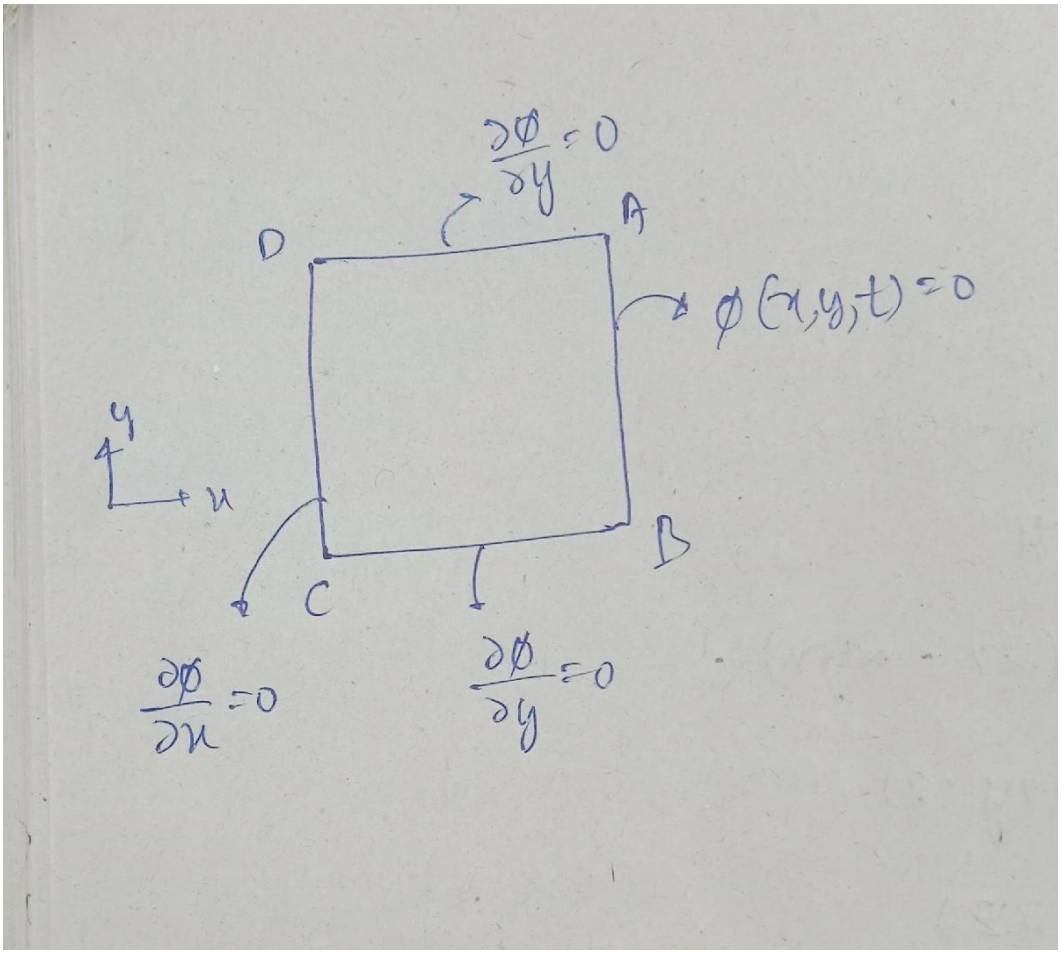
Understanding the Eikonal Equation

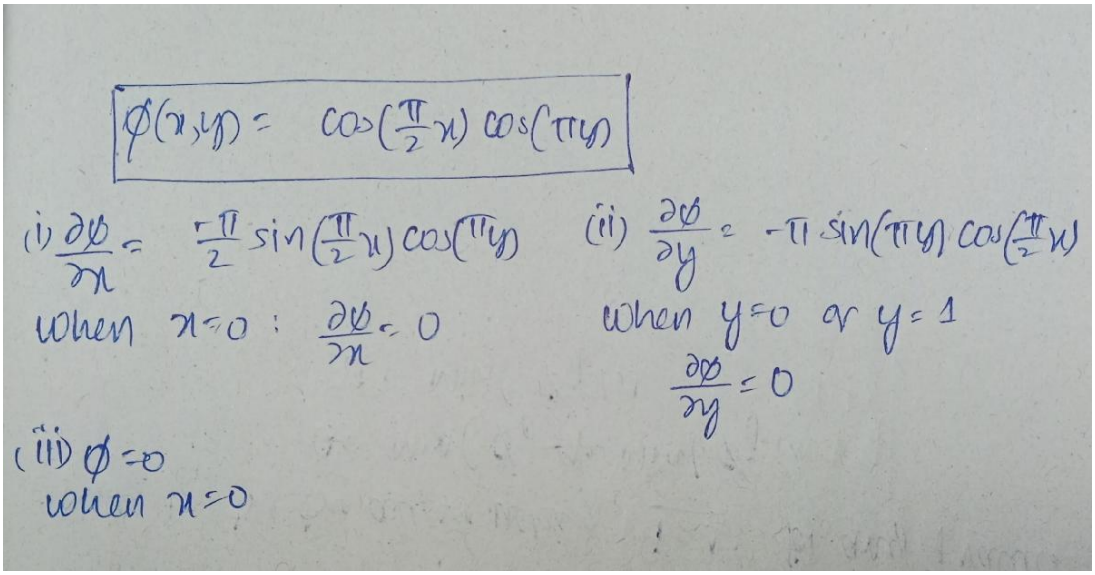
For the purpose of understanding I have designed a synthetic problem, that I ahve used to test my notion of implementing neumann boundary condition using fast sweep algorithm.

So the problem is as follows:

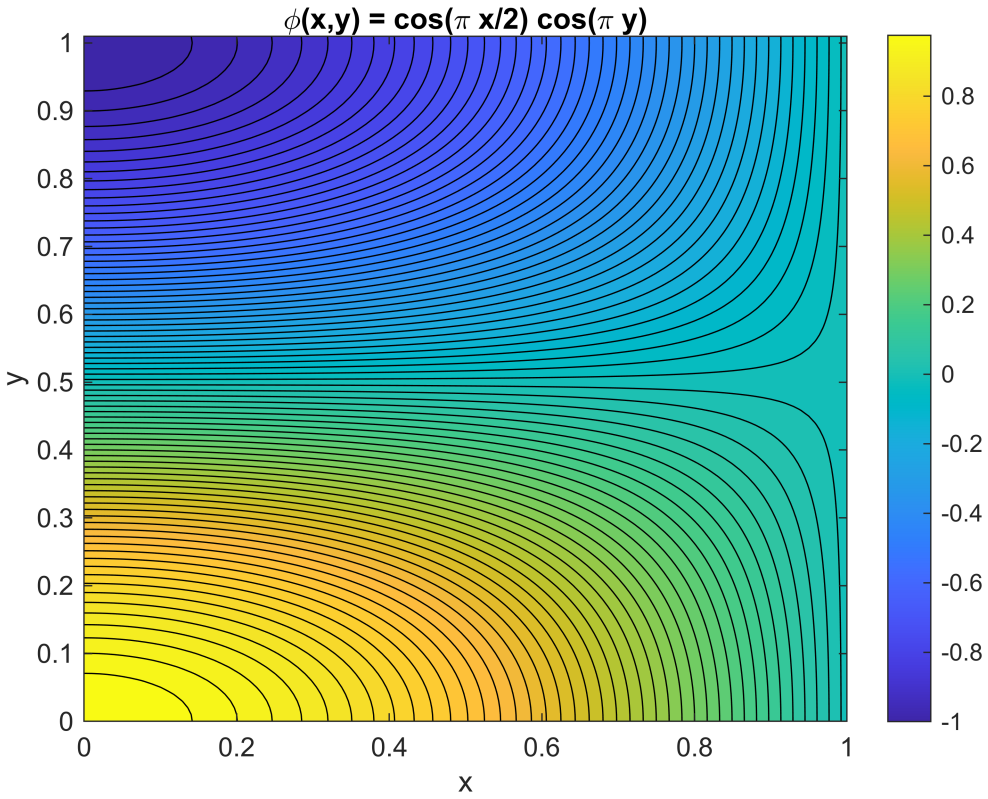
I take the test function to be $\phi(x, y, t)$ and have the following property:



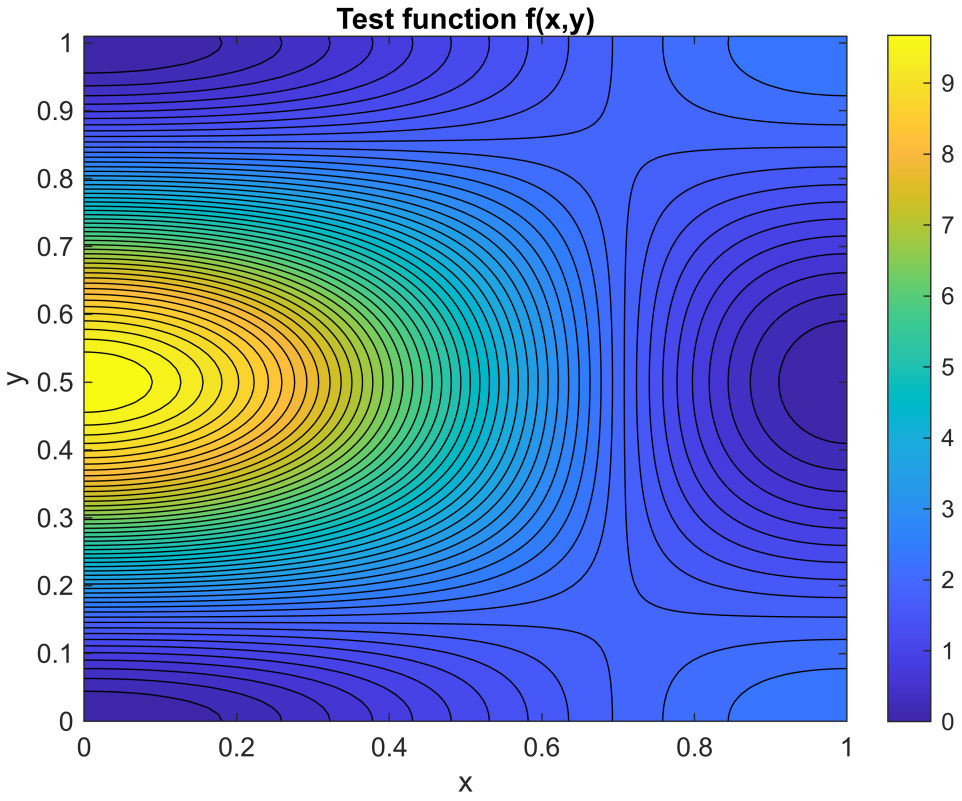
Given below is the example of the function that could be tested:



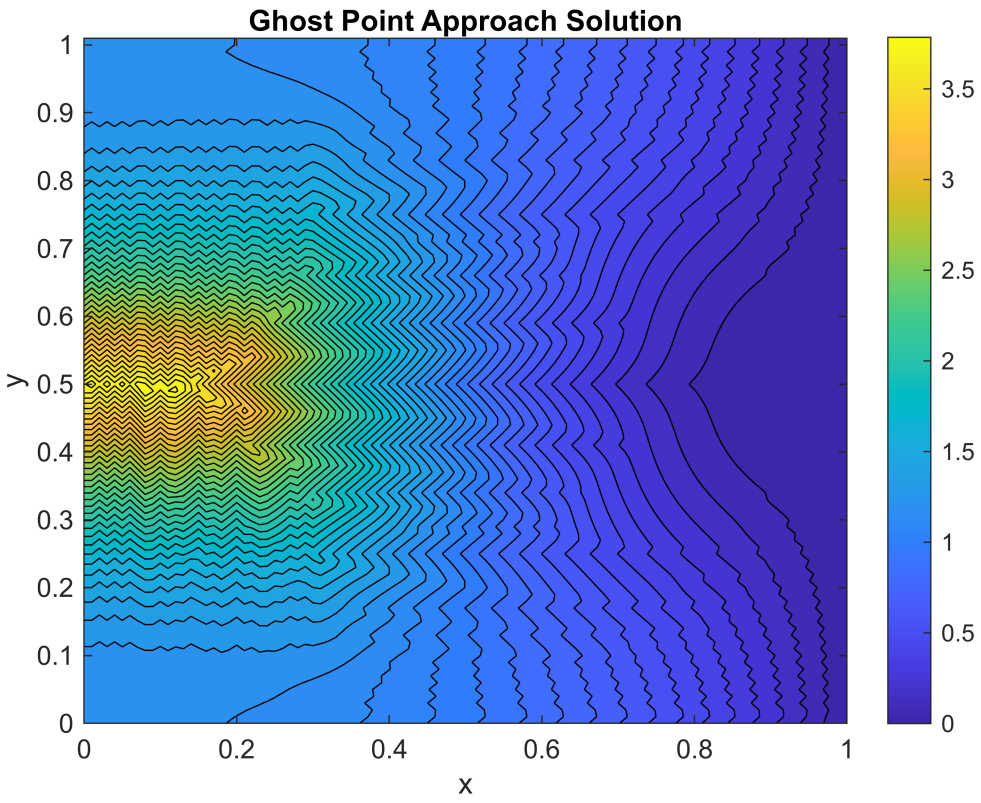

```
clear all;
% Parameters
I = 100;   % number of interior points in x
J = 100;   % number of interior points in y
Lx = 1;     % domain length in x
Ly = 1;     % domain length in y

% Grid spacing
dx = Lx / I;
dy = Ly / J;

% Index ranges with ghost points
x_index = 0:I;         % total I+1 in x (1 ghost at right boundary)
y_index = 0:J+1;       % total J+2 in y (1 ghost at top and bottom)

% Physical coordinates
X = x_index * dx;
Y = y_index * dy;

[x, y] = meshgrid(X, Y);

phi = cos((pi/2) * x) .* cos(pi * y);

% Plot phi
figure;
contourf(X, Y, phi, 80, 'LineColor', 'k');
colorbar;
xlabel('x');
ylabel('y');
title('\phi(x,y) = cos(\pi x/2) cos(\pi y)');
```



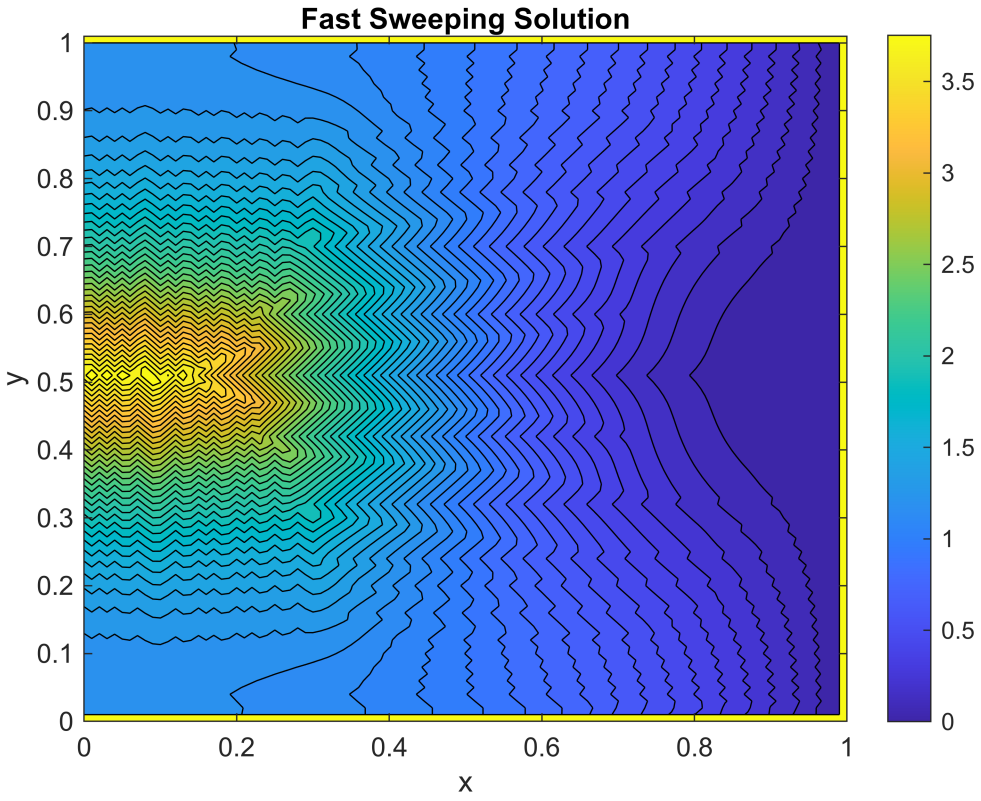

```
%% function f
Nx = I; % define Nx same as I (interior in x)
Ny = J; % define Ny same as J (interior in y)
f = (pi^2/4) * (sin((pi/2)*x).^2 .* cos(pi*y).^2) + pi^2 * (sin(pi*y).^2 .* cos((pi/
2)*x).^2);
figure;
contourf(X, Y, f, 50);
colorbar;
xlabel('x'); ylabel('y'); title('Test function f(x,y)');
```



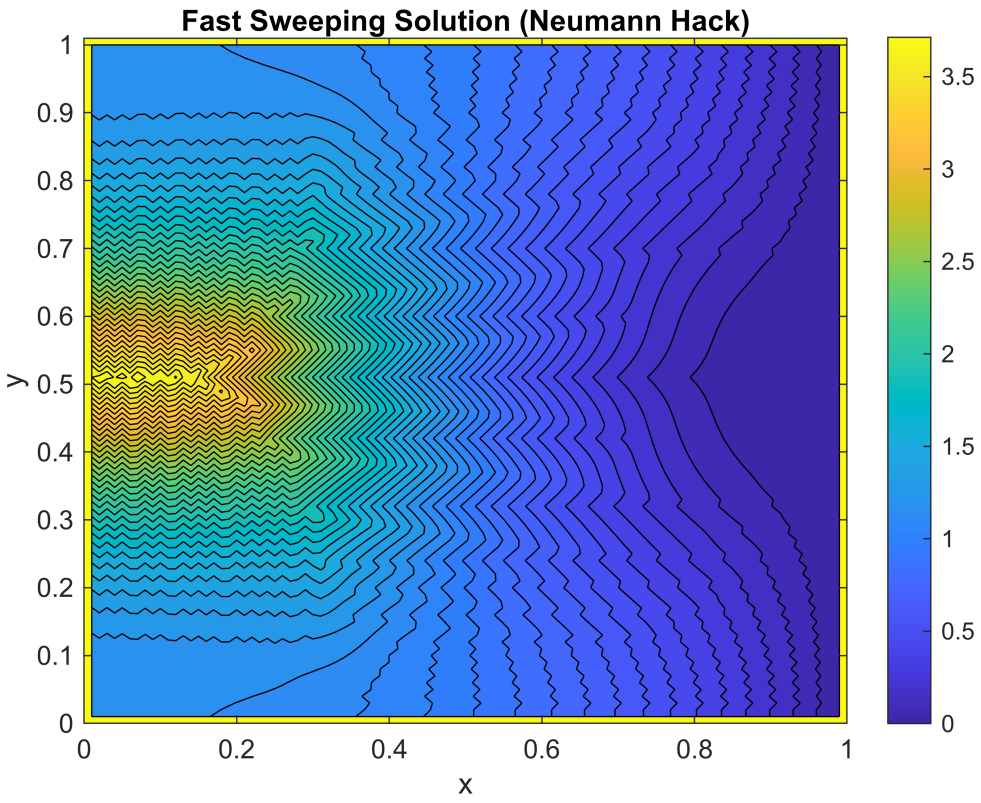

```
%% fast sweeping solver
phi_gp = ghost_point_approach(f, dx, dy, Nx, Ny);
figure;
contourf(X, Y, phi_gp, 50);
colorbar;
xlabel('x'); ylabel('y'); title('Ghost Point Approach Solution');
```




```
phi_fs = fast_sweeping(f, dx, dy, Nx, Ny);
phi_fs_plot = inf(J+2, I+1);          % same size as X, Y
phi_fs_plot(2:J+1, 1:I) = phi_fs;     % insert interior values

figure;
contourf(X, Y, phi_fs_plot, 50);
colorbar;
xlabel('x'); ylabel('y'); title('Fast Sweeping Solution');
```




```
%% fast sweeping with neumann hack
phi_nh = fast_sweeping_Neumann_Hack(f, dy, dx, Nx, Ny);
phi_nh_plot = inf(J+2, I+1);      % match size with X, Y
phi_nh_plot(2:J+1,1:I) = phi_nh;  % insert interior values

figure;
contourf(X, Y, phi_nh_plot, 50);
colorbar;
xlabel('x'); ylabel('y');
title('Fast Sweeping Solution (Neumann Hack)');
```



Fast Sweep Method - First Order Approximation

```
function phi = fast_sweeping(f,dy,dx,Nx,Ny)
    %Boundary Condition
    phi = inf(Ny,Nx);
    for sweep=1:200
        % The limit starts from column and row 2, and ends at column and row
        % This excludes the boundary points.
        % Updating the boundary points. For doing so i am going to sweep in
        % that too
        phi(:, Nx) = 0;       % Dirichlet on right boundary
        phi(:, 1)  = phi(:, 2); % Neumann left boundary
        phi(Ny, :) = phi(Ny-1, :); % Neumann top
        phi(1, :)  = phi(2, :);    % Neumann bottom


        for boundary_sweep = 1:2
            % Updating the side AB

            % Updating the side BC
            for i = 2 : Nx-1
                phi(1,i) = min((min(phi(1,i-1),phi(1,i+1)) + f(1,i)*dx),phi(1,i));
            end
            for i = Nx-1 : 2 : -1
                phi(1,i) = min((min(phi(1,i-1),phi(1,i+1)) + f(1,i)*dx),phi(1,i));
```

```
            end

            % Updating the side AD
            for i = 2 : Nx-1
                phi(Ny,i) = min((min(phi(Ny,i-1),phi(Ny,i+1)) +
f(Ny,i)*dx),phi(Ny,i));
            end
            for i = Nx-1 : 2 : -1
                phi(Ny,i) = min((min(phi(Ny,i-1),phi(Ny,i+1)) +
f(Ny,i)*dx),phi(Ny,i));
            end

            % Updating the side CD
            for j = 2 : Ny-1
                phi(j,1) = min((min(phi(j-1,1),phi(j+1,1))+f(j,1)*dy),phi(j,1));
            end
            for j = Ny-1 : 2 : -1
                phi(j,1) = min((min(phi(j-1,1),phi(j+1,1))+f(j,1)*dy),phi(j,1));
            end
        end

        % Now we are done with updating the boundary phi values with the
        % neumann type boundary condition at the three sides and one
        % dirichlet type boundary condition at the other side AB

        % Sweep direction 1
        for i=2:Nx-1
            for j=2:Ny-1
                phi(j,i) = update_point(phi, f, i, j, dx);
            end
        end

        % Sweep direction 2
        for i = Nx-1 : 2 : -1
            for j = 2 : Ny-1
                phi(j,i) = update_point(phi, f, i, j, dx);
            end
        end

        % Sweep direction 3
        for i = 2 : Nx-1
            for j = Ny-1 : 2 : -1
                phi(j,i) = update_point(phi, f, i, j, dx);
            end
        end

        % Sweep direction 4
        for i = Nx-1 : 2 : -1
            for j = Ny-1 : 2 : -1
                phi(j,i) = update_point(phi, f, i, j, dx);
```

```
            end
        end
    end
end
```

Fast Sweep Method - Neumann hack

```
function phi = fast_sweeping_Neumann_Hack(f,dy,dx,Nx,Ny)
    %Boundary Condition
    phi = inf(Ny,Nx);
    for sweep=1:200
        % The limit starts from column and row 2, and ends at column and row
        % This excludes the boundary points.
        % Updating the boundary points. For doing so i am going to sweep in
        % that too
        for boundary_sweep = 1:2
            % Updating the side AB
            phi(:,Nx) = 0;
            % Updating the side BC
            for i = 2 : Nx-1
                phi(1,i) = min((min(phi(1,i-1),phi(1,i+1)) + f(1,i)*dx),phi(1,i));
            end
            for i = Nx-1 : 2 : -1
                phi(1,i) = min((min(phi(1,i-1),phi(1,i+1)) + f(1,i)*dx),phi(1,i));
            end

            % Updating the side AD
            for i = 2 : Nx-1
                phi(Ny,i) = min((min(phi(Ny,i-1),phi(Ny,i+1)) +
f(Ny,i)*dx),phi(Ny,i));
            end
            for i = Nx-1 : 2 : -1
                phi(Ny,i) = min((min(phi(Ny,i-1),phi(Ny,i+1)) +
f(Ny,i)*dx),phi(Ny,i));
            end

            % Updating the side CD
            for j = 2 : Ny-1
                phi(j,1) = min((min(phi(j-1,1),phi(j+1,1))+f(j,1)*dy),phi(j,1));
            end
            for j = Ny-1 : 2 : -1
                phi(j,1) = min((min(phi(j-1,1),phi(j+1,1))+f(j,1)*dy),phi(j,1));
            end
        end

        % Now we are done with updating the boundary phi values with the
        % neumann type boundary condition at the three sides and one
        % dirichlet type boundary condition at the other side AB
```

```
        % Sweep direction 1
        for i=2:Nx-1
            for j=2:Ny-1
                phi(j,i) = update_point(phi, f, i, j, dx);
            end
        end

        % Sweep direction 2
        for i = Nx-1 : 2 : -1
            for j = 2 : Ny-1
                phi(j,i) = update_point(phi, f, i, j, dx);
            end
        end

        % Sweep direction 3
        for i = 2 : Nx-1
            for j = Ny-1 : 2 : -1
                phi(j,i) = update_point(phi, f, i, j, dx);
            end
        end

        % Sweep direction 4
        for i = Nx-1 : 2 : -1
            for j = Ny-1 : 2 : -1
                phi(j,i) = update_point(phi, f, i, j, dx);
            end
        end
    end
end
```

Fast Sweep Method - Ghost Point Method

```
function phi = ghost_point_approach(f, dx, dy, Nx, Ny)
    % Initialize phi
    phi = inf(Ny+2, Nx+1);  % include ghost points

    % Iterations
    for sweep = 1:200
        % Right boundary: Dirichlet (x = 1)
        phi(:, Nx+1) = 0;

        % Apply Neumann boundaries:
        % Left boundary: mirror value
        phi(:,1) = phi(:,2);

        % Bottom boundary: mirror value
        phi(1,:) = phi(2,:);

        % Top boundary: mirror value
```

```
        phi(Ny+2,:) = phi(Ny+1,:);

        % Sweep direction 1
        for i = 2:Nx
            for j = 2:Ny+1
                phi(j,i) = update_point(phi, f, i, j, dx);
            end
        end

        % Sweep direction 2
        for i = Nx:-1:2
            for j = 2:Ny+1
                phi(j,i) = update_point(phi, f, i, j, dx);
            end
        end

        % Sweep direction 3
        for i = 2:Nx
            for j = Ny+1:-1:2
                phi(j,i) = update_point(phi, f, i, j, dx);
            end
        end

        % Sweep direction 4
        for i = Nx:-1:2
            for j = Ny+1:-1:2
                phi(j,i) = update_point(phi, f, i, j, dx);
            end
        end
    end
end
```

Fast Sweep Algorithm

```
function val = update_point(phi, f, i, j, dx)
    a = min(phi(j,i-1), phi(j,i+1)); % neighbors in x
    b = min(phi(j-1,i), phi(j+1,i)); % neighbors in y

    if abs(a-b) >= f(j,i)*dx
        val = min(a,b) + f(j,i)*dx;
    else
        val = (a+b + sqrt((2*f(j,i)*dx)^2 - (a-b)^2)) / 2;
    end
end
```