MM3110 - Expt 5 - Phase Field Modelling

1 Introduction

A **Phase field model** is a mathematical model for solving interfacial problems. In this report this method has been implemented to understand the solidification dynamics of a single component melt growth. Phase field models with certain form of anisotropy already exist and they are capable of simulating various dendritic morphologies [1]. Here we try to reproduce one of those models.

2 Mathematical model

To understand the evolution of the solid phase two partial differential equations have to be solved simultaneously.

$$\tau\frac{\partial P}{\partial t} = -\frac{\partial}{\partial x}\left(\epsilon\epsilon'\frac{\partial P}{\partial y}\right) + \frac{\partial}{\partial y}\left(\epsilon\epsilon'\frac{\partial P}{\partial x}\right) + \vec{\nabla}\cdot\left(\epsilon^2\vec{\nabla}P\right) + P(1-P)(P-0.5+m) \tag{1}$$

$$\frac{\partial T}{\partial t} = \nabla^2 T + \frac{\partial P}{\partial t} \tag{2}$$

Equation 2 is non-dimensionalised such that the melting point (T_{eq}) is taken as `1` and initially the whole of the domain is under-cooled with `T = 0`. A noise term is also added to Equation 1 to determine the stability of the interface.

The various terms and constants used in the calculation are given below.

Table 1 : Function handles used

Noise term : a$\times P \times (1-P) \times \chi$,
Anisotropy term : $\sigma(\theta) = 1 + \delta cos(j[\theta - \theta_0])$
$m(T) = \frac{\alpha}{\pi} tan^{-1}\left(\gamma[T_{eq} - T]\right)$
$\epsilon = \bar{\epsilon} \times \sigma(\theta)$

Table 2 : Values of variables used

$\theta_0 = 0.0;$
$j = 4;$
$\alpha = 0.9;$
$\gamma = 10.0;$
$K = 2.0;$
$a = 0.01;$
$\tau = 0.0003;$
$\bar{\epsilon} = 0.01;$
$\Delta t = 0.0002;$

3 Code

A code is written in MATLAB . The code is given below

```
clear all
clc
%%Initialization of variables and Domain
% Domain for dendritic soldiifcation
xlength = 9.0;
halfx = xlength/2.0;
ylength = 9.0;
```

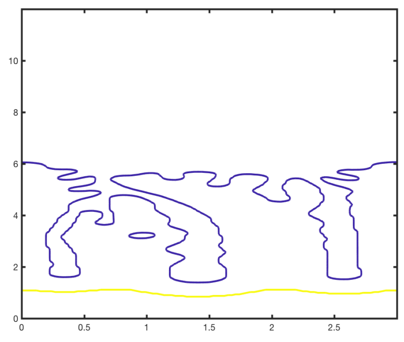
```
halfy = ylength/2.0;
Nx = 301;
Ny = 301;

% For rectangular domain
% xlength = 12.0;
% halfx = xlength/2.0;
% ylength = 3.0;
% halfy = ylength/2.0;
% Nx = 401;
% Ny = 101;
delx = xlength/(Nx-1);
dely = ylength/(Ny-1);
x = zeros(Nx,1);
y = zeros(Ny,1);
for i = 2:Nx
x(i) = x(i-1) + delx;
end
for i = 2:Ny
y(i) = y(i-1) + dely;
end
[X,Y] = meshgrid(x,y);
Teq = 1.0;
temp_old = zeros(Nx,Ny);
temp_new = zeros(Nx,Ny);
p_new = zeros(Nx,Ny);
p_old = zeros(Nx,Ny);
angle = zeros(Nx,Ny);
delpy = zeros(Nx,Ny);
delpx = zeros(Nx,Ny);
% Values for variables
theta0 = 0.0;
j = 4;
delta = 0.05;
alpha = 0.9;
gamma = 10.0;
K = 2.0;
a=0.01;
tau = 0.0003;
epsilonbar = 0.01;
delt = 0.0002;
temp_old(:,:) = 0;
% Function handles
mfactor = @(temp) (alpha/3.1415)*atan(gamma*(Teq - temp));
sigma = @(theta) 1 + delta*cos(j*(theta - theta0));
epsilon = @(theta) epsilonbar*sigma(theta);
epsilondash = @(theta) epsilonbar*delta*j*(-sin(j*(theta-theta0)));
value = [1,0.5];
%initial profile
% For directional soldiification
%p_old(1:20,:) = 1.0;
% For dendritic soldiification
for m = 2:Nx-1
for n = 2:Ny-1
    dist = (x(m) - halfx)^2 + (y(n)- halfy)^2;
    dist = sqrt(dist);
    if(dist <= 0.125)
        p_old(m,n) = 1.0;
    end
end
end
mesh(p_old);
view(2);
hold off
ax = gca ;
set(ax, 'linewidth',2.0);
```

```matlab
pause(1);


%% Calculation of the evolution of P and T
for i = 1:3500
    str = ['i = ',num2str(i)];
    disp(str);
    for m = 2:Nx-1
        for n = 2:Ny-1
           delpy(m,n) = (p_old(m,n+1) - p_old(m,n-1))/(2*dely);
           delpx(m,n) = (p_old(m+1,n) - p_old(m-1,n))/(2*delx);
           if(delpy(m,n) == 0 )
               angle(m,n) = 0;
           else
               angle(m,n) = atan(delpy(m,n)/delpx(m,n));
           end
        end
    end

        for m = 2:Nx-1
        for n = 2:Ny-1
           term1 = ((epsilon(angle(m+1,n)) - epsilon(angle(m-1,n)))*(...
           epsilondash(angle(m,n))*delpy(m,n))/(2*delx));
           term2 = ((epsilondash(angle(m+1,n)) - epsilondash(angle(m-1,n)))*(...
           epsilon(angle(m,n))*delpy(m,n))/(2*delx));
           term3 = ((delpy(m+1,n) - delpy(m-1,n))*epsilon(angle(m,n))*(...
           epsilondash(angle(m,n)))/(2*delx));

           term4 = ((epsilon(angle(m,n+1)) - epsilon(angle(m,n-1)))*(...
           epsilondash(angle(m,n))*delpx(m,n))/(2*dely));
           term5 = ((epsilondash(angle(m,n+1)) - epsilondash(angle(m,n-1)))*(...
           epsilon(angle(m,n))*delpx(m,n))/(2*dely));
           term6 = ((delpx(m,n+1) - delpx(m,n-1))*epsilon(angle(m,n))*(...
           epsilondash(angle(m,n)))/(2*dely));

           term7 = (2*epsilon(angle(m,n))*delpx(m,n)*(epsilon(angle(m+1,n))  -(...
           epsilon(angle(m-1,n)))/(2*delx)));
           term8 = epsilon(angle(m,n))*epsilon(angle(m,n))*(...
           (p_old(m+1,n)+p_old(m-1,n)- 2*p_old(m,n) )/(delx*delx));

           term9 = (2*epsilon(angle(m,n))*delpy(m,n)*(epsilon(angle(m,n+1))  -(...
           epsilon(angle(m,n-1)))/(2*dely)));
           term10 = epsilon(angle(m,n))*epsilon(angle(m,n))*(...
           (p_old(m,n+1)+p_old(m,n-1)- 2*p_old(m,n) )/(dely*dely));
           term11 = p_old(m,n)*(1-p_old(m,n))*(...
           (p_old(m,n) - 0.5 + mfactor(temp_old(m,n))));
           if((p_old(m,n)>=0.5)&&(p_old(m,n)<1))
           term12 = a*p_old(m,n)*(1-p_old(m,n))*(rand() - 0.5);
           else
               term12 = 0.0;
           end
           p_new(m,n) = p_old(m,n) + delt*(-term1 - term2 - term3 + (...
            term4 + term5 + term6 + term7 + term8 + term9 + term10 + (...
            term11 + term12 )/tau)) ;

           temp_new(m,n) = temp_old(m,n) + delt*(temp_old(m+1,n) +(...
           temp_old(m-1,n) -2*temp_old(m,n))/(delx*delx));
           temp_new(m,n) = temp_new(m,n) + delt*(temp_old(m,n+1) +(...
           temp_old(m,n-1) -2*temp_old(m,n))/(dely*dely));
           temp_new(m,n) = temp_new(m,n) + K*(p_new(m,n) - p_old(m,n));
        end
    end

    %Adiabiatic BC for P and T
    temp_new(1,:) = temp_new(2,:);
    temp_new(Nx,:) = temp_new(Nx-1,:);
    temp_new(:,1) = temp_new(:,2);
    temp_new(:,Ny) = temp_new(:,Ny-1);
    p_new(1,:) = p_new(2,:);
    p_new(Nx,:) = p_new(Nx-1,:);
    p_new(:,1) = p_new(:,2);
    p_new(:,Ny) = p_new(:,Ny-1);

    %Updating the values
    p_old = p_new;
    temp_old = temp_new;

    % Plotting for every 100 time steps
    if(mod(i,100) == 0 )
    contour(p_new,value,'linewidth',2.0)
    view(2);
    hold off
    ax = gca ;
    str1 = ['Contour',num2str(i)];
    set(ax, 'linewidth',2.0);
    print(str1,'-dpng');
    pause(1);
    end

end
```

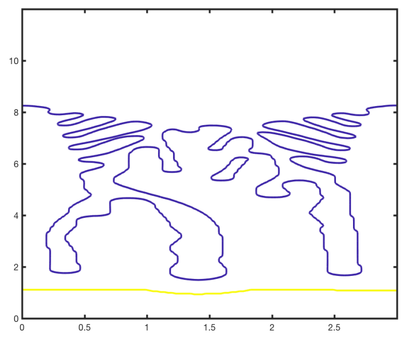
4 Results

4.1 Directional Isotropic

In this case the code is run in a rectangular domain. No flux boundary conditions are applied.σ is taken as 1 and independent of θ . The plots obtained are given Figure 1.



(a) t = 0.5 s



(b) t = 0.7 s

Figure 1: Directional Isotropic Solidification

4.2 Directional An-isotropic

In this case the code is run in a rectangular domain. No flux boundary conditions are applied.σ is taken as a function of θ . The plot obtained is given in Figure 2.

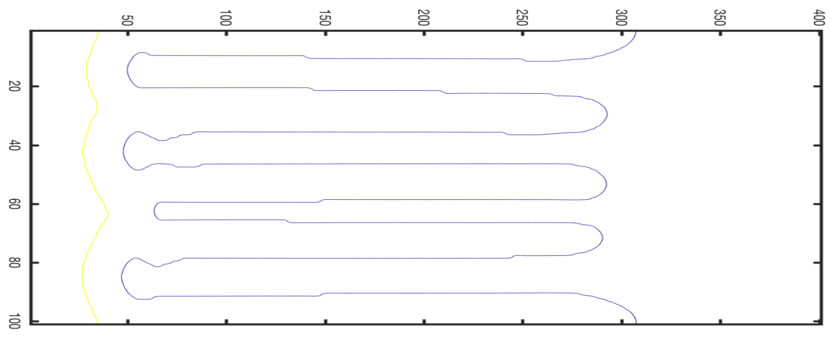
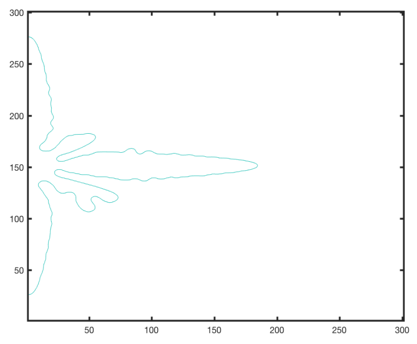


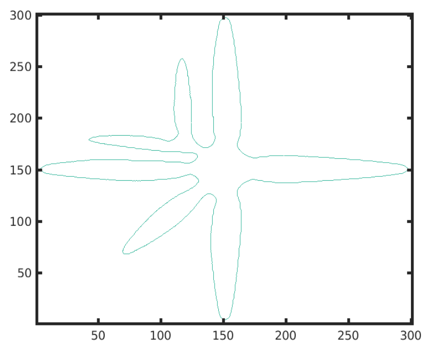
Figure 2: Directional Anisotropic Solidification at t = 0.7s

4.3 Dendritic Growth

In this case the code is run in a square domain. Initially `P` is made solid in one semi circular region on one side whose growth can be seen in Figure 3a and circular region in the center whose growth can be seen in Figure 3b. No flux boundary conditions are applied.σ is taken as a function of θ . The plots obtained are given in Figure 3.



(a) The solid nucleus in semi circular and placed on one side



(b) The solid nucleus in circular and placed in center

Figure 3: Dendritic Solidification

References

[1] Ryo Kobayashi. "Modeling and numerical simulations of dendritic crystal growth". In: *Physica D: Nonlinear Phenomena* 63.3 (1993), pp. 410–423. ISSN: 0167-2789. DOI: https://doi.org/10.1016/0167-2789(93)90120-P. URL: http://www.sciencedirect.com/science/article/pii/016727899390120P.