Level Set Methods and Their Applications

Homework 4

Student: Faranak Rajabi

Student No.: 6831093

Instructor: Dr. Fredric Gibou

Department of Mechanical Engineering
UC Santa Barbara

In this homework we want to solve the level set function for a front which evolves normal to itself over the time. We need to reinitialize the level set function in order to get the function that has a norm of 1. To do so, we implement the Godunov scheme. We will see the results for two cases, when th magnitude of the value $V_n = 1$ and when it's -1.

1 Problem description

The basic equation will be as follows:

$$\phi_t + a|\nabla\phi| = 0 \tag{1}$$

where a here is 1 and -1. When a > 0 the interface moves in the normal direction, and when a < 0 the interface moves opposite the normal direction. Here, we see the power of signed distance functions. When ϕ_0 is a signed distance function, we can write down the exact solution of the level set equation as $\phi = \phi_0 - at$. On the other hand, when ϕ_0 is not a signed distance function, level set equation needs to be solved numerically by treating it as a Hamilton-Jacobi equation.

2 Results

2.1 Reinitialization

We want to discuss the results as expected in the homework description.

1. The figure 1 is the figure for the initial square function.
2. The figure 2 shows the level set function with one step re-initialization at time = 0.3.
3. The figure 3 shows the function after 25 re-initialization.
4. The figure 4 shows the function after 100 re-initialization, to make sure that we reach the steady-state. As it can bee seen, the level set function gets a better look as it gets more re-initialization steps.

2.2 Moving in normal direction

1. The figure 6 shows the level set function with 100 step re-initialization at time = 0.1.
2. The figure 6 shows the level set function with 100 step re-initialization at time = 0.3. As it can be seen, the geometry in the left plot is getting bigger and evolves in normal direction to itself.

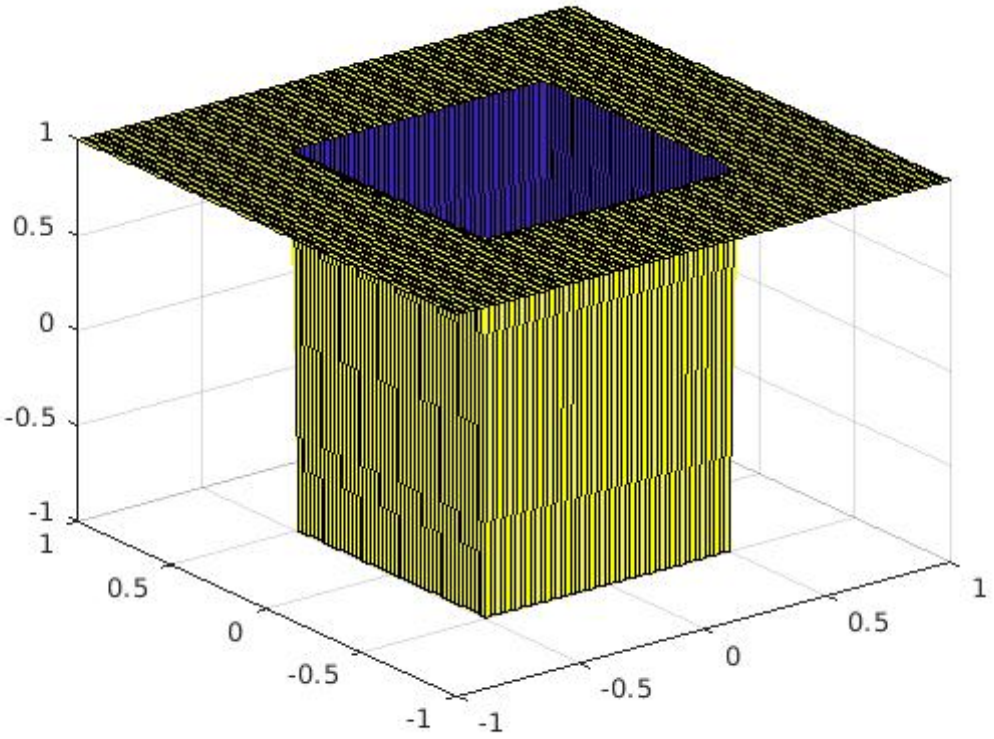

Fig. 1: Initial square function

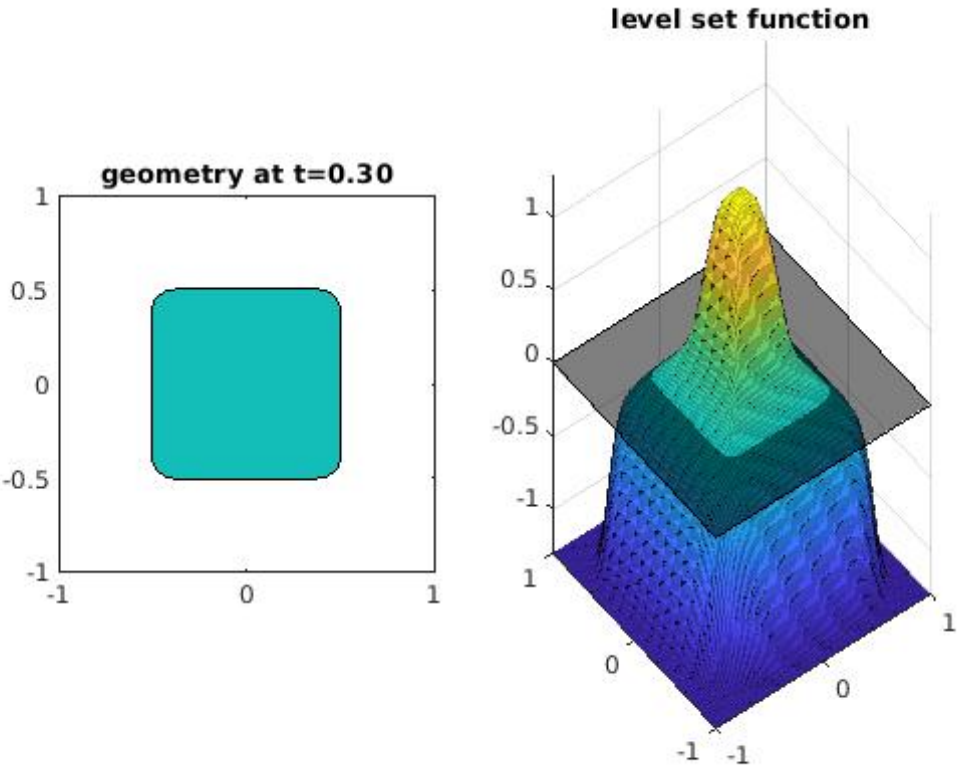

Fig. 2: The level set function with one step re-initialization at time = 0.3.

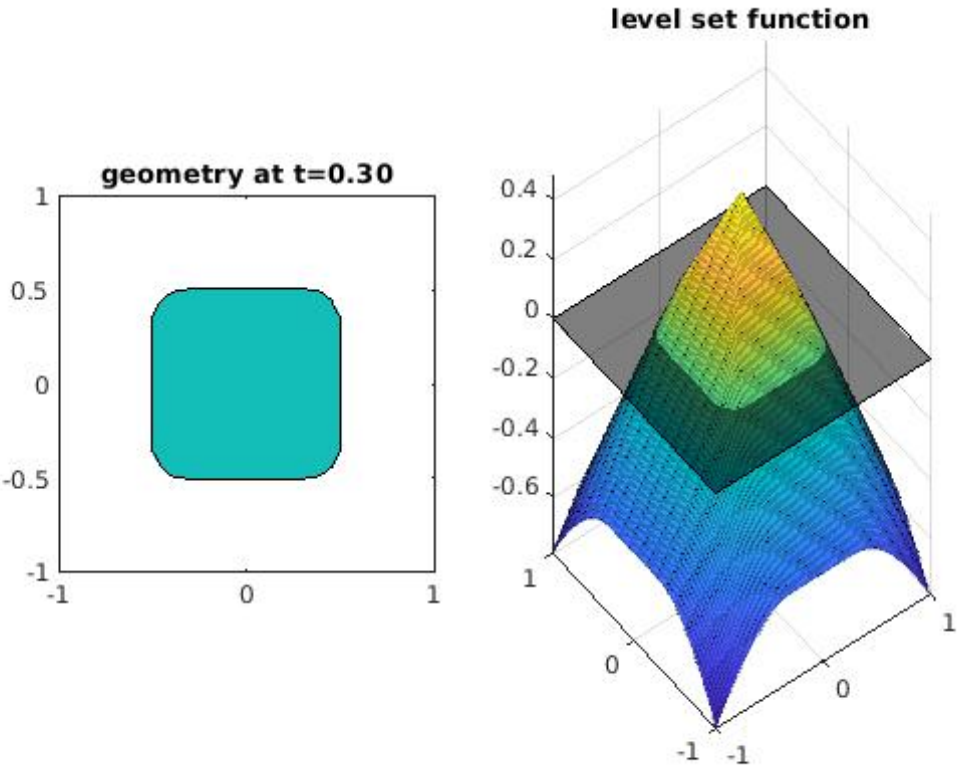

Fig. 3: The level set function with 25 steps re-initialization at time = 0.3.

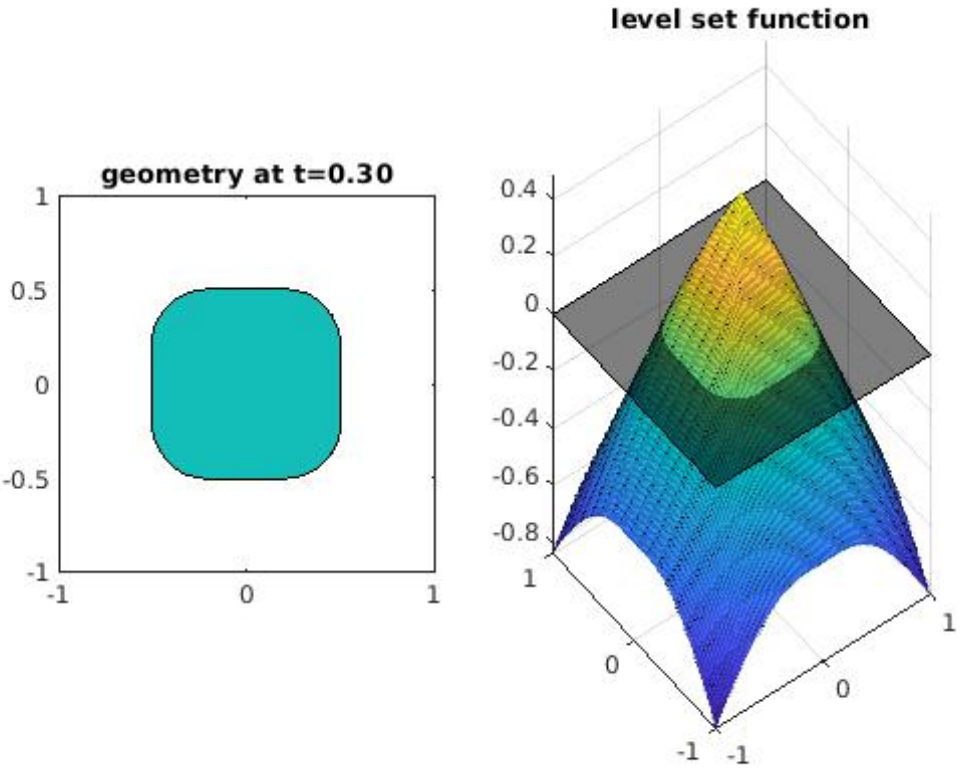

Fig. 4: The level set function with 100 steps re-initialization at time = 0.3 (steady state)

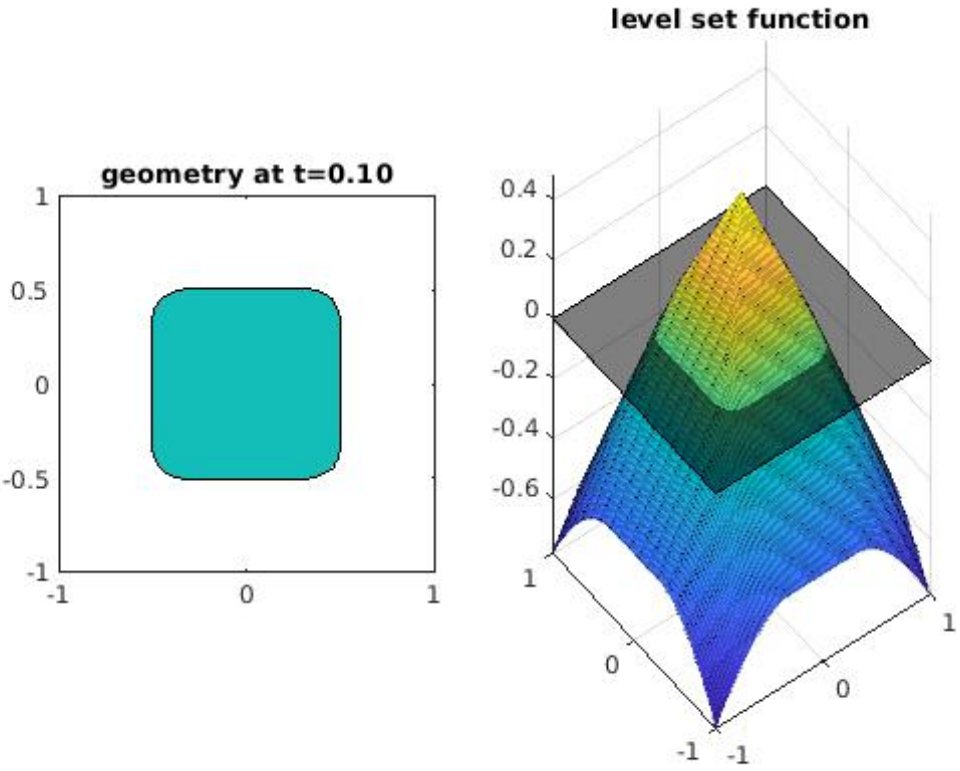

Fig. 5: The level set function with 100 step re-initialization at time = 0.1 for $a = -1$

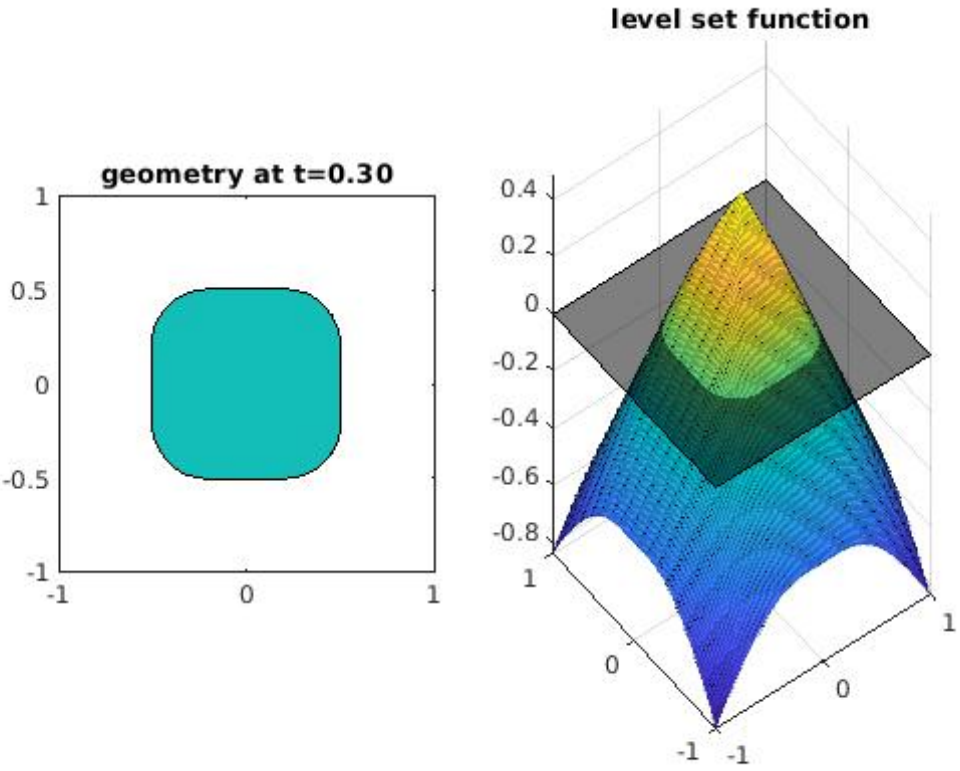

Fig. 6: The level set function with 100 step re-initialization at time = 0.3 for $a = -1$

2.3 Matlab code

```
%% level set methods
function LevelCal(method)
%-------------------------------------------------------------------
nodesNum = 100;        % number of space gridpoints
dt = 8e-3;     % time step
tf = 0.6;     % final time
reinitNum = 100;        % number of reinitialization steps
nsteps = 10;  % number of steps with graphic output
%-------------------------------------------------------------------
if nargin<1  % if the user didnt select the velocity type
    method = 2;
end
nTimeStep = ceil(tf/dt); dt = tf/nTimeStep;
x = linspace(-1,1,nodesNum); y = linspace(-1,1,nodesNum); h = x(2)-x(1);
[X,Y] = meshgrid(x); ax = [min(x) max(x) min(y) max(y)];
%-------------------------------------------------------------------
% initial phi
P = zeros(nodesNum,nodesNum);
for i = 1:nodesNum
    for j = 1:nodesNum
        x(i) = -1 + (i-1) * h;
        y(j) = -1 + (j-1) * h;
        if isInside(x(i),y(j),0,0,1)
            P(i,j) = -1;
        else
            P(i,j) = 1;
        end
    end
end
%surf(x,y,P); hold on; axis ([ax]);
%-------------------------------------------------------------------
for it = 1:nTimeStep
    switch method
        case 1, F = 0.05;              % movement with constant velocity
        case 2, F = curvature(P,h)*15e-3;    % movement under curvature
    end
        P = P-dt*FabsgradP(P,h,F);                          % level set upda
    for ir = 1:reinitNum                                % reinitialization
        P = P-dt*FabsgradP(P,h,P./sqrt(P.^2+(2*h)^2),1);
    end

    if it==1 | floor(nsteps*it/nTimeStep)>floor(nsteps*(it-1)/nTimeStep) %
        clf
```

```
        subplot(1,2,1), contourf(x,x,-P,[0 0],'k-') % this makes the graph
        axis equal, axis(ax)
        title(sprintf('geometry at t=%0.2f',it*dt))
        subplot(1,2,2), surf(x,x,-P,'EdgeAlpha',.2), hold on
        patch([-1 1 1 -1],[-1 -1 1 1],[0 0 0 0],'k','FaceAlpha',.5);
        hold off, axp = [min(0,min(min(-P))) max(0,max(max(-P)))];
        axis([ax axp]), title('level set function')
        drawnow
    end
end

%=========================================================================
function result= isInside(x,y,xc,yc,s)
    if abs((y-yc)) <= s/2 && abs((x-xc)) <= s/2
        result = true;
    else
        result = false;
    end
%-------------------------------------------------------------------------
% computing the gradient
function grad = gradi(P,h)
% computes normal
Px = (P(3:end,:)-P(1:end-2,:))/(2*h); Px = Px([1 1:end end],:);
Py = (P(:,3:end)-P(:,1:end-2))/(2*h); Py = Py(:,[1 1:end end]);
grad = sqrt((Px.^2+Py.^2).^2);
% grad = min(max(F,-1/h),1/h);

%-------------------------------------------------------------------------
% computing the phix and phiy godunov
function dP = FabsgradP(P,h,F,c)
if nargin<4
    c = 0;
if nargin<3
    F = 1;
end
end
DxP = diff(P)/h;   DxmP = DxP([1 1:end],:); DxpP = DxP([1:end end],:);
DyP = diff(P')'/h; DymP = DyP(:,[1 1:end]); DypP = DyP(:,[1:end end]);
Np = sqrt(max(DxmP,0).^2+min(DxpP,0).^2+max(DymP,0).^2+min(DypP,0).^2);
Nm = sqrt(min(DxmP,0).^2+max(DxpP,0).^2+min(DymP,0).^2+max(DypP,0).^2);
dP = max(F,0).*(Np-c)+min(F,0).*(Nm-c);
%-------------------------------------------------------------------------
function F = curvature(P,h)
% computes curvature by central differences
Pxx = diff(P([1 1:end end],:),2)/h^2;
```

```
Pyy = diff(P(:,[1 1:end end])',2)'/h^2;
Px = (P(3:end,:)-P(1:end-2,:))/(2*h); Px = Px([1 1:end end],:);
Py = (P(:,3:end)-P(:,1:end-2))/(2*h); Py = Py(:,[1 1:end end]);
Pxy = (Px(:,3:end)-Px(:,1:end-2))/(2*h); Pxy = Pxy(:,[1 1:end end]);
F = (Pxx.*Py.^2-2*Px.*Py.*Pxy+Pyy.*Px.^2)./(Px.^2+Py.^2).^1.5;
F = min(max(F,-1/h),1/h);
```