Hybrid Model for Spinodal Decomposition using Cahn-Hilliard Equation and Principle Component Analysis

Kartikeya Sharma

Undergraduate

Department Of Materials Science and Engineering

Indian Institute of Science

Submitted in partial satisfaction of the requirements for the
Degree of Bachelor of Science (Research)
in Materials Science

Supervisor Dr. Abhik Choudhury

June 2020

Declaration

I certify that -

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

Date - 25 June, 2020 Kartikeya Sharma

Place - IISc, Bangalore 11-01-00-10-91-16-1-13704

UG PROGRAMME

INDIAN INSTITUTE OF SCIENCE

BANGALORE - 560012, INDIA

CERTIFICATE

This is to certify that the Bachelor's thesis entitled "**Hybrid Model for Spinodal Decomposition using Cahn-Hilliard Equation and Principle Component Analysis**" submitted by **Kartikeya Sharma** (Sr No. 11-01-00-10-91-16-1-13704) to Indian Institute of Science, Bangalore towards partial fulfilment of requirements for the award of degree of Master of Science(Research) in Materials Science is a record of bona fide work carried out by him under my supervision and guidance during Academic Year, 2019-20.

Date - 25 June, 2020

Place - IISc, Bangalore

Dr. Abhik Choudhury
Materials Engineering

Indian Institute of Science
Bangalore - 560012, India

Abstract

Phase-field method has become a widely used technique for modelling and simulations in materials science due to its fundamental roots and simplicity in coding. It is thermodynamically consistent model due to which it can take a long time to reach the desired timestep.

While data-driven research is common in areas like biology and chemistry, it was only recently that a surge of interest in machine learning has hit the area of materials science. The large-scale use of simulation techniques to produce a vast amount of data allows us to use data science techniques for further enhancement. One such technique called Principal Component Analysis has been used in this work to fasten up the phase-field simulation of spinodal decomposition in a binary alloy.

A 'hybrid model' was developed combining the standard phase-field method and use of PCA with images in predicting the microstructures. Using the hybrid model, it was shown that a significantly faster approach is possible to simulate the time evolution of microstructures with great accuracy.

Acknowledgements

I would like to begin by expressing my gratitude and appreciation for my supervisor, Dr. Abhik Choudhury, whose guidance, support and encouragement has been invaluable throughout this project. His overall insights in this field have made this an inspiring experience for me.

I am extremely grateful to Dr. S Karthikeyan and Dr. Abhik Choudhury again, for our friendly chats and your personal support in my Undergraduate life at IISc.

I also thank the UG Programme and the Dept. of Materials Engineering for the facilities provided.

I am forever thankful to my friends and family for their support.

Thanks to Sidharth, Kanav, Aditya, Ocima, Abhigya and others for always being there and making the time enjoyable.

I enjoyed my stay xqcL

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Microstructures

The study of microstructures lies in the heart of materials science. The properties of most modern materials are engineered by changing its microstructure in desired ways. The free boundary dynamics and non-equilibrium phase transformation govern the development of these microstructures. To eliminate the trial and error method with real materials to obtain desired microstructures, we can model this process using modern simulation tools, such as phase-field methods.

1.2 Aim of the thesis

The main objective of this work was to use the modern data science tools with phase-field simulation to construct a hybrid model which can make the simulations considerably faster. A 2-dimensional phase-field simulation of spinodal decomposition in a virtual A-B alloy was done in python using the Cahn-Hilliard equation. Using the microstructure images generated from the simulation, Principal Component Analysis (PCA) was used on them to get the trend of the principle components (PC). Using the trend of PCs as microstructure develops, the next few sets of PCs were predicted using different methods such as - regression, Convoluted Neural Network (CNN) and curve interpolation. The phase-field simulations for spinodal decomposition were made considerably faster using this prediction based model called "hybrid model" hereafter in the thesis.

1.3 Spinodal Decomposition

The spontaneous phase separation of one thermodynamic phase into two co-existing phases due to thermal fluctuations is called spinodal decomposition. It is generally a fast process happening at nano-scale, making it hard to detect experimentally and results in a reduction of the free energy of the system.

Assuming regular solution model of A-B alloy, the molar Gibbs free energy of the system, G, is given by -

$$G = G_A(1-c) + G_B c + RT[c\ln c + (1-c)\ln 1 - c] + \Omega c(1-c) \tag{1.1}$$

where, R is universal gas constant and Ω is the molar heat of mixing. Cahn showed that the chemical spinodal is given by [2] -

$$\frac{\partial^2 G}{\partial c^2} = 0 \tag{1.2}$$

The critical temperature is the highest coexistence temperature and for this system, it is given by -

$$T_c = \frac{\Omega}{2R} \tag{1.3}$$

For temperature below T_c, critical fluctuations in ϕ result in change in ϕ from the initial value to their asymptotic values on the phase diagram.

1.4 Phase-field methods

Compared to sharp interface models, the phase-field method is a relatively new technique that is used by many materials scientists due to its more fundamental nature which avoids the problems faced by the former models. It has been used to study a variety of problems including phase transitions, directional solidification, grain growth, nucleation and spinodal decomposition, etc. [1] It introduces an additional continuum field called *order parameter* to describe the diffused interface between phases. It has become the "standard method" for numerical simulations of solidification microstructures.[7] Cahn and Hilliard showed that the phase-field methods are the appropriate choice to study the time evolution of the microstructure in spinodal decomposition. [3] The geometry of the microstructures is represented by scalar functions called *phase fields*, which take constant values in the bulk of each phase and vary smoothly through the interfaces between the phases.

In Cahn-Hilliard's approach for a conserved order parameter, such as concentration of impurity, the free energy *functional* of concentration for a binary alloy is written as -

$$F(c, \nabla c, ..) \approx N_V \int_V dV[f_0(c) + \kappa|\nabla c|^2] \tag{1.4}$$

where κ is the gradient energy coefficient and $f_0(c)$ is the bulk free energy density. The chemical potential is given by the variational derivative of the functional -

$$\mu = \frac{\delta(F/N_V)}{\delta c} = \frac{\partial f_0}{\partial c} - \kappa\nabla^2 c \tag{1.5}$$

Using this functional, the equation of motion for concentration of impurity in a phase separating alloy mixture is given by -

$$\frac{\partial c}{\partial t} = \nabla \cdot \left(M\nabla\frac{\delta F}{\delta c}\right) = \nabla \cdot \left(M\nabla\left(\frac{\partial f_0}{\partial c} - \kappa\nabla^2 c\right)\right) \tag{1.6}$$

where M is the mobility of the solute. This is known as the Cahn-Hilliard equation. For non-conserved order parameters, equation of motion for the parameter is given by the Allen-Cahn equation -

$$\frac{\partial \phi}{\partial t} = -M\nabla\left(\frac{\delta F[\phi, T]}{\delta \phi}\right) \tag{1.7}$$

1.5 Principal Component Analysis

Principal Component Analysis (PCA) is a technique used to reduce dimensionality or simplify complicated and large datasets. It was first introduced by Karl Pearson in 1901 [5] and was used as a practical computing method with machines by Hotelling in 1933 [6]. General classification for a set of microstructures has been done using PCA. It was shown that analysis and quantification of the features in an ensemble of microstructures can be done using data-driven approaches. [4]

Consider a set of n interrelated variables $\boldsymbol{x_1, x_2, \ldots x_n}$. The goal of PCA is to transform these variables by calculating variance and maximising it, to get an uncorrelated set of variables $\boldsymbol{\phi_1, \phi_2, \ldots \phi_r}$ with $r << n$. These $\boldsymbol{\phi_i}$ are called Principal Components (PCs). This can be achieved by finding the covariance matrix of the original data and performing eigenvalue decomposition on the covariance matrix. Finally, PCA projects the data to an orthogonal space whose directions are ordered from highest to lowest variance. For example, if in a large dataset, 4 of the first PCs capture 90% of the total variance, we can choose these PCs to represent the original dataset without loss of much information.

The python library `sklearn.decomposition` was used to perform PCA. It creates a matrix which summarises how the interrelated variables relate to one another. This matrix contains information about the "directions" (eigenvectors) and their associated weights. The weights signify how important a direction is for a particular variable. If ϕ_1's weight is lesser for x_1 than that of x_2, we can say that x_2 is closer to ϕ_1 than x_1. Further in the thesis, when we say *"for image 200, PC values are (6, 3, 2)"*, it means the weights of first, second and third direction are 6,3 and 2 respectively. The matrix can also be used to back transform the PCs into the real space by performing an inverse transformation.

Chapter 2

Model

All codes were written in python. The model is divided into three main categories -

1. Phase-field model
2. Image Processing and PCA
3. Predicting microstructures

2.1 Phase Field Model

Consider a binary mixture of A-B atoms with c giving the atomic fraction of B atoms. At a given point, using Taylor expansion in gradients (Cahn-Hilliard's approach), the free energy density can be approximated around mean composition as -

$$f(c, \nabla c, \nabla^2 c \ldots) = f_0 c + \sum_i L_i \frac{\partial c}{\partial x_i} + \sum_{i,j} m_{ij} \frac{\partial^2 c}{\partial x_i \partial x_j} + \frac{1}{2} \sum_{i,j} n_{ij} \frac{\partial c}{\partial x_i} \frac{\partial c}{\partial x_j} + \ldots \quad (2.1)$$

where,

$$\begin{aligned}L_i &= \left[\frac{\partial f}{\partial\left(\frac{\partial c}{\partial x_i}\right)}\right]_0\\ m_{ij} &= \left[\frac{\partial f}{\partial\left(\frac{\partial^2 c}{\partial x_i \partial x_j}\right)}\right]_0\\ n_{ij} &= \left[\frac{\partial^2 f}{\partial\left(\frac{\partial c}{\partial x_i}\right)\partial\left(\frac{\partial c}{\partial x_j}\right)}\right]_0\end{aligned} \tag{2.2}$$

Assuming centrosymmetric crystal, $L_i = 0$ since the sign of the term changes as we change the sign of x-axis. Similarly, for other cases, we have (replacing partial derivatives with gradients for generalisation)-

$$\begin{aligned} & L_i = 0\\ m_{ij} & \begin{cases} = m = \left[\frac{\partial f}{\partial \nabla^2 c}\right]_0, & \forall i = j\\ = 0, & \forall i \neq j\end{cases}\\ n_{ij} & \begin{cases} = n = \left[\frac{\partial^2 f}{\partial |\nabla c|^2}\right]_0, & \forall i = j\\ = 0, & \forall i \neq j\end{cases}\end{aligned} \tag{2.3}$$

Using equations (2.2) and (2.3) in (2.1), Free energy of the system integrated over volume is given by -

$$F = N_V \int_V [f_0(c) + m\nabla^2 c + \frac{n}{2}(\nabla c)^2 + \ldots]dV \tag{2.4}$$

where N_V is the number of atoms per unit volume. Integrating the second term of (2.4) by parts and assuming infinite system so that surface integral vanishes,

$$\int_V m\nabla^2 c dV = -\int_V \frac{dm}{dc}(\nabla c)^2 dV + \int_S m\nabla c \cdot \hat{n} dS = -\int_V \frac{dm}{dc}(\nabla c)^2 dV \tag{2.5}$$

So we have,

$$\begin{aligned} F &= N_V \int_V [f_0(c) + \left(\frac{n}{2} - \frac{dm}{dc}\right)(\nabla c)^2 + \ldots]dV \\ &= N_V \int_V [f_0(c) + \kappa(\nabla c)^2 + \ldots]dV \end{aligned} \tag{2.6}$$

where the gradient energy coefficient, $\kappa = \left(\frac{n}{2} - \frac{dm}{dc}\right)$. Dropping higher order terms, we get the Ginzburg–Landau free energy in N_V units -

$$\mathcal{F} = \int_V [f_0(c) + \kappa(\nabla c)^2]dV \tag{2.7}$$

The first term in (2.7) is the free energy of the homogeneous solution of this volume and second term is the gradient energy which is a function of local composition. And, the functional derivative of $\mathcal{F}$ is given by Euler-Lagrange equation -

$$\frac{\delta \mathcal{F}}{\delta c} = \frac{\partial f}{\partial c} - 2\kappa \nabla^2 c \tag{2.8}$$

Now, to conserve total number of B atoms, this must hold true -

$$\lambda \int_V [c - c_0]dV = 0 \tag{2.9}$$

where c_0 is the alloy composition and λ is a Langrange multiplier. So, we modify (2.7) -

$$\begin{aligned} \mathcal{F} &= \int_V [f_0(c) + \kappa(\nabla c)^2 - \lambda(c - c_0)]dV \\ \delta\mathcal{F} &= \int_V \delta[[f_0(c) + \kappa(\nabla c)^2] - \lambda\delta(c - c_0)]dV \end{aligned} \tag{2.10}$$

Using (2.8), we get -

$$\delta\mathcal{F} = \int_V \left[\frac{\partial f}{\partial c} - 2\kappa\nabla^2 c - \lambda\right] \delta c \, dV \tag{2.11}$$

At equilibrium, $\delta\mathcal{F} = 0$. So, we have -

$$\lambda = \frac{\partial f}{\partial c} - 2\kappa\nabla^2 c = \frac{\delta\mathcal{F}}{\delta c} \tag{2.12}$$

λ is called the *generalised chemical potential.*

The classical diffusion equation is $\mathcal{J} = -M\nabla(\mu_A - \mu_B)$, where $\mathcal{J}$ is the flux, μ is chemical potential and M is mobility. We can modify it by using the generalised chemical potential to get -

$$\mathcal{J} = -M\nabla\lambda \tag{2.13}$$

And, from mass conservation, we get -

$$\begin{aligned} \frac{\partial c}{\partial t} &= -\nabla \cdot \mathcal{J} \\ \frac{\partial c}{\partial t} &= \nabla \cdot \left(M\nabla \frac{\delta \mathcal{F}}{\delta c} \right) \\ \frac{\partial c}{\partial t} &= \nabla \cdot \left(M\nabla \left(\frac{\partial f}{\partial c} - 2\kappa\nabla^2 c \right) \right) \end{aligned} \tag{2.14}$$

which is the Cahn-Hilliard equation.

In the experiment, simulations were done for 2-D binary alloy and the function $f(c)$ (free energy potential) was chosen to be a double-well potential of the form $f = Wc^2(1-c)^2$ as shown in Fig. 2.1. As shown in the figure above, the two stable

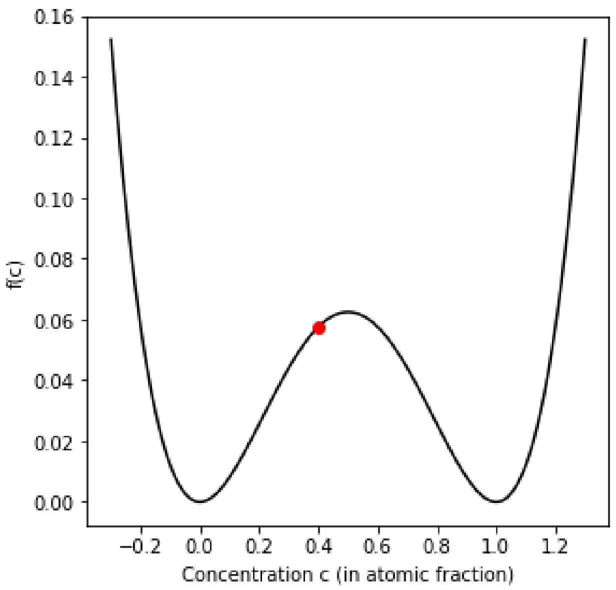


Figure 2.1: A double well potential for thermodynamical free energy $f(c)$.

states are at $c = 0$ and $c = 1$. If we start from an intermediate value of c, let's say,

$c_0 = 0.4$, the system will go under spinodal decomposition and we will observe grain coarsening in the binary alloy.

Using our double well potential in the Cahn-Hilliard equation, we get the following equation of motion -

$$\frac{\partial c}{\partial t} = \nabla \cdot \Big(M \nabla (-2Wc(1-c)(1-2c) - 2\kappa \nabla^2 c) \Big) \tag{2.15}$$

Assuming the mobility of B atoms, M, doesn't change with c in the model (to neglect the $\frac{\partial M}{\partial c}$ term). Writing eq (2.15) in terms of generalised chemical potential, λ, in 2-D -

$$\frac{\partial c}{\partial t} = M \nabla^2 \lambda = M \left(\frac{\partial^2 \lambda}{\partial x^2} + \frac{\partial^2 \lambda}{\partial y^2} \right) \tag{2.16}$$

In 2-D, we can write the generalised chemical potential, λ as -

$$\lambda = \frac{\delta \mathcal{F}}{\delta c} = -2Wc(1-c)(1-2c) - 2\kappa \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) \tag{2.17}$$

2.1.1 Cahn-Hilliard Solver

Presented below is a numerical method to solve the eq.(2.16) by finite difference method. Let the computational domain be a $(100, 100)$ grid with cartesian coordinates. So, $c^n_{i,j}$ will be the concentration of B at point (i, j) in the grid at n^{th} timestep, where total time, $t_n = n\Delta t$.

In the simulations, following were the values of constants -

Constant	Value
Timestep (Δt)	0.05
Mobility (M)	0.1
W	1.0
Gradient energy constant (κ)	1.0
Grid size (n_x, n_y)	(100, 100)
Step in x-direction (Δx)	1.0
Step in y-direction (Δy)	1.0

Table 2.1: Values of constants in the model.

The model starts with randomising the mesh points with the given c_0 (starting concentration) and adding random noise around c_0 to make the program a bit faster during initiation. This part of model is shown below -

```python
c = np.zeros( (mesh_x, mesh_y) )
pct = int(sys.argv[1])  #Asks user for c_0 value (in percent)

def initialize_random(c):
    ### Saving random array with SEED ###
    np.random.seed(2020)
    random_no = np.random.random(mesh_x * mesh_y)
    i = 0
    for x in range(mesh_x):
        for y in range(mesh_y):
            c[x][y] = 0.01*pct + random_no[i]*0.1 # random noise
            i = i+1
    return c
```

Random seed was used to have reproducibility of the same microstructure over different simulations. While imposing periodic boundary conditions, 2 buffer points at each boundary were used since we are using 4^{th} derivative in space (∇^4) -

$$c_{i,0} = c_{i,n_y-4} \qquad c_{i,1} = c_{i,n_y-3} \qquad c_{i,n_y-2} = c_{i,2} \qquad c_{i,n_y-1} = c_{i,3}$$

$$c_{0,i} = c_{n_x-4,i} \qquad c_{1,i} = c_{n_x-3,i} \qquad c_{n_x-2,i} = c_{2,i} \qquad c_{n_x-1,i} = c_{3,i}$$

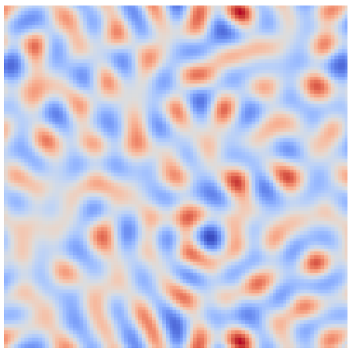
Writing the generalised chemical potential from (2.17) in finite elements -

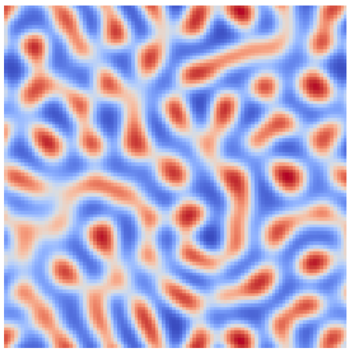
$$\begin{aligned}\lambda^n_{i+1,j} = &-2Wc^n_{i,j}(1-c^n_{i,j})(1-2c^n_{i,j})\\ &-2\kappa\left(\frac{c^n_{i+1,j}-2c^n_{i,j}+c^n_{i-1,j}}{(\Delta x)^2}+\frac{c^n_{i,j+1}-2c^n_{i,j}+c^n_{i,j-1}}{(\Delta y)^2}\right)\end{aligned} \tag{2.18}$$

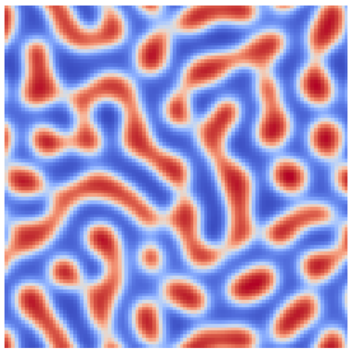
Similarly, writing the Cahn-Hilliard equation from (2.16) in finite elements -

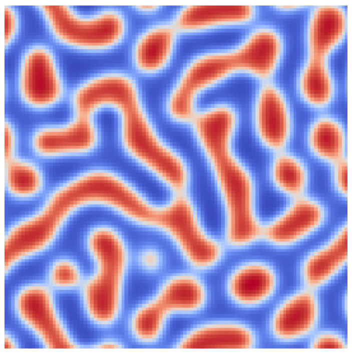
$$\frac{c^{n+1}_{i,j}-c^n_{i,j}}{\Delta t} = M\left(\frac{\lambda^n_{i+1,j}-2\lambda^n_{i,j}+\lambda^n_{i-1,j}}{(\Delta x)^2}+\frac{\lambda^n_{i,j+1}-2\lambda^n_{i,j}+\lambda^n_{i,j-1}}{(\Delta y)^2}\right) \tag{2.19}$$

The equations (2.18) and (2.19) need to be solved with the given periodic boundary conditions to model the spinodal decomposition in A-B alloy.


(a) $t = 5000$


(b) $t = 8000$


(c) $t = 15000$


(d) $t = 15000$

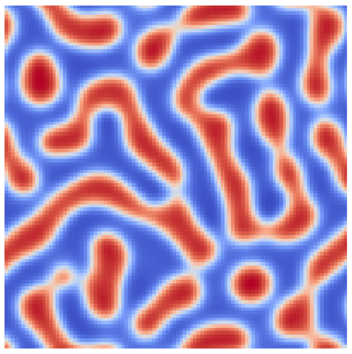

(e) $t = 24000$

Figure 2.2: Microstructures with c=0.4 at different timesteps

2.2 Image processing and PCA

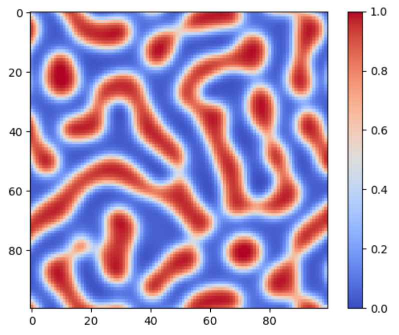
2.2.1 Making dataset

Using the solver above to create a dataset, images were saved every 100 timesteps after 5000 timesteps to give the microstructure some time to grow and develop initial clusters. Some of the images that were saved initially are shown in Fig. 2.2. They are coloured RGB images with 4 channels, one for each primary colour and one 'alpha' channel which controls transparency. Loading them as an array showed that each of them had pixel attributes like - $(r_i, g_i, b_i, 1)$ where all r_i, g_b and b_i had different values for each pixel. Alpha channel was ignored since it had the same value for each pixel, which was 1. To make an easier dataset, the same composition profile was saved in `binary_r`. This means that for each pixel, $r_i = g_b = b_i$. Now, the task was to only store one value for each pixel in an image instead of storing 3, i.e., for an image of resolution 369×369 instead of storing an array of shape $(369, 369, 3)$, an array of shape $(369, 369)$ was used for further processing.

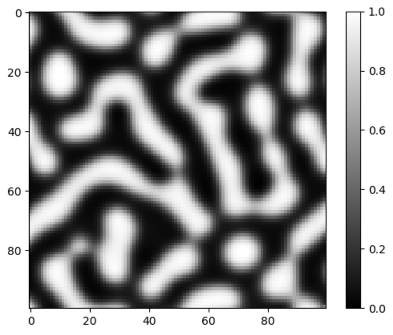
Using attributes `vmin=0` and `vmax=1` in the `matplotlib.pyplot` package makes sure that the data is mapped one-to-one between $[0, 1]$ which is the range of variation of concentration of B atoms. It is also worth noting that it was not possible to save the microstructure, which means saving the c array of shape $(100, 100)$, as an image of 100×100 resolution with one-to-one mapping. Using the function `pyplot.savefig` made its own representation (for the sake of image quality) and it was saved as an image of resolution 369×369. On adjusting the `dpi` attribute of the function, 100×100 resolution image was saved but the one-to-one mapping to c was not present. It is necessary to get the c values directly from image for the model to work, i.e., each pixel value must be equal to the concentration of B at that point in mesh grid.

$$\text{pixel}(i, j) = c_{i,j} \tag{2.20}$$

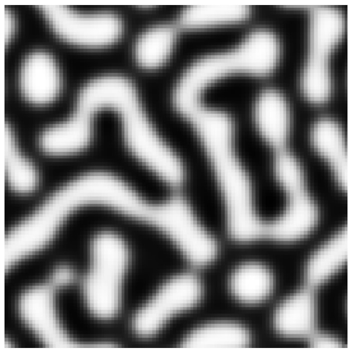
For (2.20) to hold true, `pyplot.imsave` function was used which allowed the image to be saved with the given dimension of array which is 100×100 in this case and the mapping was one to one.



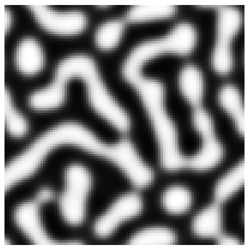
(a) 369×369 coloured image with a colour-bar



(b) 369×369 binary image with a colour-bar and options `vmin=0` and `vmax=1`



(c) 369×369 binary image with options `vmin=0` and `vmax=1`. White borders were removed.



(d) 100×100 binary image without borders, saved using `pyplot.imsave` with array of dimensions $(100, 100)$

Figure 2.3: (a)-(d) represent how changes in images were made in order to make them suitable for further use. (a)-(c) were saved using `pyplot.savefig`

2.2.2 Performing PCA

With the images stored in correct format, a function called `imread` from the package `matplotlib.image` was used to read the images in an array format, taking values only from the first channel (as discussed above). The $(100, 100)$ was flattened using `numpy.flatten` to get the dimension to $(10000,)$ to stack all 10000 pixel values for further processing. Each image was stored in another 2-D array where the first dimension shows the number of images and second one shows the number of pixels in each image, e.g., (200, 10000) means that the array is storing 200 images with 10000 pixels each.

Our goal is to predict all the pixels of the next few images with great accuracy. In the example above, this would mean predicting image 201, 202, 203 ... and so on. For example, let's say we want to predict the next 5 images, this would mean we have to correctly find the next 50,000 numbers! This is where PCA comes in handy.

On performing PCA on 200 images, it was found that nearly 99% of the variance was captured by the first three PCs, as shown in the Fig. 2.4. In this case, variance captured by PC1, PC2 and PC3 are 75.7132684%,, 20.2868834% and 3.18736412% respectively. The PCA was performed by using `sklearn.decomposition` package in python.

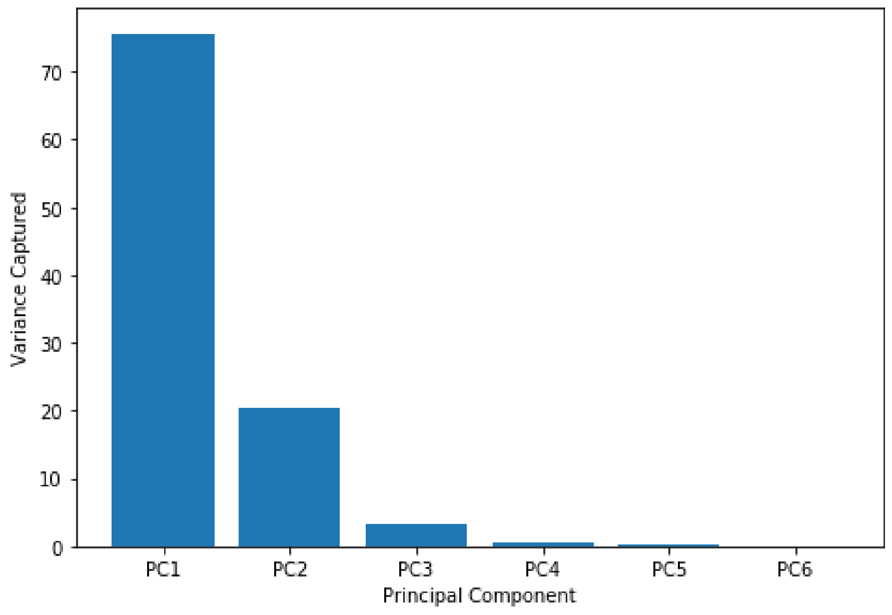


Figure 2.4: Variance captured by PCs.

So, by using PCA, we can represent the data in $(200, 10000)$ array by a $(200, 3)$ array and a transformation matrix. And instead of predicting 50,000 numbers for next 5 images, we just have to predict 15 numbers, 3 for each image. Since each image represents the microstructure after 100 timesteps, we can jump 500 timesteps in a much lesser time.

2.3 Predicting microstructures

Over the course of the project, multiple machine learning techniques were used, such as regression, Convoluted Neural Networks (CNN) and Long Short Term Memory (LSTM) networks, to predict the PCs for next 20-30 images. These techniques were time consuming and the results were sub-par (Fig.2.5). The reason I worked with these models was coherent with some other ideas I wanted to explore, which will not be discussed in this thesis.

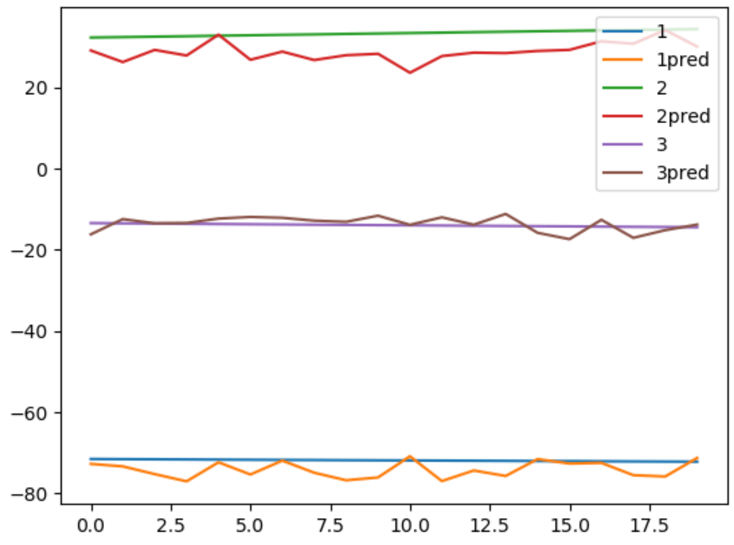


Figure 2.5: Predicting PCs of next 20 images using CNN (1, 2 and 3 are actual PC of images by "normal" model and 1pred, 2pred and 3pred are their respective predictions)

I switched focus to just predicting the PCs of the images and for that, a simple interpolation function in python which could extrapolate next few values of a curve worked quite well. The package `scipy.interpolate` was used for predictions in the hybrid model. The predicted PCs were then inverse transformed to get the predicted images. It was done by reshaping the $(10000,)$ array from `PCA.inverse_transform` function to $(100, 100)$ array which gives the value of c.

The Fig. 2.6, shows variation of PCs with timesteps and from the curves, it becomes clear why extrapolation of few more PCAs can be used to predict more images.

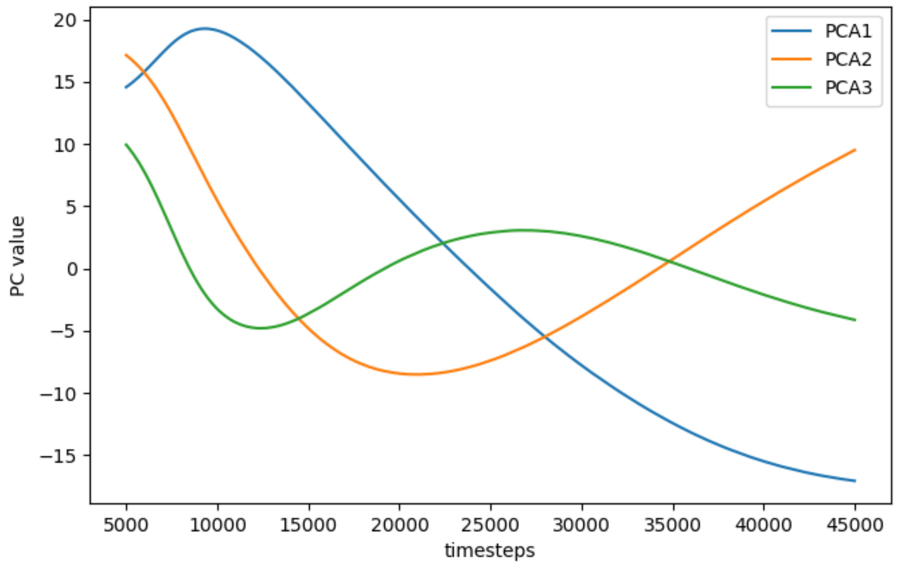


Figure 2.6: Variation of PCs with timesteps. The images were taken at every 100 timesteps.

For the hybrid model, new sets of variables were introduced which are explained below -

- `n_pred` is the number of images (or sets of PCs, (PC1, PC2, PC3) for each image) predicted and `t_pred` is the number of timesteps after which predictions are made. For example, to predict 20 images after every 5000 timesteps, `n_pred=20` and `t_pred = 5000`.
- `times_pred` is the number of times predictions are made in the model.
- *Predicted timesteps* are the timesteps where the predicted images 'replace' the images in normal model.

Algorithm of the hybrid model -

1. Solve the Cahn-Hilliard equation till at least $n = 15000$ saving image every 100^{th} step from $n = 5000$ such that 100 images are saved to do PCA on.
2. Perform PCA on the 100 images and predict next `n_pred` sets of PCs.
3. Save the images at *predicted timesteps* by performing inverse transformation.

4. Read the 20th image as an array and set it equal to the concentration c at the appropriate timestep, i.e., last *predicted timestep*.

5. Solve Cahn-Hilliard equation with the above c value for next `t_pred` timesteps.

6. Transform all saved images in previous step using PCA and predict the next `n_pred` images.

7. Repeat steps 4-7 until the desired timestep is reached.

An example code till the first prediction is shown below -

```python
for t in range(15000):
    d_free_energy_c = d_free_energy(c)
    laplacian_c = laplacian(c)
    laplacian_mu = laplacian(mu)
    for i in range(mesh_x-1):
        mu[i] = d_free_energy_c[i] - 2.0*K*laplacian_c[i]
        dc_dt[i] = M*laplacian_mu[i]
        c[i] = c[i] + dc_dt[i]*dt
        c[i] = impose_PBC(c)[i]
    if t % 100 == 0 and t >= 5000:
        plt.axis('off')
        plt.imsave("out{0}_{1}.png".format(pct, t), c, cmap='
   binary_r', vmin = 0, vmax = 1)

j = 0
train_imgs = np.zeros((450, 10000))
pca_imgs = np.zeros((450,3))

for i in range(5000, 15000, 100):
  test = imread('out40_{0}.png'.format(i))
  test = test[:, :, 0]
  train_imgs[j] = np.ndarray.flatten(test)
  j = j+1



train_imgs[:100] = np.asfarray(train_imgs[:100])

pca1 = PCA(n_components = 3, svd_solver='full')
```

```
pca_imgs[:100] = pca1.fit_transform(train_imgs[:100])


y_actual1 = pca_imgs[:100,0]
y_actual2 = pca_imgs[:100,1]
y_actual3 = pca_imgs[:100,2]

z = np.linspace(1, 100, num = 100)
z_pred = np.linspace(101, 120, num = 20)

f_1 = interp1d(z, y_actual1, fill_value = 'extrapolate')
f_2 = interp1d(z, y_actual2, fill_value = 'extrapolate')
f_3 = interp1d(z, y_actual3, fill_value = 'extrapolate')

f_pred1 = f_1(z_pred)
f_pred2 = f_2(z_pred)
f_pred3 = f_3(z_pred)

t=t+1
for i in range(n_pred):
    pred_img = np.array((f_pred1[i], f_pred2[i], f_pred3[i]))
    pred_img = pca1.inverse_transform(pred_img)
    pred_img = pred_img.reshape(-1,100)
    c = pred_img
    plt.axis('off')
    plt.imsave("out{0}_{1}.png".format(pct, t + 100*i), c, cmap='
    binary_r', vmin = 0, vmax = 1)
    pred_img = np.ndarray.flatten(pred_img)
    train_imgs[100+i] = pred_img
    j = j + 1

t = t + 100*(n_pred-1)
print(t)

del z, z_pred, f_1, f_2, f_3, f_pred1, f_pred2, f_pred3, y_actual1,
    y_actual2, y_actual3
###
```

To predict the images through time evolution of PCs, **all** saved images were used in testing phase of the model. It was found that the time evolution depends on the local time history rather than all previous history. The prediction results by using all the previous images (Fig. 2.7) and by only using the images from step 4 of the algorithm, i.e., in `t_pred` steps (Fig. 2.8) are compared below.

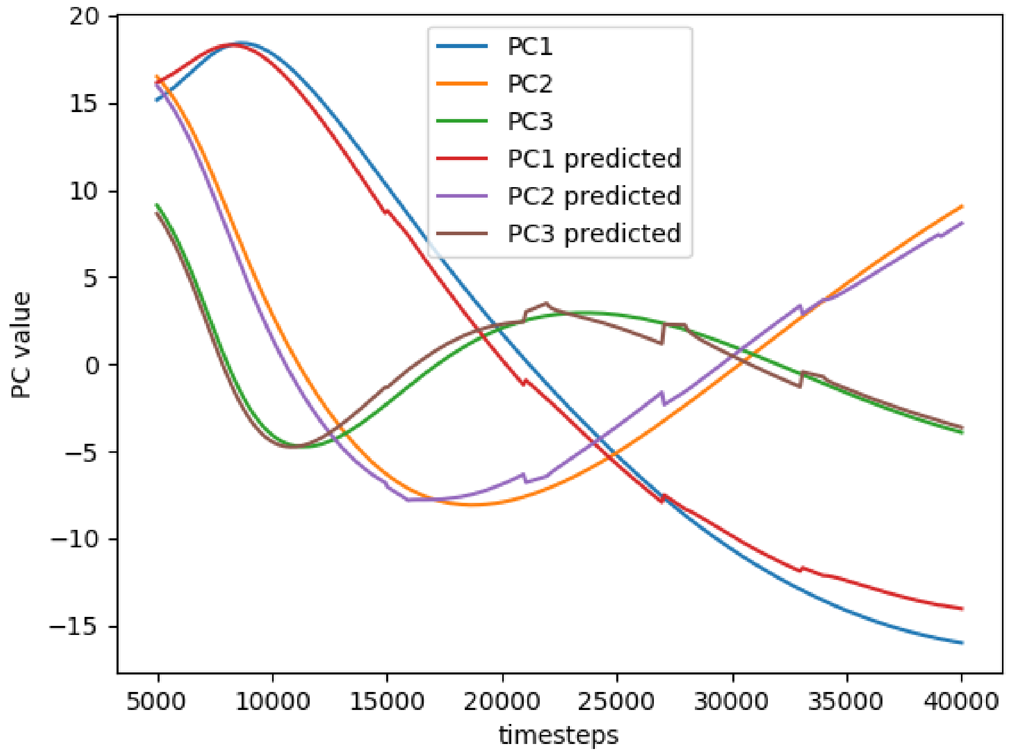


Figure 2.7: Predictions using all previous images with $c_0 = 0.4$, `n_pred` $= 10$, `t_pred`$=50$ and `times_pred` $= 5$

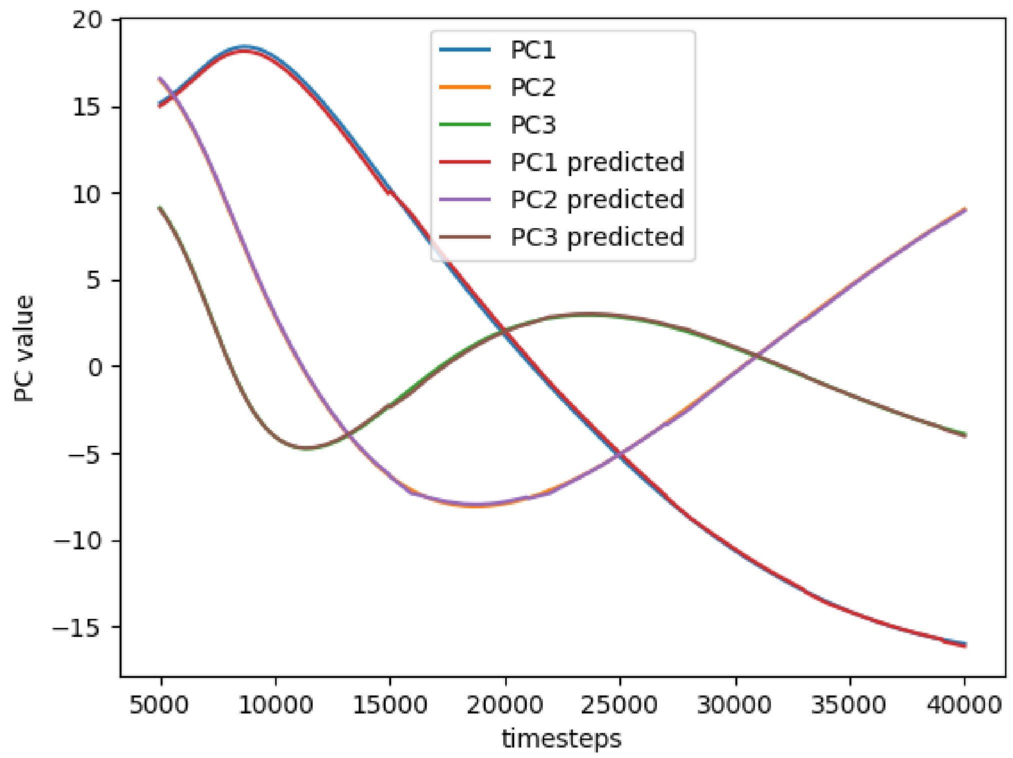


Figure 2.8: Predictions using last `t_pred` images with $c_0 = 0.4$, `n_pred` = 10, `t_pred`=50 and `times_pred` = 5

In both figures above, predictions were made 5 times for the next 10000 timesteps. Prediction sets were right after these timesteps - 14900, 20900, 26900, 32900 and 38900. The results in Fig. 2.8 indicated that more images can be predicted in each set, i.e., in this case 20 next images predicted using 50 images instead of only 10 predicted. The results with `n_pred`=20 and `t_pred`=50 are shown in Fig. 2.9.

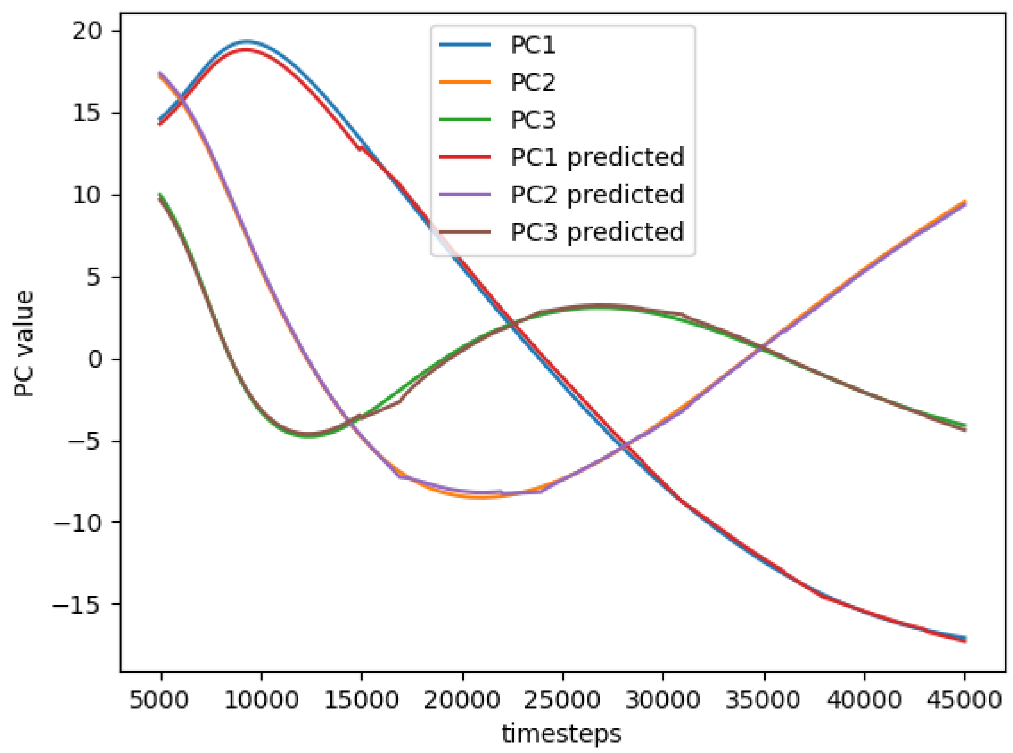


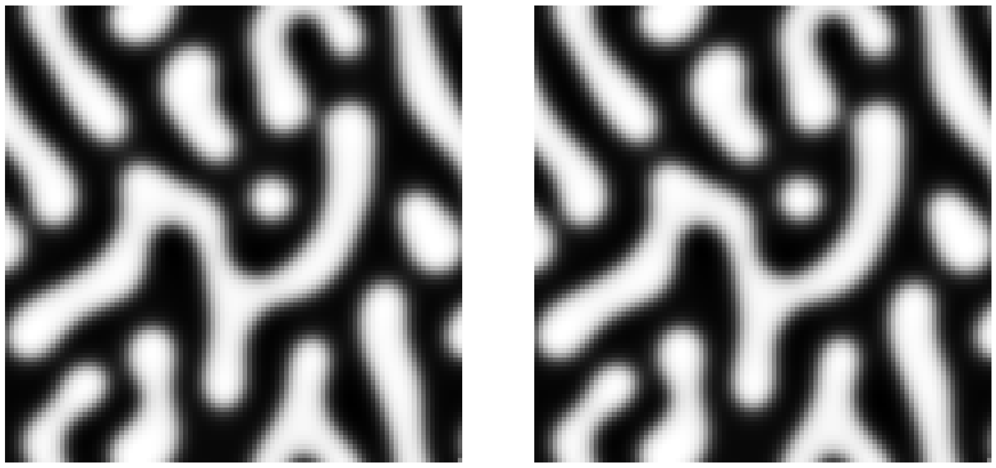
Figure 2.9: Predictions using last `t_pred` images with $c_0 = 0.4$, `n_pred` $= 20$, `t_pred`$=50$ and `times_pred` $= 5$

In Fig. 2.9, the simulation was much faster to reach 45000 timesteps, when compared to the normal thermodynamical model. Normal model took 93min 19.770s whereas the hybrid model only took 72min 24.980s to reach the same timestep which is 20min 54.79s faster.

Chapter 3

Results and Discussion

The hybrid model was compared to the thermodynamical model several times based on their run time on the same machine to ensure same computational power for both. Parameters of hybrid model, mainly the values `n_pred` and `t_pred`, were changed to optimise the model.



(a) Image from thermodynamical model

(b) Image from hybrid model with `n_pred`=10 and `t_pred`=50

Figure 3.1: Visual comparison between images from thermodynamical model and the hybrid model at timestep = 39900 and $c_0 = 40$

In the hybrid model used for 3.1(b), variance captured by the 3 PCs and the comparison between weights of PCs in Fig. 3.1(a) and (b) is shown in tables below -

Principal Component	Variance captured (in %)
PC1	71.696766
PC2	21.707210
PC3	4.419871
Total	97.823847

Table 3.1: Variance captured by 3 PCs

Principal Component	Thermodynamic Model	Hybrid Model
PC1	-15.9829503	-16.14033459
PC2	9.04661943	8.96257997
PC3	-3.91159133	-4.02082259

Table 3.2: Comparison between weights of PCs of images in Fig. 3.1

To get least difference between the weights of PCs, make sure that the hybrid model reaches the required (or final) timestep in step 5 of the algorithm. This ensures that the final microstructure comes out to from the thermodynamically well grounded part of the model. As an example, instead of comparing weights of PCs in images from $n = 39900$, wights from timestep $n = 38900$ (which is before the next prediction step) were compared in Table 3.3.

Principal Component	Thermodynamic Model	Hybrid Model
PC1	-15.73876213	-15.72338542
PC2	8.22508817	8.17324703
PC3	-3.53304386	-3.52758436

Table 3.3: Comparison between weights of PCs of images at timestep $n = 38900$

Results comparing the time taken to run a simulation in thermodynamic model and hybrid model for $c_0 = 0.4$ are compared below -

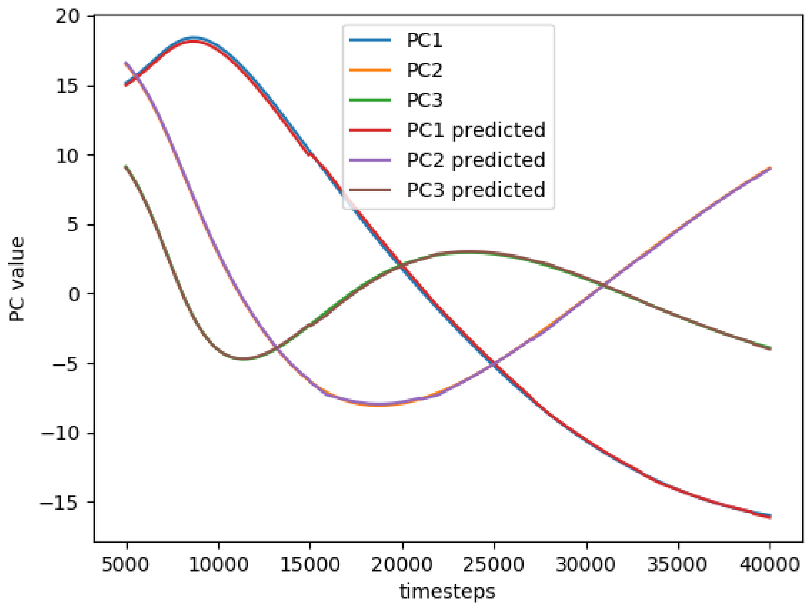


Figure 3.2: Time taken for thermodynamic model - 83min 22.142s
Time taken for hybrid model - 76min 17.265s
Hybrid model parameters - `n_pred` = 10, `t_pred`=50 and `times_pred` = 5

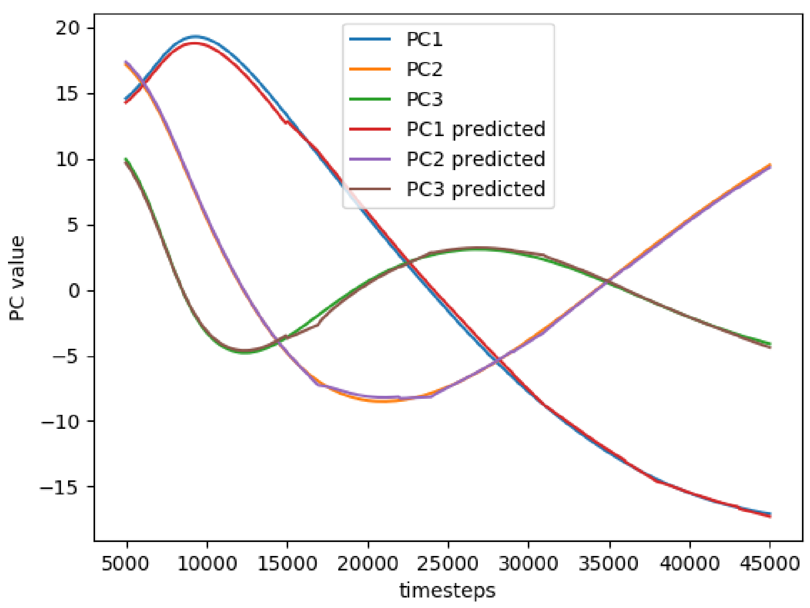


Figure 3.3: Time taken for thermodynamic model - 93min 19.770s
Time taken for hybrid model - 72min 24.980s
Hybrid model parameters - `n_pred` = 20, `t_pred`=50 and `times_pred` = 5

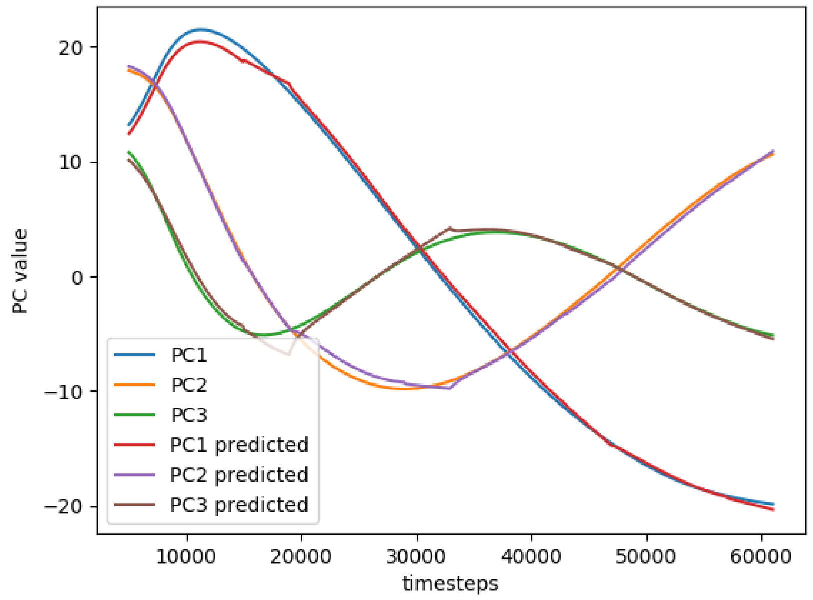


Figure 3.4: Time taken for thermodynamic model - 125min 38.264s
Time taken for hybrid model - 94min 32.146s
Hybrid model parameters - `n_pred` = 40, `t_pred`=100 and `times_pred` = 5

Fig 3.5 and Fig 3.6 show how the PC vary for $c_0 = 35$ and $c_0 = 30$. Hybrid model parameters were `n_pred` = 20, `t_pred`=50 and `times_pred` = 5 for both.

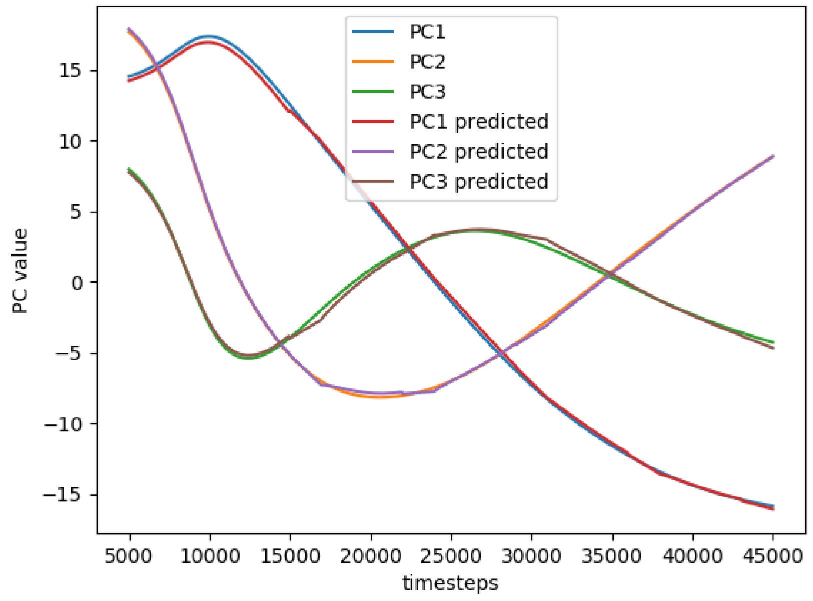


Figure 3.5: Time taken for thermodynamic model, $c_0 = 35$ - 94min 56.024s
Time taken for hybrid model, $c_0 = 35$ - 74min 16.263s

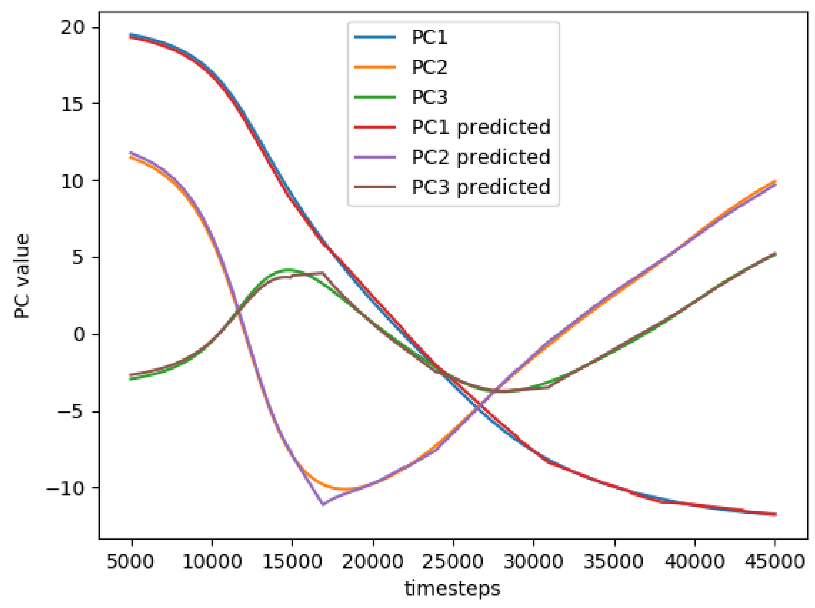


Figure 3.6: Time taken for thermodynamic model, $c_0 = 30$ - 93min 43.767s
Time taken for hybrid model, $c_0 = 30$ - 75min 13.828s

From figures 3.2-3.4, we can say that `n_pred` = 20, `t_pred`=50 (predicting 20 images after 50) is optimal for timesteps around 40000.

The predictions near the curvatures are generally bad due to the linear nature of extrapolation function. In the limited time, higher order polynomial extrapolation could not be optimised. Guessing the type of function and its adjusting parameters by studying the variation of PCs in large number of simulations could be done to make the model perform even better.

Chapter 4

Conclusions, Work planned and Future scope

4.1 Conclusions

- The current phase-field model for solving spinodal decomposition in a binary alloy is based on solving the Cahn-Hilliard equation numerically, updating the composition at every point in the mesh grid at each timestep.
- It is a slow process to solve the differential equations and can take several hours to run.
- In this work, a hybrid model which can fasten up the simulation process by a significant amount of time using data-driven techniques has been developed and studied.
- Several comparisons between both models in terms of time taken were made. It was shown that without losing much accuracy in terms of the development of the microstructure, the speed of the model increased by about 20%.

4.2 Work Planned and Future Work

Due to the COVID-19 pandemic, the workflow of the project was disrupted for a long time. The current code of hybrid model has the timesteps values hard-coded inside. Instead of having to manually change the numbers for timesteps, it was planned to make a better, modular and elegant code once the hybrid model started to show appreciable results.

The predictions part of the current code uses very simple extrapolation technique which has great room for improvement. Variations of PCs from multiple sets of microstructures with different initial composition, thermodynamic values, etc., can be studied to make a better guess of interpolation function whose parameters can be modified to make better predictions.

The machine learning part of the project had to be completely dropped and remains unexplored. Machine learning algorithms can help to look for the similarities and patterns in microstructures generated from real-life experiments.

References

[1] W. J. Boettinger, J. A. Warren, C. Beckermann and A. Karma, *Phase-field simulation of solidification*, 2002. DOI: `10.1146/annurev.matsci.32.101901.155803` (cit. on p. 2).

[2] J. W. Cahn, 'On spinodal decomposition', *Acta Metallurgica*, 1961, ISSN: 00016160. DOI: `10.1016/0001-6160(61)90182-1` (cit. on p. 2).

[3] J. W. Cahn and J. E. Hilliard, 'Free energy of a nonuniform system. III. Nucleation in a two-component incompressible fluid', *The Journal of Chemical Physics*, 1959, ISSN: 00219606. DOI: `10.1063/1.1730447` (cit. on p. 3).

[4] A. Choudhury, Y. C. Yabansu, S. R. Kalidindi and A. Dennstedt, 'Quantification and classification of microstructures in ternary eutectic alloys using 2-point spatial correlations and principal component analyses', *Acta Materialia*, 2016, ISSN: 13596454. DOI: `10.1016/j.actamat.2016.03.010` (cit. on p. 4).

[5] K. P. F.R.S., 'Liii. on lines and planes of closest fit to systems of points in space', *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. DOI: `10.1080/14786440109462720` (cit. on p. 4).

[6] H. Hotelling, 'Analysis of a complex of statistical variables into Principal Components. Jour. Educ. Psych., 24, 417-441, 498-520', *The Journal of Educational Psychology*, 1933 (cit. on p. 4).

[7] N. Provatas and K. Elder, *Phase-Field Methods in Materials Science and Engineering*. 2010, ISBN: 9783527407477. DOI: `10.1002/9783527631520` (cit. on p. 3).