Understanding Eikonal Equation

So in this section I wish to understand the **Fast Sweep Method** that was used in the previous attempts to solve the hughe's problem.

Problem Description:

The problem is formulated as follows. I have taken a function `phi` and this is function of `X`. And since it's a `1D` problem i will have to discritize the domain (which is 1D line) into grid points. I have discritize it into 80 points.

$$\begin{cases} \left(\frac{\partial \phi}{\partial x}\right)^2 + \left(\frac{\partial \phi}{\partial y}\right)^2 = f(x,y)^2, \\ \phi(1,y) = \phi(N_x,y) = \phi(x,1) = \phi(x,N_x) = 0, \quad \text{which means } \phi \text{ is zero on the boundary,} \\ f(x,y) = \sqrt{(y(y-1)(2x-1))^2 + (x(x-1)(2y-1))^2}. \end{cases}$$

[10pt]The problem is now completely set. We can solve it using the fast sweeping method and compare with the exact solution to

The solution to the baove problem could be analytically found out to be: $\phi(x) = \pm x$. And this is the solution to the problem. In the following section I have tried usign the newly known **FAST SWEEP ALGORITHM**, and try solving the above problem.

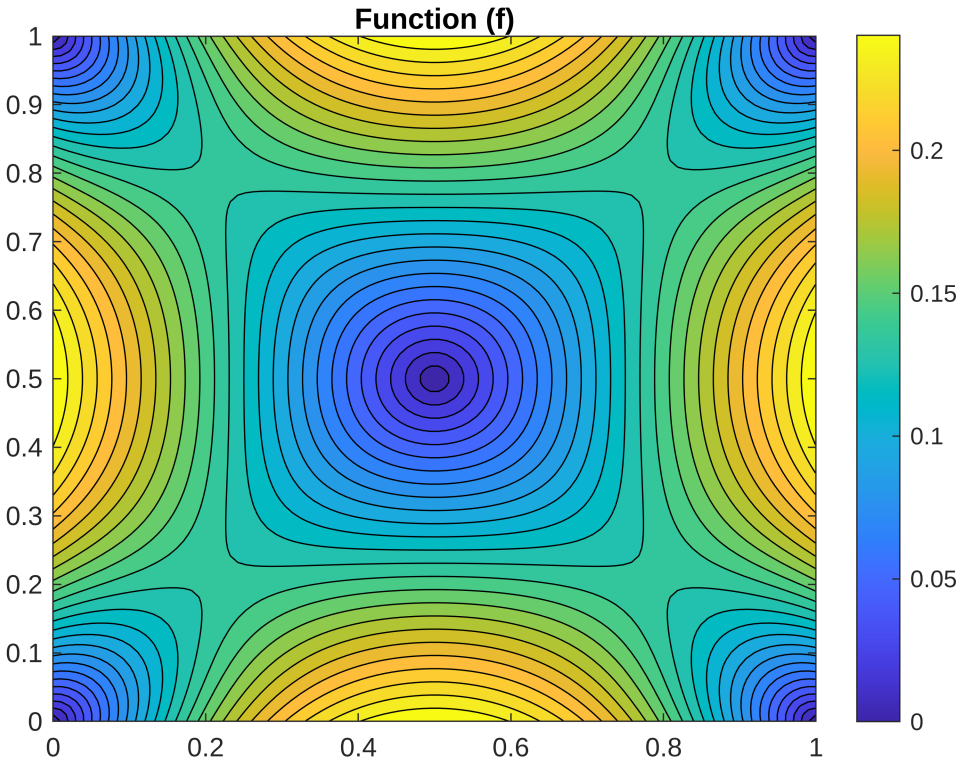
```
Nx = 100;
Ny = 100;
Lx = 1;
Ly = 1;
dx = Nx/Lx;
dy = Ny/Ly;
[x,y] = meshgrid(linspace(0,Lx,Nx),linspace(0,Ly,Ny));
```

Here we have defined the domain which is square one, with dimension being $1 * 1$ meter squared. And the mesh is discritized into 100 points in x direction and 100 points in the y direction. Now we once plot the known graph to the problem for better understanding of the algorithm This will be helpful to calculate the error contour for comparing both the solution.

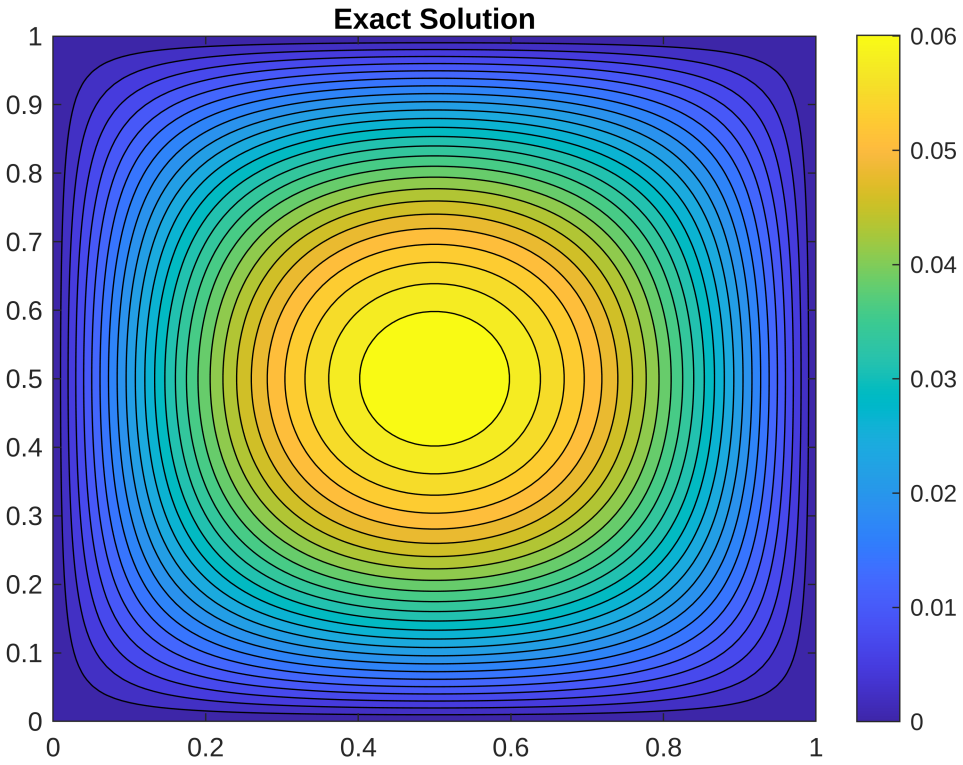
Fucntion (f)

For the reference, following is the plot of the function `f`. And this will be used to make the

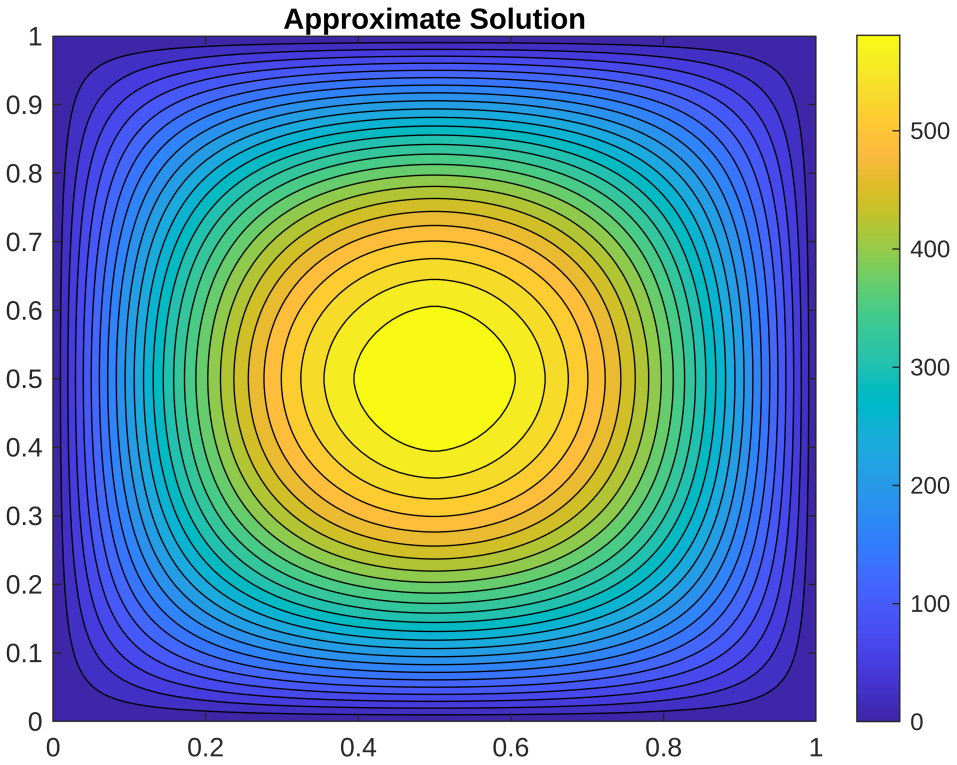
```
f_exact = ((y.*(y-1).*(2*x-1)).^2 + (x.*(x-1).*(2*y-1)).^2).^(0.5);
contourf(x,y,f_exact,25);
title("Function (f)");
colormap("parula");
colorbar;
```




```
Phi_exact = x.*(x-1).*y.*(y-1);
contourf(x,y,Phi_exact,25);
title("Exact Solution");
colormap("parula");
colorbar;
```




```
Phi_approx = fast_sweeping(f_exact,dx,Nx,Ny);
contourf(x,y,Phi_approx,25);
title("Approximate Solution");
colormap("parula");
colorbar;
```



It could be seen from the above that **FAST SWEEP METHOD** worked.

Given below is the code for the FAST SWEEP ALGORITHM.

```
function phi = fast_sweeping(f,dx,Nx,Ny)
    %Boundary Condition
    phi = inf(Ny,Nx);
    for sweep=1:100
        %These are the dirichlet condition that I have set while
        %formulating this problem.
        phi(:,1) = 0;
        phi(:,end) = 0;
        phi(end,:) = 0;
        phi(1,:) = 0;

       % This will take the function from 2 to n-1 in x and 2 to n-1 in y
        for i=2:Nx-1
            for j=2:Ny-1
                % Calculating the minimum value of phi, in it's neighbour in
x direction
                a = min(phi(i,j-1), phi(i,j+1));
                % Calculating the minimum value of phi, in it's neighbour in
y direct
                b = min(phi(i-1,j), phi(i+1,j));
                tmp = sort([a, b]);
```

```
                if abs(tmp(1)-tmp(2)) >= f(i,j)*dx
                    %phi = min(a,b) + (step_time)
                    phi(i,j) = tmp(1) + f(i,j)*dx;
                else
                    %phi = solution of quadratic equation
                    phi(i,j) = (tmp(1)+tmp(2) + ...
                        sqrt(2*(f(i,j)*dx)^2 - (tmp(1)-tmp(2))^2))/2;
                end
            end
        end

        % This will take the function from n-1 to 2 in x and n-1 to 2 in y
        for i=Nx-1:-1:2
            for j=Ny-1:-1:2
                % Calculating the minimum value of phi, in it's neighbour in
x direction
                a = min(phi(i,j-1), phi(i,j+1));
                % Calculating the minimum value of phi, in it's neighbour in
y direct
                b = min(phi(i-1,j), phi(i+1,j));
                tmp = sort([a, b]);
                if abs(tmp(1)-tmp(2)) >= f(i,j)*dx
                    %phi = min(a,b) + (step_time)
                    phi(i,j) = tmp(1) + f(i,j)*dx;
                else
                    %phi = solution of quadratic equation
                    phi(i,j) = (tmp(1)+tmp(2) + ...
                        sqrt(2*(f(i,j)*dx)^2 - (tmp(1)-tmp(2))^2))/2;
                end
            end
        end

        % This will take the function from 2 to n-1 in x and n-1 to 2 in y
        for i=2:Nx-1
            for j=Ny-1:-1:2
                % Calculating the minimum value of phi, in it's neighbour in
x direction
                a = min(phi(i,j-1), phi(i,j+1));
                % Calculating the minimum value of phi, in it's neighbour in
y direct
                b = min(phi(i-1,j), phi(i+1,j));
                tmp = sort([a, b]);
                if abs(tmp(1)-tmp(2)) >= f(i,j)*dx
                    %phi = min(a,b) + (step_time)
                    phi(i,j) = tmp(1) + f(i,j)*dx;
                else
                    %phi = solution of quadratic equation
                    phi(i,j) = (tmp(1)+tmp(2) + ...
                        sqrt(2*(f(i,j)*dx)^2 - (tmp(1)-tmp(2))^2))/2;
                end
```

```
            end
        end

        % This will take the function from n-1 to 2 in x and 2 to n-1 in y
        for i=Nx-1:-1:2
            for j=2:Nx-1
                % Calculating the minimum value of phi, in it's neighbour in
x direction
                a = min(phi(i,j-1), phi(i,j+1));
                % Calculating the minimum value of phi, in it's neighbour in
y direct
                b = min(phi(i-1,j), phi(i+1,j));
                tmp = sort([a, b]);
                if abs(tmp(1)-tmp(2)) >= f(i,j)*dx
                    %phi = min(a,b) + (step_time)
                    phi(i,j) = tmp(1) + f(i,j)*dx;
                else
                    %phi = solution of quadratic equation
                    phi(i,j) = (tmp(1)+tmp(2) + ...
                        sqrt(2*(f(i,j)*dx)^2 - (tmp(1)-tmp(2))^2))/2;
                end
            end
        end
    end
end
```