Phase Field Modeling of Eutectic Systems: Enhanced Efficiency of Non Equilibrium State Simulation

Punn Augsornworawat

Department of Mining and Materials Engineering, McGill University, Montreal Quebec, Canada
Contact Email: punn.augsornworawat@mail.mcgill.ca

Abstract

Phase formation plays an important role in determining material properties to suit the desired application. Both empirical and mechanistic models are used to simulate phase transformation in a Phase Field Model. Nevertheless, pre-existing simulation tools lack efficiency in computation. To overcome such limitation, the simulation package has been developed to a different programing platform and reformatted from MATLAB to C language. Such transition showed several orders of reduced simulation time and improvements in data processing efficiency. The two main models in Phase field modeling, conservative and non-conservative models, have been implemented to successfully simulate eutectic solidification of materials.

Keywords: Phase Field Model, Non Equilibrium Systems, Simulation, Eutectic, Solidification, Free Energy

Outline

Introduction

Phase transformations and thermodynamic properties of many materials have been investigated quite profoundly over the past few decades and have significant applicable features toward material synthesis and design. The ability to predict the microstructural outcomes plays a fundamental step in controlling some of the materials' key important features. (i.e. Mechanical, electrical, magnetic, chemical etc.) Despite well-established experimented data, most material properties may be calculated, using mechanical models and fundamental properties, usually for equilibrium state conditions. Very few material mixtures or compounds have been analyzed at non-equilibrium due to the complexity and accountability of greater variables in computations. Since most encountered systems are at non-equilibrium, never reaching its stable phase (metastable phases), investigation of these states are also important, especially in phase transformation. Tracking non-equilibrium systems by experiments require numerous tests and considerations of non-equilibrium parameters, which include space, time, diffusion, lattices, structures, stability, etc. Non-equilibrium systems are therefore suggested to be modeled by computations rather than experimental. [1-4] With modern computing power, sophisticated computations and simulations have been carried to compensate these experimental problems.

The keystone behind phase and microstructural modeling of materials relies on the mechanistic and empirical model, which involves the scientific principles of atomic properties and experimental observations. The empirical portion of the simulation serves to simplify computations, disregarding complex parameters and variables that are difficult or impossible to determine. Generally, these values are in a form of constants and determined by minimizing the solution residuals to fit experimental data. Parameters based on scientific principles are determined

theoretically or experimentally. They are usually in a form of mathematical formulas involving constants that relate to the material properties. This paper will deal with both aspects quite rigorously in phase field modeling.

Phase Field Simulations of Non-equilibrium States

The phase field model was developed to monitor the microstructural pattern of two correlated distinctive phases at both steady and unsteady state conditions. The two phases may be termed based on the properties or state that the user wishes to model. These states may include concentration profiles, phase distribution, electrical properties, magnetic properties, density states, etc. (as long as the two distinctions are related by scientific phenomenon such as concentration and solid/liquid phases.) [1-2] However, phase field modeling usually deals with the distribution in the states of matter, elemental components and thermodynamic properties of materials.

Computing the phase pattern under certain conditions require the use of a phase diagram for certain metals or compounds. Before performing any time dependent microstructural simulations, the phase diagram must be modeled and optimized correctly to match experimental results. Previous reports have developed ways to plot phase diagrams using the following formulas for liquid and solid respectively. [1]

$$c_L \approx \frac{\mu_0}{w + 2\beta(r/a)^{1/2} + \alpha\beta\Delta T/r}. \tag{1}$$

$$c_S \approx \pm\left[\frac{2\beta(r/a)^{1/2} - w - \alpha\beta\Delta T/r}{b - \beta^2/r}\right]^{1/2} + \frac{\mu_0}{2[2\beta(r/a)^{1/2} - w - \alpha\beta\Delta T/r]}, \tag{2}$$

The parameters ΔT and μ represents the change in temperature from the eutectic or transitioning point and the chemical potential of the compound respectively. The remaining values are assumed constants based on the free energy distribution and surface properties, which are optimally defined to accurately match the phase diagram. [1-2] An example, generated on MATLAB is shown in figure 1.

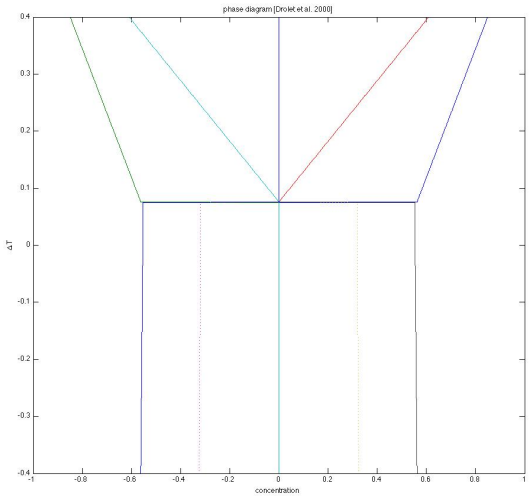


Figure 1 Phase diagram of binary phase systems where x and y axis are c (normalized concentrations) and ΔT (normalized temperature change) respectively. The parameters (r, a, b, w, α, β) were pre adjusted to (1,1,1,0,0.15,0.15). [1]

The free energy equation plays a remarkable role in determining the stability of certain phases for eutectic growth modeling. Based on the phase diagram, the free energy distribution is determined from this equation. [5]

$$\mathcal{F}\{c\} = \int d\vec{r}\left(f(c) + g(\vec{\nabla}c)\right). \tag{3}$$

The capitalized F is dependent on the two minor energy functions, f and g. The minor f is the bulk free energy function of the phase with dependence to the ordered parameter, i.e. concentration of the element or compound. The g is a function relating to the surface energy between the interfaces of the two determined phases and is composed of gradient terms that may be up to several orders. Generally however, only the first term is considered as is sufficient for estimating the overall free energy distribution. The equations for f and g functions are given below. [1,5]

$$f(c) = \frac{r}{2}(\delta c)^2 + \frac{u}{4}(\delta c)^4, \tag{4}$$

$$g = \frac{K}{2}|\vec{\nabla}c|^2 + B|\nabla^2 c|^2 + \cdots \tag{5}$$

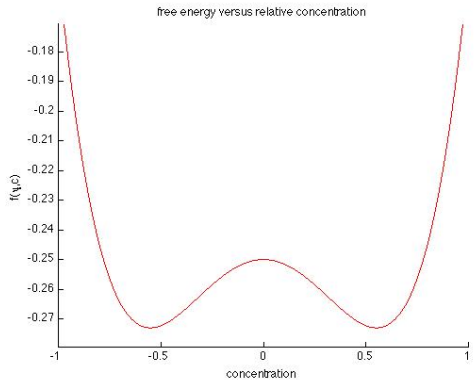
Plugging back into the F function, these two equations simplify to the following form of the free energy equation. The r and u values are called phenomenological constants which are determined from the parameters embed in the phase diagram. The K parameter consists of the Boltzmann constant and temperature of the system. It is also worth

noting that the ordered paratmeter, c, is a function of time and position, c(x,t).

$$\mathcal{F}\{c\} = \int d\vec{r}\left(\frac{r}{2}(\delta c)^2 + \frac{u}{4}(\delta c)^4 + \frac{K}{2}|\vec{\nabla} c|^2\right). \tag{6}$$

This final form of the free energy determines the stability of certain phases and plays an important role for the phase field simulation. [1] Depending on the preset initial conditions, the free energy equation will be minimized to the lowest possible value when approaching equilibrium. A free energy diagram is shown based on the previously generated phase diagram.

a)



b)

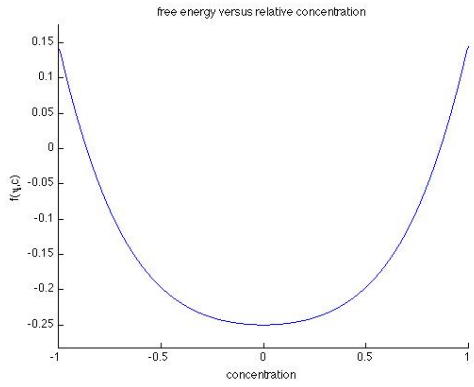


Figure 2 Free energy diagram at ΔT = 0 for a liquid and solid phase respectively.

Single-phase stability, figure 2(a), is relatively simple to model, as there is only one minimum. In cases where multiple phases exist (ie. Metastable phases) the simulation can become a bit more complex and accounts other variables in minimizing free energy solutions. In the case of figure 2(b), there are two solutions. At equilibrium, the point on the free energy curve will approach at the minimum. As for non-equilibrium states, the point on the energy curve is dependent on time, temperature and mobility of the atomic particles in the systems. Mobility is the main factor that defines the rate of diffusion and potential change in the system of a phase transformation. The mobility consists of a chemical potential gradient, which may or may not be dependent on the ordered parameter, in this case is the concentration surrounding the local point. This arises to the concept of Model A and Model B where the ordered parameter or concentration profile is non-conserved and conserved respectively. The free energy equation and mobility are incorporated and simplified to the following equation known as the Langevin or dynamics equation. [5-7]

$$\frac{\partial \phi}{\partial t} = -M\left(-\nabla^2\right)^a \frac{\delta F}{\delta \phi}, \tag{7}$$

For generalization, the concentration variable, c, has been replaced by a psi symbol, ϕ. The "a" value determines whether the ordered parameter is conserved or not conserved. (a=0 for non-conserved model or Model A while b = 1 is for conserved model or Model B)

Classification of the Dynamics Equation

Model A

When the double gradient term, ($-\nabla^2$), is taken out of the dynamics equation, the ordered parameter (ie. Concentration or state of matter) is no longer conserved. This means that it does not account for the change in concentration around its local point. The non-conserved form of the dynamics equation is therefore simple in terms of dynamic calculation and can be applied to simple case scenarios such as the phase solidification. [7]

Model B

For conserved model where a = 1, which includes the gradient term, the surrounding values of the ordered parameter is taken into account. Model B defines ϕ as concentration. Note however, that the conservation field in this equation is only applied "locally" and does not conserve the overall concentration of the system. The conservation from the potential gradient is described with the ($-\nabla^2$) factor and δϕ, which accounts the local surrounding concentration. In this formulation, several complicated transitions may be simulated. This may include the transformation between stable and metastable phase, which relies on the concentration mobility. As mentioned, the mobility affects the diffusion rate for the transfer of the local ordered parameter.

When Model B was introduced, an additional term was also accounted. The new formula for the dynamics equation, seen below, consists of the mobility, gradient, free energy functions and the new noise term. [8] This is known as the Allen - Cahn equation. [6]

$$\frac{\partial \phi}{\partial t} = M\nabla^2\left(\frac{\partial f}{\partial \phi} - K\nabla^2\phi\right) + \eta, \tag{8}$$

The noise term, η, is defined by the thermal fluctuation of the atomic particles in the system. The degree of fluctuation is dependent on the temperature as vibrations increases with energy. The noise term is not related to the concentration, space or time. Since the rate of vibration is much greater compared to the rate of diffusion in phase transformation, the noise term is therefore evaluated on a random generated number. Because of this, the noise term should not contain any bias and must satisfy the following condition. [5, 8]

$$\langle \eta \rangle = 0. \tag{9}$$

The sum of all the noise term in the system must approach 0 to maintain this un-biased condition.

Model C

Drolet et al. introduced Model C by using Model A and Model B and correlating one parameter with the other. [1] In this case, there must be two equations, one conserved and the other non-conserved, to evaluate the solutions. Previously discussed, Model A can evaluate solidification by setting ϕ, the non-conserved ordered parameter, as solid phase or liquid phase. As for model B, ϕ was assigned to concentration as a conserved ordered parameter. Because both ϕ are being considered in the model, the free energy equation must also account both parameters. Going back to equation (3), the bulk free energy equation, f, is a function of both the solid/liquid state and the concentration. While the surface free energy, g, consist of energy terms base on the two solutions. The total free energy equation is given below.

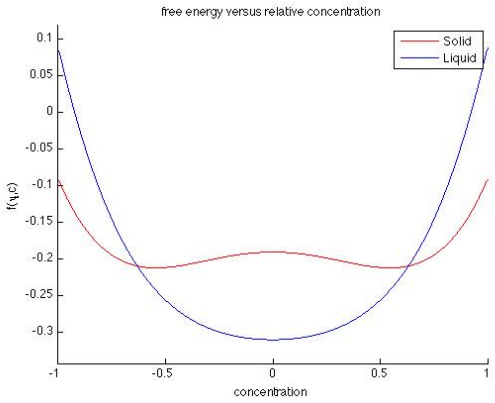
$$\mathcal{F}\{c,\psi\} = \int d\vec{x}\left[f(c,\psi) + \frac{K_\psi}{2}|\vec{\nabla}\psi|^2 + \frac{K_c}{2}|\vec{\nabla}c|^2 \right], \tag{10}$$

where the linearized bulk free energy function is as follow.

$$f = -\frac{r}{2}\psi^2 + \frac{a}{4}\psi^4 + (\alpha\Delta T - \beta c^2)\psi + \frac{w}{2}c^2 + \frac{b}{4}c^4. \tag{11}$$

Since there are two ϕ definitions, the non-conserve parameter is assigned with the ψ symbol (ϕ for state of matter, solid/liquid), while the conserve parameter is assigned as c (ϕ for concentration). Again, the free energy approaches the minimum as the system approaches equilibrium. The following figure illustrates the free energy curve that accounts both parameters.

a)



b)

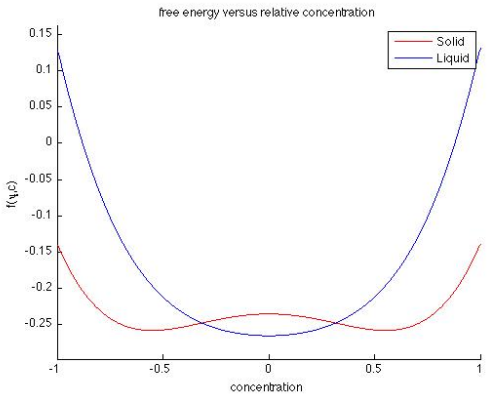


Figure 3 Free energy diagrams of the liquid and solid-state phases. The x and y axis have been normalized for concentration and free energy respectively. (a) is the free energy curve at ΔT = 0.4 on the phase diagram. (Comprise of a stable liquid phase at -0.6 < c < 0.6) (b) is a free energy curve at ΔT = 0.1 (lower temperature), where the solidification is enhanced at greater concentration ranges and the liquid stability range is reduced.

To complete the analysis of the concentration and solid/liquid state, the free energy equation is plugged into the Cahn Hilliard Cook equations. Note that the first equation evaluates the solid/liquid state (a=0, non-conserve) while the second evaluates the concentration (a=1, conserve). [1]

$$\frac{\partial \psi}{\partial t} = -M_\psi \frac{\delta \mathcal{F}}{\delta \psi} + \eta_\psi \tag{12}$$

$$\frac{\partial c}{\partial t} = M_c \nabla^2 \frac{\delta \mathcal{F}}{\delta c} + \eta_c \tag{13}$$

Linearizing the Allen – Cahn – Hilliard – Cook for Simulation

To apply the Cahn-Hilliard-Cook equations in model C, the solutions must be linearized so that it can easily be

coded as a program. Drolet et al. developed the estimated linearized form of ψ and c solutions. [1]

$$\psi_{i,j}(n+1) = \psi_{i,j}(n) + M_{\psi}\Delta t\left[r\psi_{i,j}(n) - a\psi_{i,j}^{3}(n) + (\beta c_{i,j}^{2}(n) - \alpha\Delta T_j + K_{\psi}\mathcal{L}\psi_{i,j}(n) + \frac{v(\psi_{i,j+1}-\psi_{i,j})}{\Delta x}\right] \quad (14)$$

$$c_{i,j}(n+1) = c_{i,j}(n) + M_c\Delta t\left\{\mathcal{L}\left[wc_{i,j}(n) - 2\beta c_{i,j}(n)\psi_{i,j}(n) + (bc_{i,j}^{3}(n) - K_c\mathcal{L}c_{i,j}(n) + \right] + \frac{v(c_{i,j+1}-c_{i,j})}{\Delta x}\right\} \quad (15)$$

The $\mathcal{L}$ is known as the spatial operator, similar to the Laplacian operator, that accounts the surrounding local values. For example,

$$\mathcal{L}\psi_{i,j} = \frac{\psi_{i+1,j}+\psi_{i,j+1}+\psi_{i-1,j}+\psi_{i,j-1}-4\psi_{i,j}}{\Delta x^2} \quad (16)$$

where Δx is known as the mesh size of the particle unit. Δt is the step size for time and n is the step number. The step number determines the number of iterations to run the simulation. Multiplying the step time size with the step number will give the total time of the modeling. At equilibrium, nΔt approaches ∞. The remaining parameters are associated with the free energy constants such as kinetic factor, surface energy factor, diffusion gradient, mobility etc.

Model C Coding on MATLAB – work by previous COOP students

Models A, B and C were implemented, using the derived solutions for ψ and c, onto MATLAB. MATLAB is an analytical software that is willing to manipulate experimental and simulation data to generate graphical and imaging analysis. Codes may be written in MATLAB's "language" to perform computations. Having many features in the software library, algorithms of the phase field model may be implemented quite easily to run the simulation. Figure 4 shows a sample result of the simulation where each column of images represents the ψ and c microstructure respectively and the rows are time dependent and is a function of iterations or number of time steps, nt. (Time t = nt*dt, dt is time size)

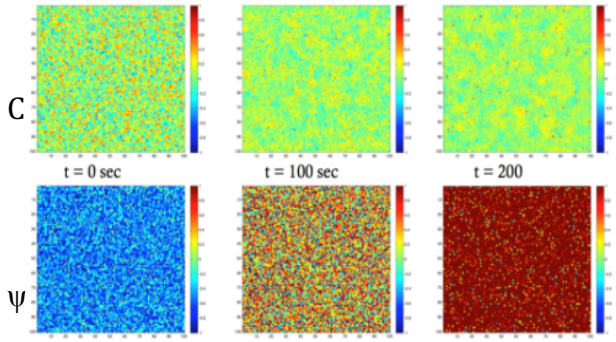


Figure 4 Sample result generated from Model C of MATLAB.

Previous developments of the phase field model, in which for this purpose was to model the eutectic growth of binary alloys, uses the derived equations found in Drotlet's paper. [1] The first development by Eric (previous COOP students), implemented the equation for Model C without the consideration of the noise term. Farah and Andrew (another previous COOP students) later inserted and optimized the noise term by using the Box-Muller, which satisfy the Gaussian distribution of randomly generated numbers around 0, accordingly to equation (9). Jason (latest COOP student) modified the several parameters to match the simulation file with experimental data. For improved accuracy and completion of the simulation to equilibrium, the program was modified to a function form, which can be simulated on supercomputers.

Though modeling the phase field using MATLAB produced promising results, there are some drawbacks concerned with the simulation efficiency. Coding with MATLAB does not require any variable declarations or space allocations and therefore consumes a fair amount of processing memory. The completion of a eutectic solidification would require a great number of time steps, which affects the number of iterations that the code must run. Without supercomputers or devices with high computational capabilities, generating a group of results is time consuming. Running phase field model simulations are therefore suggested on C, which involves greater control over the computer's data processing. There are some disadvantages however, with C language as will be discussed later. The remaining portion of the paper covers the conversion and details of the phase field model (Model C) in C language.

Development of Model C in C Language

There are several steps in the algorithm for the phase field model when developing it in C. The first step requires importing a list of libraries that contain

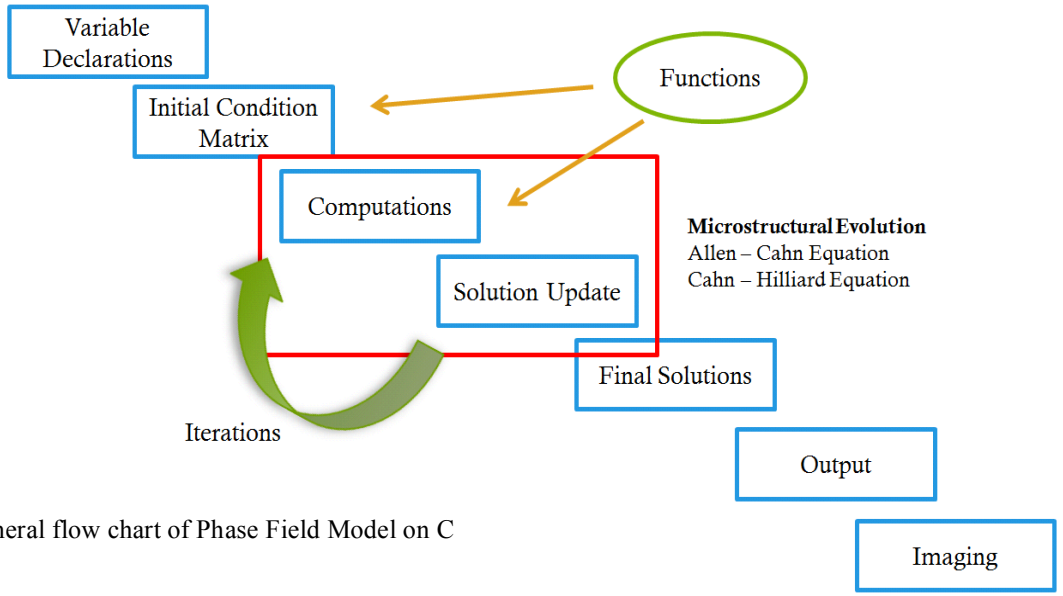


Figure 5 General flow chart of Phase Field Model on C

functions needed to perform the simulation. Some functions used in the previous MATLAB codes are not available in C. Therefore some functions, such as the random generator and image generator are coded manually in this program. Figure 5 shows a general flow chart of the phase field model algorithm.

The code is followed by the initial matrix setup. Before any variables are taking into account for computations, they must be declared with a certain data type which determines the amount of memory this value will occupy. To deal with sufficient calculation precisions and prevent the use of too much memory, most variables are restricted to double type. This portion can be seen below, code 1.

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>

double randgen();
//int imageconc();
//int writefile();

void main()
        //adjustable/main parameters
        double mpsi, mconc, r, a,b,w,deltat,kappaconc,kappapsi,alpha,beta,dtt,dx;
        double temp, cm;
        //space matrix paraters
        int ni, nj, nt;
        //constants
        double kb = 0.010, pi = M_PI, lapl_pref = 2.0/3.0;
        //other parameters or variables
        double dx2, invdx2,c_noise;
        double sum=0, sum1=0, sum2=0, sum3=0, sum4=0, sum5=0, sum6=0, totalarrays;
 int i, j, tloop, aup,adown,aleft,aright,aup2,adown2,aleft2,aright2;
        double d,noise_corr;
        double corr_conc, lapl_conc_cubed, lapl_conc, dub_lapl_conc, ppconc,corr_psi , euler, pppsi; //microstructural evolution parameters
        //random variable set up
        double ranr1, ranr2, ranr3, ranr4;
        time_t t;
        srand((unsigned) time(&t));
        //finishing variables
        double percentage, percentagev=10;

//Defining Prealllocated space

        //non conservered parameter
        static double npsi[200][200], noise[200][200], nnoise[200][200];
        static double nnoisec[200][200], psi[200][200];

        //conserved parameter
        static double conc[200][200], nconc[200][200];
        static double noise1[200][200], noise2[200][200];
        static double noisec1[200][200],noisec2[200][200];
```

Code 1 – Variable declaration, psi and conc matrix setup.

The parameters required for the phase field model, involving fitting parameters and material properties, are declared as well. Variables associated with the term “psi” (ψ) and “conc” (c) represents the microstructure of the material. These variables are 2D arrays with each value representing the solid/liquid state and level of concentration for psi and conc respectively. The noise matrix represents the thermal fluctuation that will be applied to the microstructure. The ranges for each values must be within [-1 to 1]. Note also that the matrix size is 200 x 200, which is also the size array or “pixel” of the microstructure which will be processed with the dynamics equation.

The psi and conc matrix is further manipulated by adjusting the values to create an initial condition of the

microstructure. The setup of the phase distribution and concentration in the microstructure are constructed base on the user's preference. Usually, these arrays are adjusted to match with microstructures found in real experiments. The program provides three forms of distribution of the psi and conc matrices, uniform, random generation in restricted range, and Gaussian distribution. These codes are seen in code 2.

```
//INITIAL CONDITION SET UP
  //setting up the initial space, psi and conc
          for (i = 0; i<ni; i++)
          for (j = 0; j<nj; j++)
                    ranr1 = randgen();
                    ranr2 = randgen();
                    ranr3 = randgen();
                    ranr4 = randgen();
                    //generating random concentration using gaussian distribution
                    //conc[i][j]                    = cos(2*pi*ranr1)*sin(2*pi*ranr2);
                    //generating random concentration lower than 0.05
                    conc[i][j] = 0.0 - 0.5*ranr1 + 0.5*ranr2;

                    //uniform concentration
                    //conc[i][j]      =      0.05;      //setting concentration to be uniform at 0.05

                    //generating random psi by gaussian around -0.5
                    //psi[i][j]          =          -0.5          + cos(2*pi*ranr3)*sin(2*pi*ranr4)/2.0;

                    //generating psi based on litereature pg. 6709 2nd paragraph on left
                    psi[i][j] = -1 + (-0.1 + 0.2*ranr3);
                    //uniform psi/ phase
                    //psi[i][j] = -0.8;
```

Code 2 – Initial set up of the microstructure in psi and conc matrices

This portion is important in determining the position of the microstructure on the phase diagram. The Gaussian distribution around a certain ψ or c value implements the Muller – Box transform by an addition of the cos(2π*ranr) x sin(2π*ranr) term. The method was also used in the random noise term. Several other adjustments can be made by inserting a seed of a solid phase, known as the droplet effect, or inserting a solid interface of the preferential form. These codes can be adjusted by the user to insert certain shape to enhance nucleation and alter the end result of microstructure. Several papers have inserted dendritic crystals to observe solidification of certain materials.

The dynamics equation is the main driving force of this simulation. The coding of this equation uses equation 14 and 15 to perform computations of each ψ or c value in the array. The formulation takes account of the surrounding values by using the Laplace operator, presented in equation 16. The portion of the dynamic code in this program is shown in code 3 and is fairly laborious. In summary, the simulation takes in a ψ array, formulate the values starting from position (1,1) and moves on to (2,1) , (3,1) and so on, until all the values in the array have been formulated with the dynamics equation. The c array is done in a similar manner but uses the conserved parameter equation. The array is updated and replaces the previous array. This process is done repeatedly depending on the number of iterations input by the user.

```
          //conc calculations

//Nearest neighbor of noise term

nnoise[i][j] = lapl_pref*(0.25*(noise1[aup][aleft] + noise1[adown][aright] + noise2[aup][aright] + noise2[adown][aleft]) + (noise1[aup][j] + noise1[adown][j] + noise2[i][aright] + noise2[i][aleft]) - 2.5*(noise1[i][j]+noise2[i][j]));

nnoisec[i][j] = lapl_pref*(0.25*(noisec1[aup][aleft] + noisec1[adown][aright] + noisec2[aup][aright] + noisec2[adown][aleft]) + (noisec1[aup][j] + noisec1[adown][j] + noisec2[i][aright] + noisec2[i][aleft]) - 2.5*(noisec1[i][j]+noisec2[i][j]));

//correlation term for concentration equation
corr_conc = -2.0*beta*lapl_pref*(0.25*(conc[aup][aleft]*psi[aup][aleft] + conc[aup][aright]*psi[aup][aright] + conc[adown][aleft]*psi[adown][aleft] + conc[adown][aright]*psi[adown][aright]) + (conc[i][aleft]*psi[i][aleft] + conc[i][aright]*psi[i][aright] + conc[aup][j]*psi[aup][j] + conc[adown][j]*psi[adown][j]) - 5*conc[i][j]*psi[i][j]);

//calculation of laplacian of concentration field cubed

lapl_conc_cubed = lapl_pref*(0.25*(pow(conc[aup][aleft],3) + pow(conc[aup][aright],3) + pow(conc[adown][aleft],3) + pow(conc[adown][aright],3)) + (pow(conc[i][aleft],3) + pow(conc[i][aright],3) + pow(conc[aup][j],3) + pow(conc[adown][j],3)) - 5*pow(conc[i][j],3));

//calculation of laplacian of concentration field

lapl_conc = lapl_pref*(0.25*(conc[aup][aleft]+conc[aup][aright]+conc[adown][aleft]
+conc[adown][aright])+(conc[i][aleft]+conc[i][aright]+conc[aup][j]+conc[adown][j])-5*conc[i][j]);

//double laplacian of concetration field
dub_lapl_conc = pow(lapl_pref,2)*(29.25*conc[i][j]-9*(conc[i][aleft]+conc[i][aright]+conc[adown][j]+conc[aup][j])-0.5* (conc[aup][aleft]
+conc[aup][aright]+conc[adown][aleft]+conc[adown][arig
```

```
ht])+1.125*(conc[i][aleft2]+conc[i][aright2]+conc[aup2][j
]+conc[adown2][j])+0.0625*(conc[aup2][aleft2]+conc[aup
2][aright2]+conc[adown2][aleft2]+conc[adown2][aright2])
+0.5*(conc[aup2][aleft]+conc[aup2][aright]+conc[adown2
][aleft]+conc[adown2][aright]+conc[aup][aleft2]+conc[au
p][aright2]+conc[adown][aleft2]+conc[adown][aright2]));

//equation 14
ppconc                                                    =
(mconc*dtt*invdx2)*(w*(lapl_conc)+corr_conc+b*(lapl_c
onc_cubed)-
(kappaconc*invdx2)*(dub_lapl_conc))+2*nnoisec[i][j];

        //psi calculations based on 9-point laplacian

//correlation term for psi calculation

corr_psi=1.0*beta*(pow(conc[i][j],2)-alpha*deltat);

//euler method
euler=lapl_pref*(0.25*(psi[aup][aleft]+psi[aup][aright]+ps
i[adown][aleft]+psi[adown][aright])+(psi[i][aright]+psi[i][
aleft]+psi[aup][j]+psi[adown][j])-5*psi[i][j]);

//equation 13
pppsi=mpsi*dtt*(r*psi[i][j]-a*pow(psi[i][j],3)
+corr_psi+(kappapsi*invdx2)*euler) +nnoise[i][j];

//npsi and nconc array update – addition

npsi[i][j] = psi[i][j]+pppsi;
nconc[i][j] =conc[i][j]+ppconc;
```

Code 3 – Dynamics calculation of ψ and c values

When the matrices go through all the iterations or time step (represented as nt) the updated ψ and c matrices are then written to a text file in the directory of the software. This ends the phase field model by giving the output of the psi.txt and conc.txt, and the log file. The updated code also goes through another function which converts the array to an image file which visualizes the microstructural evolution of the ψ and c matrices.

As mentioned before, the image generator is not built into any of the existing C language libraries. The function must therefore be coded manually. Image generation is important as it allows the user to visualize and view the microstructural change in the material. Although statistical analysis may be done using the data text file that the program outputs, the image may be used to give a general idea of what the phase will look like. In this program, the psi and c array is normalized to ranges from 0 to 255, representing the intensity of a color. A 3D array is created to and the codes are converted such that certain values in the psi or c matrix will represent a certain color. The 3D array is then written to a file to form a ppm format image, which is known to contain a relatively simple format for image files. The code is written below, code 4.

```
        FILE* imagefile1 = fopen("imageconc.ppm", "w"
);
        FILE* imagefile2 = fopen("imagepsi.ppm", "w");
        fprintf(imagefile1,"P3\n#imageconc.ppm\n%d
%d\n", ni, nj);
        fprintf(imagefile1,"%d\n",255);

        fprintf(imagefile2,"P3\n#imagepsi.ppm\n%d
%d\n", ni, nj);
        fprintf(imagefile2,"%d\n",255);

        for (i=0; i<=ni-1;i++)
        {
        for (j=0; j<=nj-1; j++)
        {
        for (k=0; k <=2; k++)
        {
        normalizedconc[i][j]    =    (nconc[i][j]+1)*255/2;
//normalizing values
        imageconc[i][j][k]                                =
(int)trunc(normalizedconc[i][j]);
        imagetemp = imageconc[i][j][k];
        if(imagetemp>250){imagetemp = 250;}
        if(imagetemp<0){imagetemp = 0;}
        if (imagetemp <= 65)//color jet - sequence is blue
green red
        {
        imageconc[i][j][1] = imagetemp*255/65;//green
        imageconc[i][j][0] = 0;//red
        imageconc[i][j][2] = 255; //blue
        }
        if (imagetemp >65 && imagetemp<= 117)
        {
        imageconc[i][j][2]                =                -
255/52*imagetemp+578;//blue
        imageconc[i][j][0] = 0;//red
        imageconc[i][j][1] = 255; //green
        }
        if (imagetemp >117 && imagetemp< 170)
        {
        imageconc[i][j][2]      =      255/51*imagetemp-
590;//blue
        imageconc[i][j][0] = 255;//red
        imageconc[i][j][1] = 0; //green
        }
        if (imagetemp >= 170)
        {
        imageconc[i][j][2] = 0;//blue
        imageconc[i][j][1]                =                765-
255/85*imagetemp;//green
        imageconc[i][j][0] = 255; //red
        }
        fprintf(imagefile1, "%d ", imageconc[i][j][k]);


        normalizedpsi[i][j]     =     (npsi[i][j]+1)*255/2;
//normalizing values
        imagepsi[i][j][k] = (int)trunc(normalizedpsi[i][j]);
        imagetemp = imagepsi[i][j][k];
        if(imagetemp>250){imagetemp = 250;}
```

```
        if(imagetemp<0){imagetemp = 0;}
        if (imagetemp <= 65)//color jet
        {
        imagepsi[i][j][1] = imagetemp*255/65;//green
        imagepsi[i][j][0] = 0;//red
        imagepsi[i][j][2] = 255; //blue
        }
        if (imagetemp >65 && imagetemp<= 117)
        {
        imagepsi[i][j][2]                =                -
255/52*imagetemp+578;//blue
        imagepsi[i][j][0] = 0;//red
        imagepsi[i][j][1] = 255; //green
        }
        if (imagetemp >117 && imagetemp< 170)
        {
        imagepsi[i][j][2] = 255/51*imagetemp-590;//blue
        imagepsi[i][j][0] = 255;//red
        imagepsi[i][j][1] = 0; //green
        }
        if (imagetemp >= 170)
        {
        imagepsi[i][j][2] = 0;//blue
        imagepsi[i][j][1]                =                765-
255/85*imagetemp;//green
        imagepsi[i][j][0] = 255; //red
        }
        fprintf(imagefile2, "%d ", imagepsi[i][j][k]);
                        }
        fprintf(imagefile1, "\n");
        fprintf(imagefile2, "\n");
        }
        }


                        //printf("\nimage        generator
complete. . .\n");
        fclose(imagefile1);
        fclose(imagefile2);
```

Code 4 – Image generator to ppm format

This code tries to mimic the color jet scheme of converting 2D arrays with values under a certain range and represent it into different colors. [9] To apply this method, low range values in the psi or c matrix, lower than -0.6, are represented are red, while values higher than 0.6 would represent blue. Any values in between would be a mixture of red with green and blue with green respectively. Values near 0 will show up as green. This representation can be seen in figure 6.

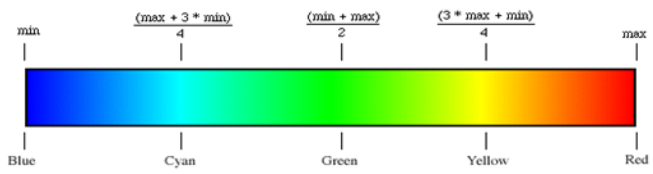


Figure 6 – color representation from 2D array [9]

Figure 7 shows a screenshot of the program after the completion of running the simulation. The files displayed on the right side of the figure, in the red boxes, are the output files created by the program. The first two are the output data files for the computed ψ and c matrices. The next two are the image files of the microstructure, while the last file is the log which records the condition input on each run of the software.

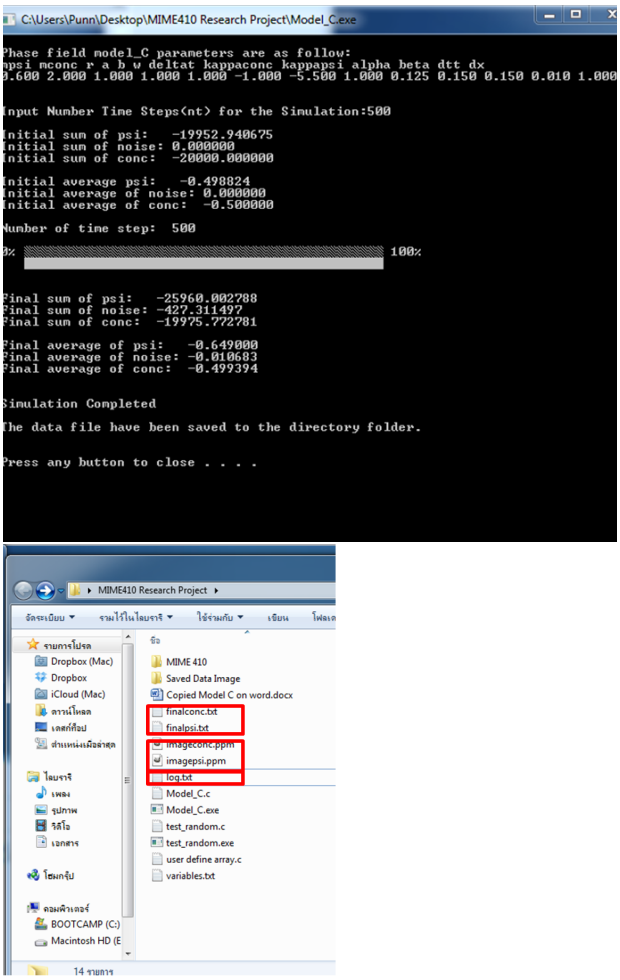


Figure 7 – View of the Phase field Model program on C

Results from Simulation Using C

Running simulations with the same parameters and same computer compared the efficiency of the simulation written in C and MATLAB. Figure 8 shows that the phase field model on C was much more efficient in terms of simulation time. Running the same simulation on C under 5000 iterations would take around 5 min to run, while it would take as long as 1 hour and 30 min when running on MATLAB. Using C to run the phase field model would therefore allow more measurements to run without the need to use powerful computers.

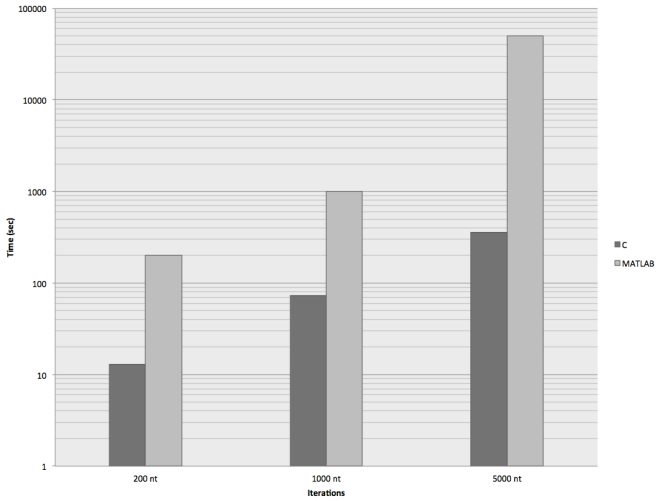


Figure 8 Simulation comparisons running on C and MATALB. The axis are time (sec) and number of iterations for x and y axis respectively.

Example Simulations

To observe some of the simulation results generated from the C program, a few examples were made. The first example will assume a uniform liquid microstructure with a Gaussian distribution concentration around 0. This means that it will contain 50wt% of element A and 50wt% of element B. The same phase diagram generated in this paper was used. Therefore, the parameters used to fit the phase diagram are the same and will be used in the dynamics equation. The microstructure will undergo a rapid quench which experiences a sudden reduction in temperature. This condition will occur when the code begins to run through the dynamics equation because it involves the condition temperature as a parameter. For this example, the condition temperature was set to $\Delta T = -0.5$.

Keep in mind that the temperature value is normalized and has its own arbitrary units. These ranges of temperatures can also been seen in the phase diagram. Figure 9.

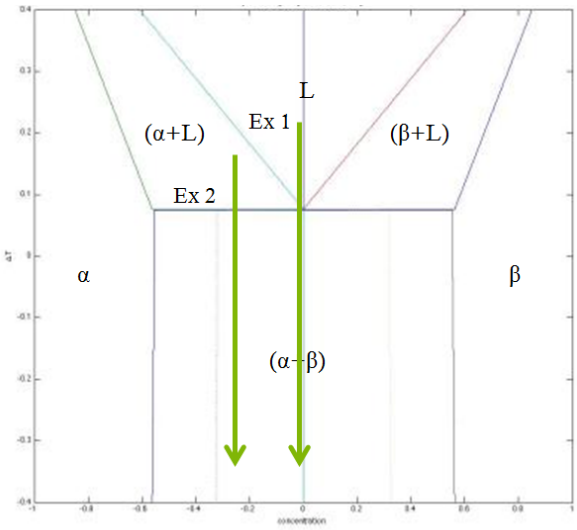


Figure 9 – Phase diagram of the assumed material

The result of the eutectic simulation shows both the ψ (solid/liquid) and c (concentration) microstructural pattern at each time interval. By starting at a randomized concentration with average around c = 0.0, the concentration segregates over time and the pattern becomes visible. The state of matter or ψ of this system doesn't change as much due to the shortened time for solidification. To obtain a steady state microstructure, the iteration time would need to approach infinity or some values where nt or number of time steps is very high. Result of this example is shown in figure 10.

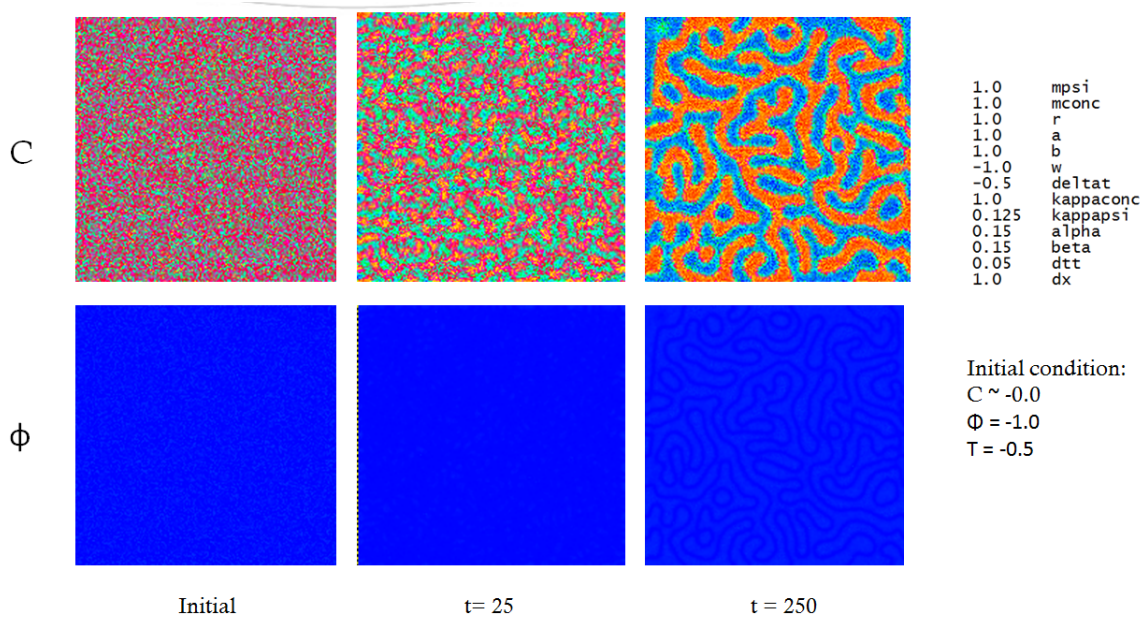


Figure 10 - Example 1 – eutectic growth of material with input parameters and conditions on the right side of the images.

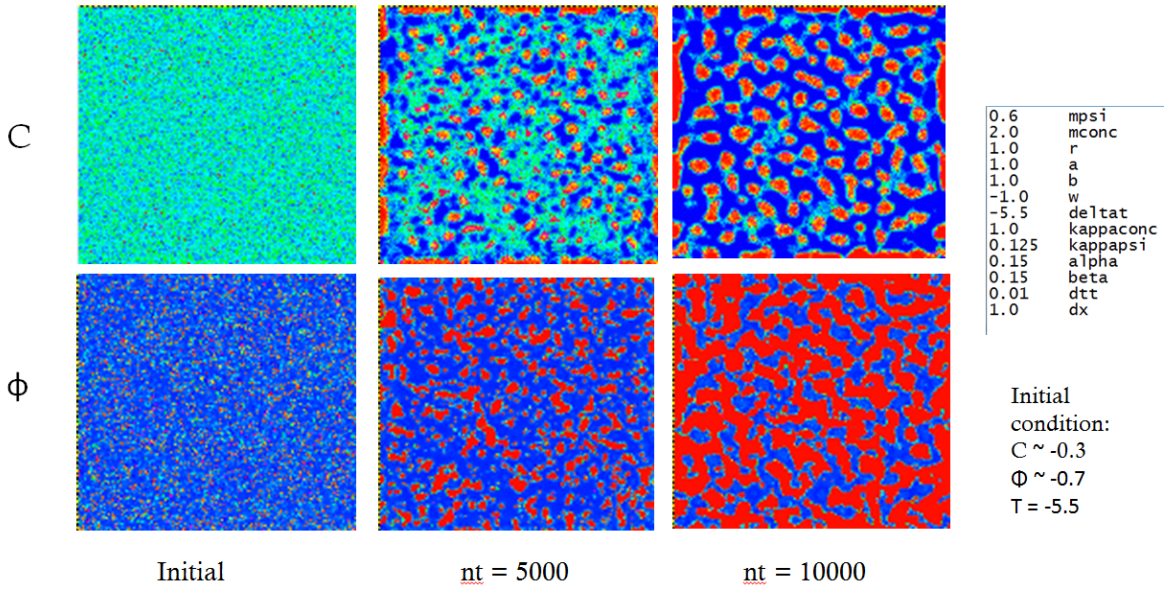


Figure 11 - Example 2 – eutectic growth of material with input parameters and conditions on the right

The second example deals with the solidification from a region between the solidus and liquidus line. (α+L) In this region, the microstructure is partially solid, where ψ is randomly distributed around ψ = -0.7 and concentration uses Gaussian distribution around c = -0.3. To show that the quench in this example can form a higher amount of solid composition, the conditions are set to a much lower temperature, ΔT = -5.5. The iteration time has been increased to 10000 time steps. With the presence of some solids in the initial condition, the nucleation for solidification is enhanced. After certain iterations or time, the solid begins to form as shown on figure 11. The pattern for concentration also experience changes where a high degree of segregation occurs. For steady state condition or at equilibrium, the ψ concentration will represent all solid while the concentration will become very segregated and form a distinctive pattern. Keep note that the ψ matrix is not conserve so the sum of all arrays can change when compared to the initial conditions. Concentration is conserved so the sum of all concentration values in the array is kept constant.

Conclusion

From the study of the phase field model theory, the equations for the dynamics term have been implemented for simulation. Model C was coded in MATLAB and produced great results. However, because of MATLAB's incompatibility to run simulations with numerous iterations, the simulation is very time consuming. The phase field model was further developed by implementing theories onto C language. Since C is more suitable to run computations with multiple iterations, the results showed the simulation time to be reduced significantly compared with MATLAB. The performance of the phase field model on C was presented by performing two sample simulations of eutectic systems, which produced reasonable results. This software can be used to compare simulation outcomes with real experimental data, as well as further analyzing its compositional and microstructural properties found in the material.

References

[1] F. Drolet, K., R. Elder, M. Grant, J. M. Kosterlitz, Physical Rev. E 61 (2000) 6705 – 6719
[2] A. J. Bray, Adv. Phys. 32 (1994) 357 - 459
[2] L. Q. Chen, Annual Rev. Mater. Res. 32 (2002) 113 – 40
[3] A. Karma, Physical Rev. E 49 (1994) 2245 - 2250
[4] Steinbach, Annu. Rev. Mater. Res. 43 (2013) 89-107
[5] K. Elder, Comp in Physics 7 (1993) 27 – 33
[6] S. M. Allen, J.W. Cahn, Acta Metall. 27 (1978) 1085
[7] J. W. Cahn, J. E. Hiliard, J. Chem. Phys. 28 (1958) 258
[8] H. E. Cook, Acta Metall. 18 (1970) 297
[9] P. Bourke, "Colour Ramping for Data Visualisation" (1996), University of Western Australia, "paulbourke.net