MA692 - Final Project; Problem 2

Kameswararao Anupindi
kanupind@purdue.edu
School of Mechanical Engineering, Purdue University

December 10, 2012

1 Write down a detailed Proof for Problem 7.8 on Page 265

2 Cosider the Allen-Cahn equation:

$$u_t - \Delta u + \frac{1}{\epsilon^2}(u^3 - u) = 0, \quad (x, y) \in (-1, 1)^2; \frac{\partial u}{\partial n}|_{\partial\Omega} = 0; \tag{1}$$

with the intial condition $u(x, y, 0) = u_0(x, y)$.

2.1 Problem part 2(ii)

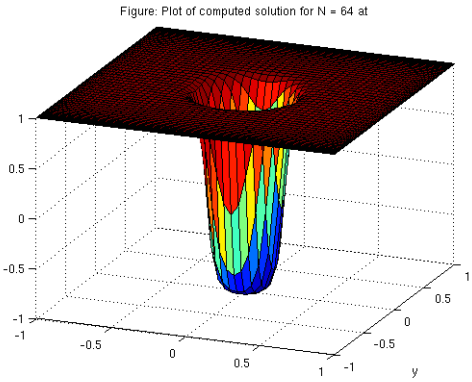
Taking $u_0(x, y) = tanh\left(\frac{\sqrt{(x^2+y^2)}-0.5^2}{\epsilon}\right)$ with $\epsilon = 0.04$. Used a grid of $65X65$ and with homogeneous Neumann boundary conditions with a time step of $\tau = 0.001$. The output and program coded up are shown in the following sections.

2.2 Observations

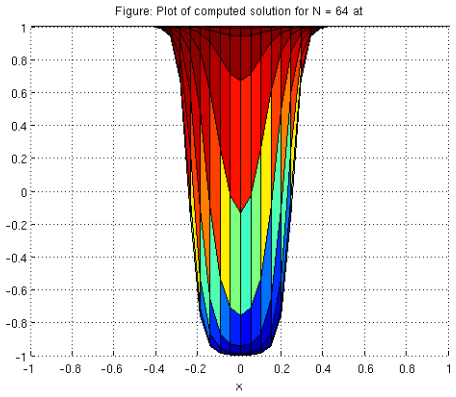
- The evolution of the interface is shown plotted in Figure 1 at six different times
- As we can see from Figure 1 the intial configuration shown in frame (a) becomes thinner in diameter as the time increases till $t = 0.03$. When time is $t = 0.04$ the interface is seen to go above the level set= 1 value by a very small magnitude of 0.005 and it totally disappears after this time instant.
- The contours of level set $u(x, y, t) = 0$ are shown plotted in Figure 2 at 8 different time instants as indicated.
- We can see from Figure 2 that after $t = 0.035$ the interface is not seen in the plot and also as the time increases the diameter of the levelset circle becomes smaller, which we observed in the previous plots in Figure 1.

2.3 Program Listing in Matlab

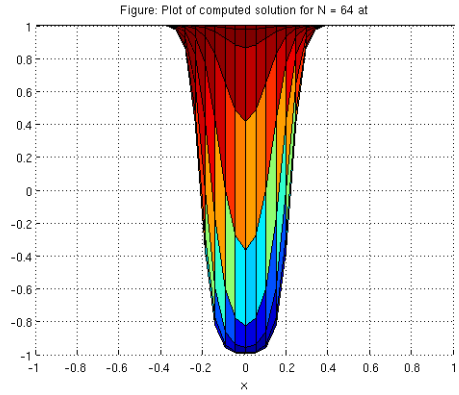
```
% Solution of Allen Cahn equation
% with Neumann boundary conditions
% Final Project (Project #6)
% Problem #2

clear all;
close all;
clc;

% number of collocation points in x, and y directions
nx = 64;
ny = nx;
% time step
tau = 0.001;
% program constants
```



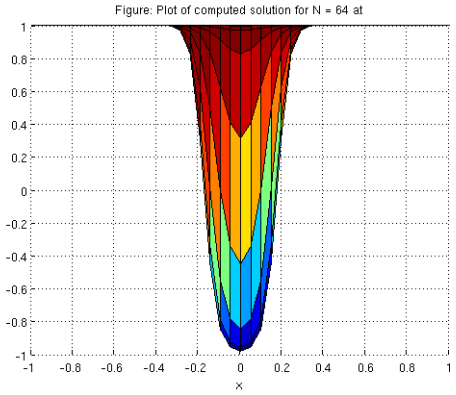
(a) Initial configuration of the interface



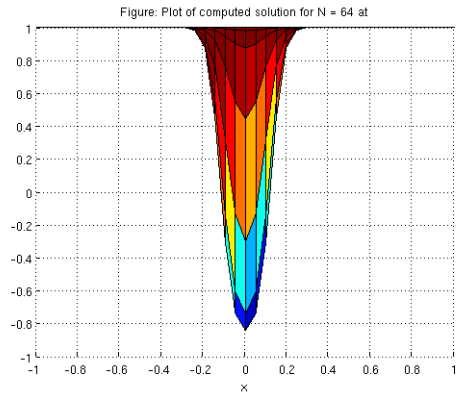
(b) $t = 0.001$



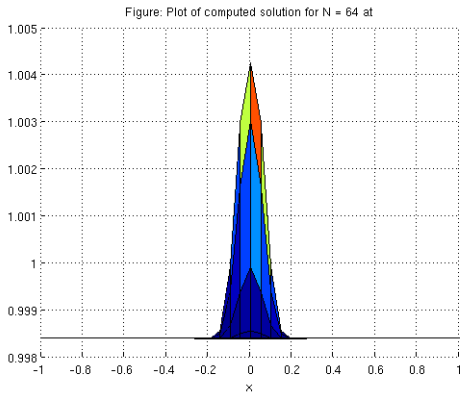
(c) $t = 0.010$



(d) $t = 0.020$



(e) $t = 0.030$



(f) $t = 0.040$

Figure 1: Evolution of the initial interface at indicated times

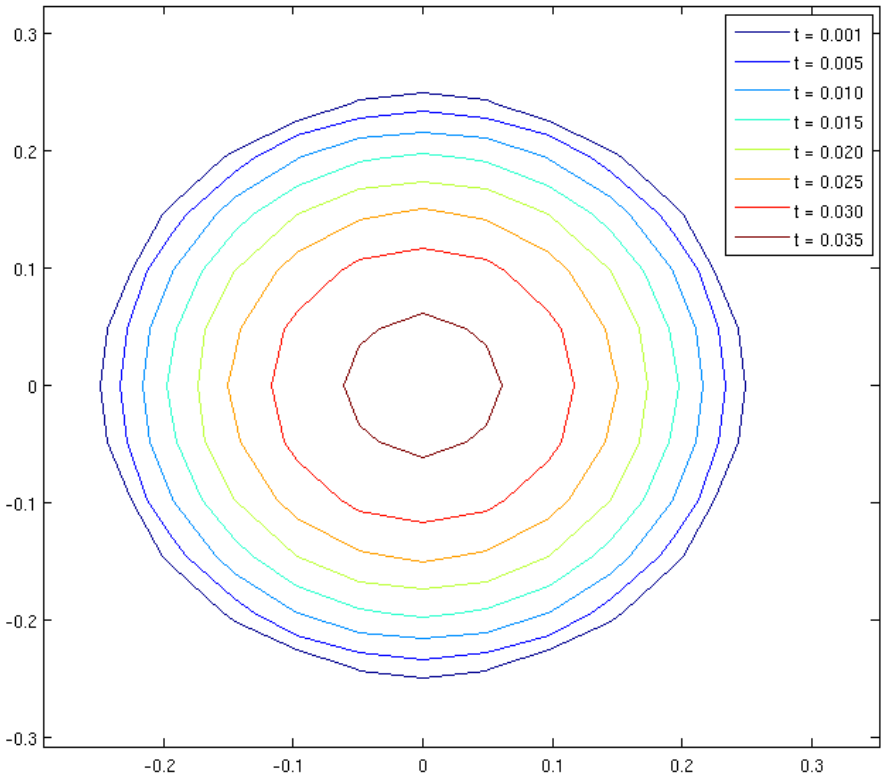


Figure 2: Levelset $u(x, y, t) = 0$ at six different times before the interface vanishes from the domain

```
alpha = 1.0/tau;
%alpha = 1.0;
epsilon = 0.04;

x(1:nx+1, 1:ny+1) = 0.0;
y(1:nx+1, 1:ny+1) = 0.0;

x1d(1:nx+1) = 0.0;
y1d(1:ny+1) = 0.0;

sxjk(1:nx+1, 1:nx+1) = 0.0;
syjk(1:ny+1, 1:ny+1) = 0.0;

wx(1:nx+1) = 0.0;
wy(1:ny+1) = 0.0;

% initial solution
un(1:nx+1, 1:ny+1) = 0.0;

% Mass matrix initialization
M(1:nx-1, 1:ny-1) = 0.0;

% eigen values
% D(1:nx+1, 1:ny+1) = 0.0;
% V(1:nx+1, 1:ny+1) = 0.0;

% rhs function
f(1:nx+1, 1:ny+1) = 0.0;
% interpolated rhs function
If(1:nx+1, 1:ny+1) = 0.0;
Iff(1:nx+1, 1:ny+1) = 0.0;
fk(1:nx-1, 1:ny-1) = 0.0;

%FF(1:nx+1, 1:ny+1) = 0.0;
G(1:nx+1, 1:ny+1) = 0.0;

% 1D rhs and interpolated functions.
f1d(1:nx+1) = 0.0;
If1d(1:nx+1) = 0.0;
fk1d(1:nx-1) = 0.0;

% compute Legendre-Gauss-Lobatto points
[x1d, wx, sxjk] = leinit(nx);
[y1d, wy, syjk] = leinit(ny);

% fill the coefficients ak in the program = bk in the book
for k = 1:ny+1
    i = k;
    ak(k) = -i*(i+1)/((i+2)*(i+3));
end

% compute basis function
% for the homogeneous Neumann boundary conditions
for k = 1:ny-2
    for l = 1:nx+1
        Phi(l, k+1) = (sxjk(l, k+1) + ak(k)*sxjk(l, k+3));
    end
end
```

```
for l = 1:nx+1
    Phi(l, 1) = 1.0/sqrt(2.0);
end

% copy the 1D points into the 2D grid

% S is identity matrix in our case
S = eye(nx-1, ny-1);

S1(1:nx-2, 1:ny-2) = eye(nx-2, ny-2);
M1(1:nx-2, 1:ny-2) = 0.0;
V1(1:nx-2, 1:ny-2) = 0.0;
D1(1:nx-2, 1:ny-2) = 0.0;

for k = 1:ny-2
        ii = k;
        S1(k, k) = -(4*ii+6)*ak(k);
end

% create the two dimensional grid
for j = 1:ny+1
    for i = 1:nx+1
        x(i, j) = x1d(i);
        y(i, j) = y1d(j);
    end
end

% initial solution
epsilon = 0.04;
for j = 1:ny+1
        for i = 1:nx+1
             un(i, j) = tanh((sqrt(x(i, j)^2 + y(i, j)^2) - 0.5^2)/epsilon);
%              for testing purposes
%              un(i, j) = cos(4.0*pi*x(i, j))*cos(4.0*pi*y(i, j));

        end
end

% for testing purposes
% uex = un;

% figure(1);
% surf(x,y,un);
% isovalues=[0.00001, 0.00001];
% contour3(x,y,un,isovalues);
%stop;

% fill Integral L_k^2 values
for k = 1:nx+1
     i = k;
    gk(k) = 2.0/(2.0*i+1);
end

% Compute the mass matrix
for j = 1:ny-2
    for i = 1:nx-2
```

```
        if (i == j)
            M1(i, j) = (gk(i) + ak(i)^2*gk(i+2));
        end

        if (i == j+2)
            M1(i, j) = +ak(j)*gk(j+2);
        end

        if (i == j-2)
            M1(i, j) = +ak(i)*gk(i+2);
        end

    end
end

% Augment the Mass matrix
M(1, 1) = 1.0;
for k = 2:ny-1
    M(1, k) = 0.0;
    M(k, 1) = 0.0;
end

% Compute the eigen values and eigen vectors of the mass matrix
% [V, D] = eig(M, S);
% Compute eigen values and eigen vectors for the smaller matrices
[V1, D1] = eig(M1, S1);

% Augment the smaller matrices to bigger ones
for j = 1:ny-2
    for i = 1:nx-2
        ii = i+1;
        jj = j+1;
        V(ii, jj) = V1(i, j);
        D(ii, jj) = D1(i, j);
        M(ii, jj) = M1(i, j);
        S(ii, jj) = S1(i, j);
    end
end

D(1, 1) = 1.0;
V(1, 1) = 1.0;
S(1, 1) = 1.0;

E = V;

figure(3);
isovalues=[0.00001, 0.00001];
contour(x,y,un,isovalues);
hold all

% Perform 60 time steps before which the interface disappears
for it = 1:60

    %fill the rhs function
        for j = 1:ny+1
                for i = 1:nx+1
                     f(i, j) = un(i, j)/tau - (1.0/epsilon^2)*(un(i, j)^3 - un(i, j));
%                      for testing purposes
```

```
%                       f(i, j) = (alpha + 32.0*pi*pi)*
%                                 cos(4.0*pi*x(i, j))*cos(4.0*pi*y(i, j));
                end
        end

% forward transform
    for j = 1:ny+1

        for i = 1:nx+1
            f1d(i) = f(i, j);
        end

        If1d = letrfm1(nx, f1d', x1d, sxjk, 0);

        for i = 1:nx+1
            Iff(i, j) = If1d(i);
        end

    end

    for i = 1:nx+1

        for j = 1:ny+1
            f1d(j) = Iff(i, j);
        end

        If1d = letrfm1(ny, f1d', y1d, syjk, 0);

        for j = 1:ny+1
            If(i, j) = If1d(j);
        end

    end

%     fill fbar
    FF(1:nx-1, 1:ny-1) = 0.0;

    for jj = 2:ny-1
        for ii = 2:nx-1
            i = ii-1;
            j = jj-1;
                FF(ii, jj) = gk(i)*gk(j)*If(ii, jj)
                           + ak(i)*ak(j)*gk(i+2)*gk(j+2)*If(ii+2, jj+2)
                           + ak(j)*If(ii, jj+2)*gk(i)*gk(j+2)
                           + ak(i)*If(ii+2, jj)*gk(i+2)*gk(j);

        end
    end

   FF(1, 1) = If(1, 1)*2;

    i = 2;
    ii = 1;
    for jj = 2:ny-1
              j = jj-1;
              FF(ii, jj) =  If(ii, jj)*gk(j)*sqrt(2.0)
                         + ak(j)*If(ii, jj+2)*gk(j+2)*sqrt(2.0);
    end
```

```
    j = 2;
    jj = 1;
    for ii = 2:nx-1
        i = ii-1;
        FF(ii, jj) =  If(ii, jj)*gk(i)*sqrt(2.0)
                    + ak(i)*If(ii+2, jj)*gk(i+2)*sqrt(2.0);
    end

    V = S*E;
    H = inv(V)*FF*inv(transpose(V));
    % using full diagonalization
    %   H = V\FF/V';
    % uncomment for partial diagonalization
    %   G = V\FF;

    for i = 1:nx-1
        for j = 1:ny-1

    %         using full diagonalization
            w(i, j) = H(i, j)/(alpha*D(i, i)*D(j, j) + D(i, i) + D(j, j));
    %         for partial diagonalization uncomment line below cand comment
    %         the one above
    %         w(i, j) = G(i, j)/((alpha*D(i, i)+1)*M(i, j)+D(i, i));

        end
    end


    % use full diagonalization
    U = E*w*E';
    % uncomment below for partial diagonalization
    %U = V*w;

    % Compute U from basis functions directly
    for j = 1:ny+1
        for i = 1:nx+1

            Us(i, j) = 0.0;
            for k = 1:ny-1
                for l = 1:nx-1

                    Us(i, j) = Us(i, j) + U(l, k)*Phi(i, l)*Phi(j, k);

                end
            end

        end
    end


   for j = 1:ny+1
       for i = 1:nx+1
           un(i, j) = Us(i, j);
       end
   end

%    print progress as the timesteps are advanced
```

```
it

% if (it == 1 || mod(it, 10) == 0)
%     figure;
%     surf(x,y,un);
%     title('Figure: Plot of computed solution for N = 64 at');
%     xlabel('x');
%     ylabel('y');
%     view([0 0]);
% %  zlim([-1 1]);
% end

% plot the u(x, y, t) = 0 contour to the figure
if (it == 1 || mod(it, 10) == 0)

figure(3);
isovalues=[0.00001+it*0.001, 0.00001+it*0.001];
contour(x,y,un,isovalues);
hold all

end


end %iteration loop

% for testing purposes
% error = norm(uex - Us, 2)

% plot the computed solution and plot
% figure(3)
% [X,Y]=meshgrid(x1d,y1d);
% mesh(X, Y, un);
% title('Figure1: Plot of computed solution for N = 64');
% xlabel('x');
% ylabel('y');
% axis tight;

% figure(3)
% surf(x, y, uexact);
% title('Figure2: Plot of exact solution for N = 64');
% xlabel('x');
% ylabel('y');
```