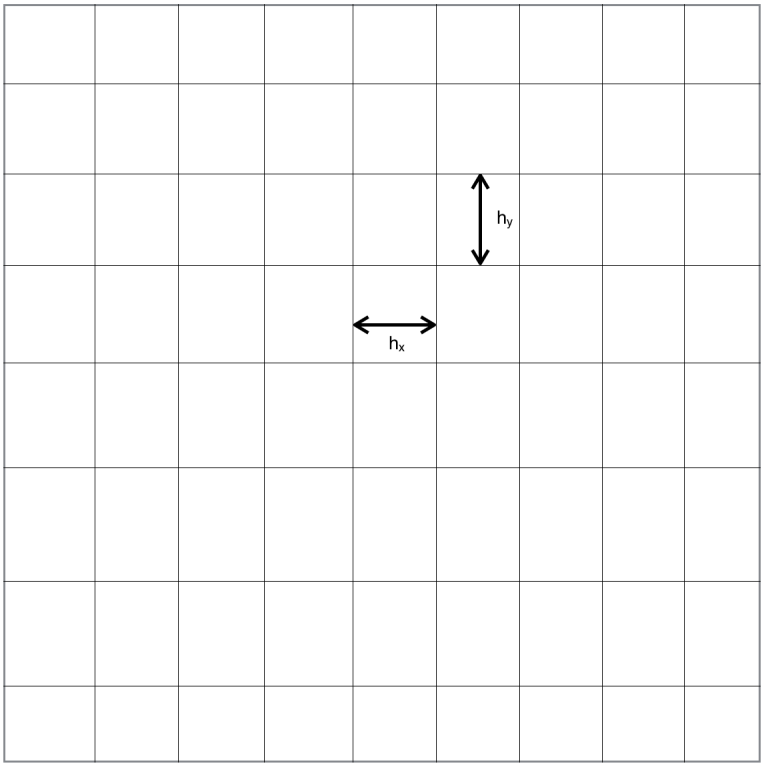


Figure 1.1: Rectangular Mesh

where N_x and N_y is the number of discretization along x and y directions respectively. Now $x_i = -1 + ih_x$ and $y_j = -1 + jh_y$ where $i = 1, 2, \ldots N_x - 1$ and $j = 1, 2, \ldots N_y - 1$.

The time discretization, Δt, can be set using a suitable CFL Condition so that the solution does not diverge for large time steps. Let u_{ij}^n denote the exact solution of the problem at the grid point (x_i, y_j) and at time $t = n\Delta t$.

Let $v_{ij}^n \approx u_{ij}^n$ at the grid point (x_i, y_j). The scheme proposed by Rouy and Tourin to solve

the Eikonal Equation is given by

$$v_{ij}^{n+1} = v_{ij}^{n} - \Delta t \left(\sqrt{D_x^2 + D_y^2} - 1 \right) \tag{1.9}$$

$$D_x = max \left(\frac{v_{i+1,j} - v_{i,j}}{h_x}, \frac{v_{i-1,j} - v_{i,j}}{h_x}, 0 \right) \tag{1.10}$$

$$D_y = max \left(\frac{v_{i,j+1} - v_{i,j}}{h_x}, \frac{v_{i,j-1} - v_{i,j}}{h_x}, 0 \right) \tag{1.11}$$

It is shown in [2] that this scheme is consistent, stable, monotone and admits a comparison principle. Thus the scheme converges to a unique viscosity solution.

The update formula is run till the steady state is achieved. That is, a preset tolerance ϵ such that $\|u^{n+1} - u^n\| < \epsilon$ must be satisfied so that the steady state is achieved. The results are shown in chapter 3.

Chapter 2

Immersed Boundary Method

So far, the fintie difference techniques can be easily applied to a rectangular cartesian grid. The objective is to extend the same finite difference techniques to a variety other domains, for example - *Circle, L-Domain, Circle with hole* etc.

To illustrate the technique, we consider a circular domain. For comfort, let us consider a unit circle. We take a rectangular cartesian grid shown in Figure 1.1 and **"immerse"** the circluar domain into the rectangular mesh as shown in Figure 2.1. This technique is known as the immersed boundary method [3] which is widely used to solve problems in Fluid Mechanics.

The first challenge in this method is to track the interior and boundary of the immersed domain. For this we introduce a data structure called the **inout** matrix. The **inout** matrix is defined as follows,

$$inout_{ij} = \begin{cases} 0, & \text{if } (x_i, y_j) \text{ lies inside the immersed domain} \\ 1, & \text{otherwise} \end{cases} \tag{2.1}$$

.
The way to track the boundary of the immersed domain using the **inout** matrix is natural. We simply ignore the grid points whose **inout** entry is 1 and set the value $u(x, y) = 0$ at those places. We then solve the eikonal equation using (1.9) - (1.11) only inside the immersed boundary where the **inout** value is 0.

To apply the boundary conditions, we are in a position to obtain the points marked in **red** in Figure 2.1. These points are easily obtained by solving the grid lines $x = -1 + ih_x$ or $y = -1 + jh_y$ with the equation of the circle. Note that, from the Dirichlet boundary condition, the value of $u(x, y)$ is prescribed in the problem at these points.

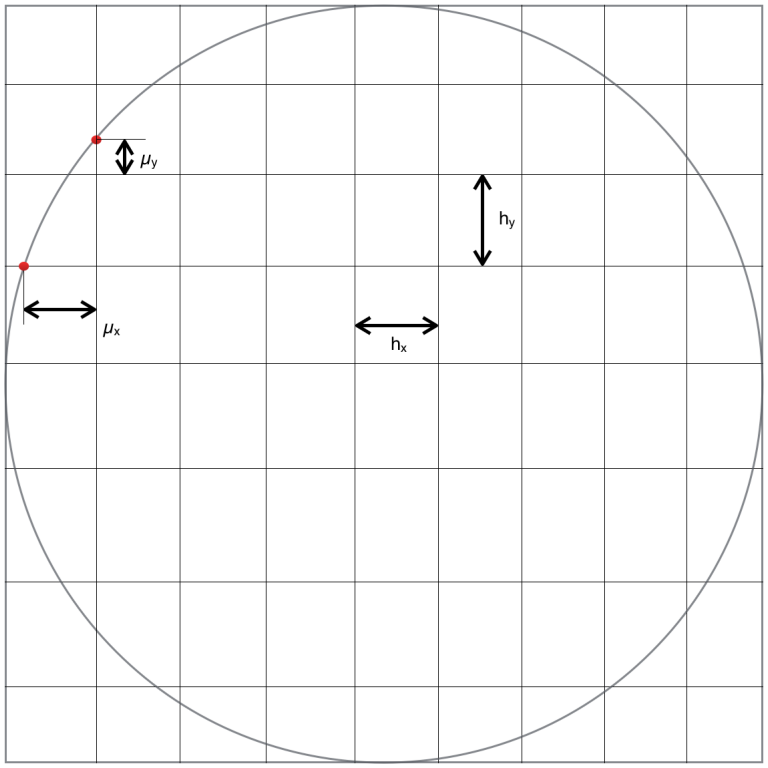


Figure 2.1: Immersed Circle

When we apply the finite difference formula on the point which is immediately next to the **red** point - inside the domain, we are in a position to use the distances μ_x and μ_y to calculate the finite differences. This is the place where we choose to ignore the points outside the immersed boundary, and take the points that lie on the circle instead.

With the distances μ_x and μ_y, the update formula (1.9)-(1.11) near the immersed boundary becomes,

$$v_{ij}^{n+1} = v_{ij}^{n} - \Delta t \left(\sqrt{D_x^2 + D_y^2} - 1 \right) \tag{2.2}$$

$$D_x = max \left(\frac{v_{i+1,j} - v_{i,j}}{\mu_1}, \frac{v_{i-1,j} - v_{i,j}}{\mu_2}, 0 \right) \tag{2.3}$$

$$D_y = max \left(\frac{v_{i,j+1} - v_{i,j}}{\mu_3}, \frac{v_{i,j-1} - v_{i,j}}{\mu_4}, 0 \right) \tag{2.4}$$

where $\mu_{1,\ldots,4}$ is chosen accordingly, depending on the position of top, bottom, left or right neighboring grid points. If the grid point above the current point, lies on the boundary, then $\mu_3 = \mu_y$ and $\mu_1 = \mu_2 = h_x$, $\mu_4 = h_y$ and so on. There may be cases where two of the neighboring grid points may lie on the boundary. All those cases must be taken proper care of. Results of some numerical experiments are shown in the next chapter.

Chapter 3

Results

Some Numerical Experiments were carried out using the Rouy and Tourin scheme for a variety of domains. First we illustrate the solution using a simple rectangular (a square rather) domain.

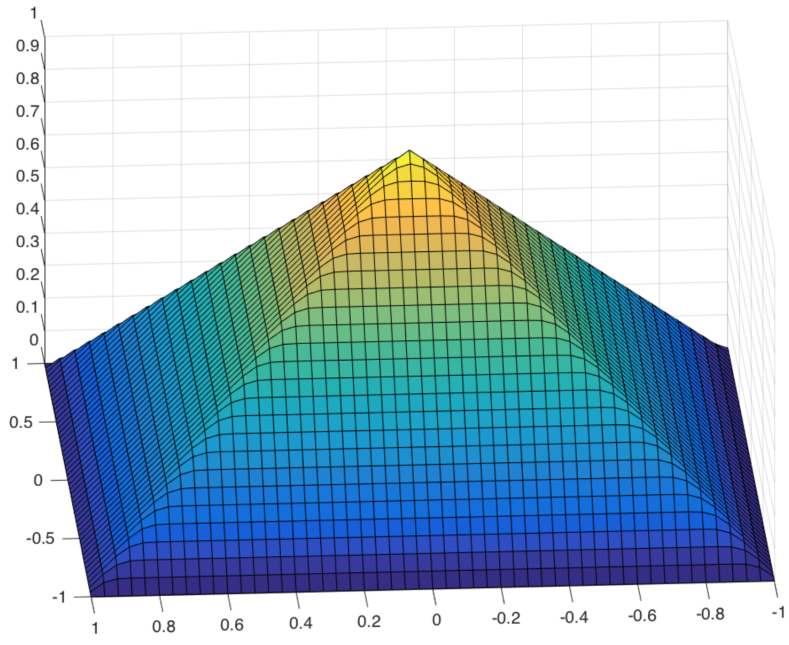


Figure 3.1: Solution in a square domain

The profile shown in Figure 3.1 is the **Viscosity Solution** of the Eikonal Equation. A 50×50 grid was used to generate the profile till steady state with $\epsilon = 10^{-5}$. Some ob-

servations can be made from the solution shown in Figure 3.1. The equation (1.3) does not accept solutions with a local minima (not a viscosity solution [1]). This gives an idea about how the solution should look like on other domains. For example, a right circlular cone is observed in the case of a circle and so on.

Numerical Experiments were conducted using **Fortran90** to find out the order of convergence of the scheme with $\epsilon = 10^{-16} \approx$ Machine Error. The **inout** matrix for a circle with a 10×10 mesh is shown in Figure 3.2

```
inout =

     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     0     0     0     0     0     1     1     1
     1     1     0     0     0     0     0     0     0     1     1
     1     0     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     0     1
     1     1     0     0     0     0     0     0     0     1     1
     1     1     1     0     0     0     0     0     1     1     1
     1     1     1     1     1     1     1     1     1     1     1
```

Figure 3.2: **Inout** Matrix for 10×10 mesh

With this **inout** matrix, we compute the necessary distances near the boundary, for use in the Finite Difference Formula. Following is the profile, obtained when the Rouy and Tourin Scheme is run for 50×50 mesh on the master rectangle with steady state $\epsilon = 10^{-16}$.

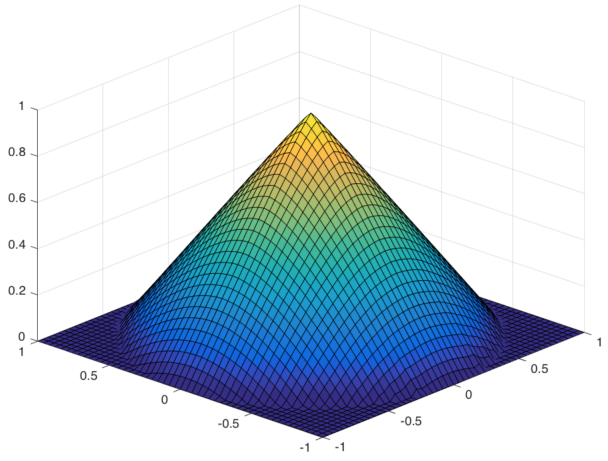
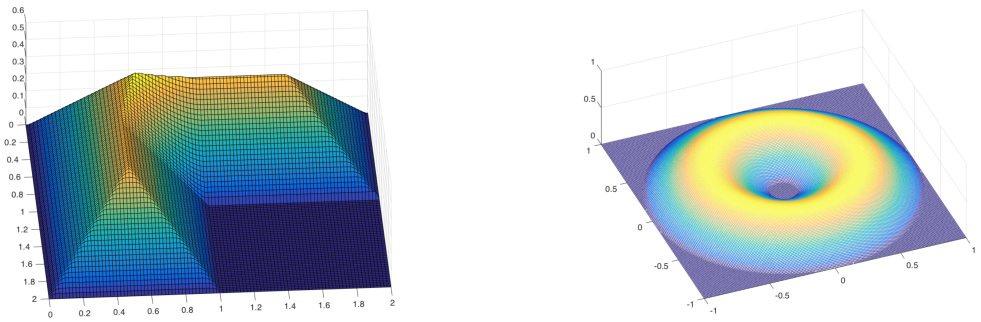


Figure 3.3: Solution in a circular domain

Using **Fortran90**, order of convergence analysis was performed on the scheme and the results are tabulated.

N	Error	Order of Convergence
10	0.9878E-01	0.685
20	0.6145E-01	0.728
40	0.3710E-01	0.738
80	0.2223E-01	0.761
160	0.1311E-01	0.783
320	0.0762E-01	0.800
640	0.0437E-01	-

The same method can be used to generate profiles for a variety of domains as shown. The idea is to tweak the **inout** matrix corresponding to the shape of the domain and appropriately take care of the immersed boundary grid points.



(a) L-Domain

(b) Circle with a hole in the middle

Figure 3.4: Other Domains

Figure 3.4a and Figure 3.4b shows the profile that is obtained when the Immersed Boundary Technique is applied to solve the Eikonal Equation on a L-Shaped Domain and a circle with a hole in it (non-convex domains) respectively. Similar Experiments can be conducted to model various problems in a variety of domains.

Chapter 4

Adaptive CFL condition for accelerating convergence

Performance of the code can be improved by changing size of the time step used at each grid point. To increase the speed of computation, we use a higher value of Δt at the interior (as the mesh parameter h is larger and hence the limit on the time step Δt is also higher) and lower value of Δt on points near the boundary.

With the update formula,

$$v_{i,j}^{n+1} = v_{i,j}^{n} - \Delta t(\dots \tag{4.1}$$

we compute the solution at $t = t_{n+1}$ using $t = t_n$. Refer Figure 4.1.

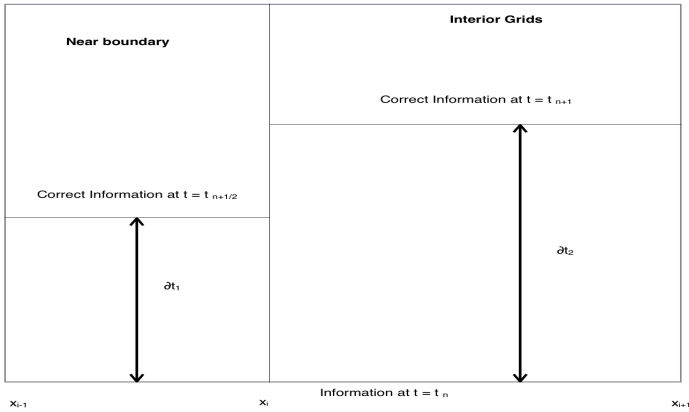


Figure 4.1: Varying time steps

We use the CFL Condition to find out the time step Δt with respect to the mesh size h. The time step will be **larger** where the mesh is **coarse** and *smaller* where the mesh is *fine*. With this we march faster in time in the interior and march slowly in time near the boundaries. So, to compute the solution in x_i (Rf. Figure 4.1), we require information at x_{i-1} and x_{i+1} at all times.

So the solution $u = u(x, y, t)$ will not give us the right result at a particular time t, because some grid points are evaluated at different times and the solution $u(x, y, t)$ would not make any sense.

But as the information is correctly propogated over time $t \to \infty$, the solution is obtained as all the grid values would have reached the steady state, but with different speeds.

We observe that the speed of computation has immensely improved and we were able to obtain the same order of convergence without the adaptive time stepping.

N	Error	Order of Convergence
10	0.9878E-01	0.685
20	0.6145E-01	0.728
40	0.3710E-01	0.738
80	0.2223E-01	0.761
160	0.1311E-01	0.783
320	0.0762E-01	0.800
640	0.0437E-01	0.814
1280	0.0249E-01	0.834
2560	0.0139E-01	-

4.0.1 Stability

In this section, we discuss the stability of the finite difference scheme. For this we consider the 1D Advection Equation and this idea can be extended for the Eikonal Equation as well.

$$u_t + u_x = 0 \tag{4.2}$$

Since the wave is propogating towards the right, we use the backward difference scheme.

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{h}\left(u_i^n - u_{i-1}^n\right) \tag{4.3}$$

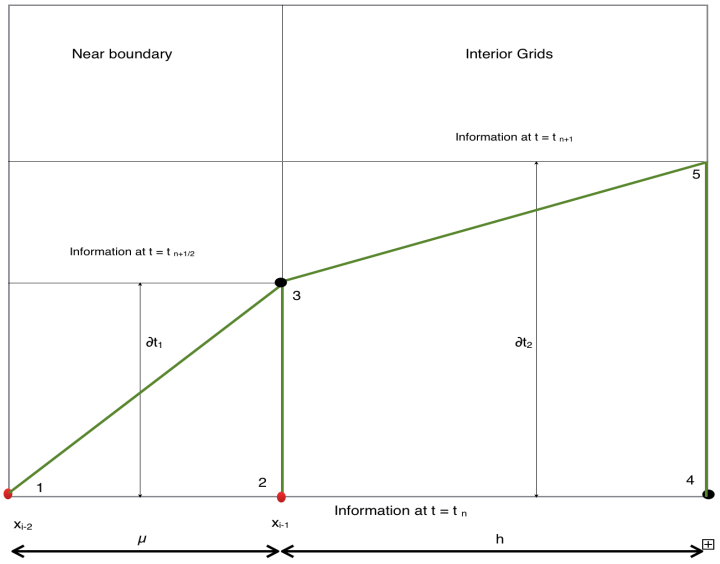


Figure 4.2: The Finite Difference stencil

Now consider the finite difference stencil, shown in Figure 4.2. The values of v at points (1)-(5) is shown in the following table. For brevity, let us denote the value of v at some t in $[t_n, t_{n+1}]$ as $v_i^{n+\frac{1}{2}}$

Point	Value
1	v_{i-2}^n
2	v_{i-1}^n
3	$v_{i-1}^{n+\frac{1}{2}}$
4	v_i^n
5	v_i^{n+1}

To find the values of $v_{i-1}^{n+\frac{1}{2}}$ and v_i^{n+1} using the known v_i^n, v_{i-1}^n, v_{i-2}^n, we have the upwind scheme,

$$v_{i-1}^{n+\frac{1}{2}} = v_{i-1}^n - \frac{\partial t_1}{h}\left(v_{i-1}^n - v_{i-2}^n\right) \tag{4.4}$$

$$v_i^{n+1} = v_i^n - \frac{\partial t_2}{h}\left(v_i^n - v_{i-1}^{n+\frac{1}{2}}\right) \tag{4.5}$$

Combining (4.4) and (4.5), we have,

$$v_i^{n+1} = v_i^n - \frac{\partial t_2}{h}\left(v_i^n - \left(v_{i-1}^n - \frac{\partial t_1}{h}\left(v_{i-1}^n - v_{i-2}^n\right)\right)\right) \tag{4.6}$$

Rearranging,

$$v_i^{n+1} = \left(1 - \frac{\partial t_2}{h}\right)v_i^n + \left(1 - \frac{\partial t_1}{h}\right)\left(\frac{\partial t_2}{h}\right)v_{i-1}^n + \left(\frac{\partial t_1}{h}\frac{\partial t_2}{h}\right)v_{i-2}^n \tag{4.7}$$

$$\implies v_i^{n+1} = H(v_{i-2}^n, v_{i-1}^n, v_i^n) \tag{4.8}$$

For the scheme to be monotone ,H must be non-decreasing in each of its variable. If we check for the same, we get

$$\partial t_1 \leq h$$
$$\partial t_2 \leq h$$

Combining these two we have the monotonicity condition.

$$\max(\partial t_1, \partial t_2) \leq h \tag{4.9}$$

The scheme is consistent and from (4.9), we can say that the scheme is convergent to the exact solution.

Bibliography

[1] Crandall, M. G., and Lions, P.-L. Viscosity solutions of hamilton-jacobi equations. *Trans. Amer. Math. Soc. 277 (1983), 1-42* (1983).

[2] Ju, Y. C. Hamilton-jacobi equations for computer vision: Application in shape from shading, 2010.

[3] Peskin, C. S. The immersed boundary method. *Acta Numerica (2002)* (2002), 479517.

[4] Rouy, E., and Tourin, A. A viscosity solutions approach to shape-from- shading. *SIAM Journal on Numerical Analysis* (1992), 867–884.