MATHEMATICAL MODELLING OF COMPLEX DYNAMICS

by

Sohail Ahmed Memon

A Thesis submitted in partial fulfilment for the requirements for the degree of Doctor of Philosophy at the University of Central Lancashire

April 2017

University of Central Lancashire

I declare that while registered as a candidate for the research degree, I have not been a registered candidate or enrolled student for another award of the University or other academic or professional institution.

I declare that no material contained in the thesis has been used in any other submission for an academic award and is solely my own work.

Signature of Candidate	[illegible]
Type of Award	**Doctor of Philosophy**
School	**Physical Sciences and Computing**

Abstract

Soft materials have a wide range of applications, which include the production of masks for nano–lithography, the separation of membranes with nano–pores, and the preparation of nano–size structures for electronic devices. Self–organization in soft matter is a primary mechanism for the formation of structure. Block copolymers are long chain molecules composed of several different polymer blocks covalently bonded into a single macromolecule, which belong to an important class of soft materials which can self–assemble into different nano–structures due to their natural ability to microphase separate. Experimental and theoretical studies of block copolymers are quite challenging and, without computer simulations, it is difficult and problematic to analyse modern experiments. The Cell Dynamics Simulation (CDS) technique is a fast and accurate computational technique, which has been used to investigate block copolymers.

The stability has been analysed by making use of different discrete Laplacian operators using well–chosen time steps in CDS. This analysis offers stability conditions for phase–field, based on the Cahn–Hilliard Cook (CHC) equations of which CDS is the finite difference approximation. To overcome grid related artefacts (discretization errors) in the computational grid, the study has been done for employing an isotropic Laplacian operator in the CDS framework. Several 2D and 3D discrete Laplacians have been quantitatively compared for their isotropy. The novel 2D 9–point BV(D2Q9) isotropic stencil operators have been derived from the B.A.C. van Vlimmeren method and their isotropy measure has been determined optimally better than other exiting 2D 9–point discrete Laplacian operators. Overall, the stencils in 9–point family Laplacians in 2D and the 19–point stencil operators in 3D have been found to be optimal in terms of isotropy and time step stability.

Considerable implementation of Laplacians with good isotropy has played an important role in achieving a proper structure factor in modelling methods of block copolymers.

The novel models have been developed by implementing CDS via more stable implicit methods, including backward Euler, Crank–Nicolson (CN) and Alternating Direction Implicit (ADI) methods. The CN scheme were implemented for both one order and two order parameter systems in CDS and successful results were obtained compared to forward Euler method. Due to the implementation of implicit methods, the CDS has achieved second–order accuracy both in time and space and it has become stronger, robust and more stable technique for simulation of the phase–separation phenomena in soft materials.

Contents

Abstract i

List of Figures vi

List of Tables x

Acknowledgements xi

Terminology xii

Chapter One 1

1 Introduction 1

1.1 Motivation of the study 2

1.2 Aim and Objectives 4

1.3 Original contributions in the thesis 5

1.4 Outline of thesis 7

Chapter Two 9

2 Literature review 9

2.1 Overview of Mathematical Contributions in Soft Materials 9

2.2 Block Copolymers and their Applications 13

2.3 Applications of Cell Dynamics Simulation (CDS) Method 15

2.4 Two Order Parameter Systems 16

2.5 Importance of Isotropic Laplacian Operators 18

2.6 Finite Difference Methods 22

2.6.1 Forward Time, Centred Space 23

2.6.2 Backward Time, Centred Space 25

2.6.3 Crank–Nicolson Scheme 26

2.6.4 Alternating Direction Implicit Method 30

Chapter Three 36

3 Method and Model Equations (One order parameter) 36

3.1 Cell dynamics simulation method 36

3.2 Laplacian schemes 39

3.2.1 Two dimensional Laplacian schemes 41

3.2.2 Three–dimensional Laplacian schemes 54

3.3 Stability analysis 62

3.4 Conclusions 64

Chapter Four 66

4 Simulation Results (One–Order Parameter) 66

4.1 Two–dimensional simulations 67

4.1.1 2D Simulation results based on anisotropic Laplacian schemes 72

4.1.2 2D Simulation results based on isotropic Laplacian schemes 77
4.2 Three–dimensional simulations 79
4.3 Conclusions 85
Chapter Five 87
5 Implementation of the Crank–Nicolson method for CDS equations 87
5.1 Implementation of the Crank–Nicolson (CN) Scheme in cell dynamics 87
5.1.1 Matrix based forward Euler and backward Euler methods for CDS 88
5.1.2 Crank–Nicolson method for CDS 95
5.1.3 Implementation of boundary conditions 98
5.2 Conclusions 106
Chapter Six 108
6 The Cell Dynamics Simulations of Two Order Parameter Systems 108
6.1 Mathematical model of two order parameter systems 108
6.2 Numerical method for two order parameter systems 112
6.2.1 Computer simulations 113
6.3 Three-dimensional simulations of *A*–*B*/*C* systems 117
6.4 Implementation of Crank–Nicolson scheme in *A*–*B*/*C* systems 121
6.5 Conclusions 130
Chapter Seven 132
7 Implementation of the Alternating Direction Implicit method for CDS equations 132
7.1 Implementation of the ADI method for CDS 132
7.1.1 Generalized ADI method 134
7.1.2 Hundsdorfer’s ADI method 143
7.2 Conclusions 146
Chapter Eight 148
8 Conclusions and Future Works 148
8.1 Conclusions 148
8.2 Future works 151
References 154
Appendix A 162
2D CDS Code (Default) 162
Appendix B 170
CDS Code for 9–point Star Laplacain Scheme 170
Appendix C 171
CDS Code for 17–point Laplacian Scheme 171
Appendix D 173

3D CDS Code (Default) 173
Appendix E 183
Conjugate Gradient Method 183
Appendix F 185
CDS CN Code 185
Appendix G 196
Two Order Parameter Systems CDS Code 196

List of Figures

Figure 2.1: Energy shells form stencils on a cubic unit shell. This image is taken from [6]. 20
Figure 2.2: Three different schemes for a two-dimensional heat equation 29
Figure 3.1: The 9–point stencil shape of Laplacian on 2D grid. The dark circle is the centre, the square boxes are its nearest neighbours (NN) and the crosses are its next nearest neighbours (NNN). 42
Figure 3.2: The value of a measure of the isotropy $d(r)$ for the actual Laplacian (red dashed line for $-k^2$) and other discrete Laplacians with r the radius. 46
Figure 3.3: The value of a measure of the isotropy $d(r)$ for the actual Laplacian (red solid line for $-k^2$) and other discrete Laplacians against the radius *r*. The BV(*D2Q9*) are newly derived stencils.. 51
Figure 3.4: Stencil shapes of Laplacian on 2D grid. a) 9–point star stencil; b) 17–point stencil.52
Figure 3.5: The value of a measure of the isotropy $d(r)$ for the actual Laplacian (red solid line for $-k^2$) and other discrete Laplacians with r the radius. 53
Figure 3.6: The stencil for three-dimensional Laplacian schemes, where NN, NNN, and NNNN are the nearest neighbours, next–nearest neighbours and next–next nearest neighbours to the point r. This image is taken with the permission of [2].. 55
Figure 3.7: The value of a measure of the isotropy $d(r)$ for the actual Laplacian (red solid line for $-k^2$) and other discrete Laplacians with r the radius. 59
Figure 3.8: The value of a measure of the isotropy $d(r)$ for the actual Laplacian (red solid line for $-k^2$) and other discrete Laplacians with r the radius. 60
Figure 4.1: Results of CDS based on OP(D2Q9) 9–point stencil, equation (3.59). Real space simulation snapshots in (a), (b) and (c) are for 100th, 10000th and 100000th time steps respectively obtained by using parameters given in Table 4.2. Real space simulation snapshot in (d) is for 100000th time step obtained by using parameters given in Table 4.3. 68
Figure 4.2: CDS results based on Laplacian scheme A(D2Q5); a) real space simulation snapshot at100000th time step by using parameters given in Table 4.2; b) real space simulation snapshot of binary blend at 100000th time step by using parameters given in Table 4.3.......................... 72
Figure 4.3: CDS results based on Laplacian scheme D2Q5 given in equation (3.57); a) real space simulation snapshot at100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3.. 73
Figure 4.4: CDS results based on Laplacian scheme D2Q9 given in equation (3.60). The real space simulation snapshot at 100th time step obtained by using parameters given in Table 4.2. 73
Figure 4.5: CDS results based on Laplacian scheme BK(D2Q9) given in equation (3.61); a) real space simulation snapshot at100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3.. 74
Figure 4.6: CDS results based on Laplacian scheme D2Q9$_{star}$; a) real space simulation snapshot at100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3...................... 75
Figure 4.7: CDS results based on Laplacian scheme D2Q17 given in equation (3.66); a) real space simulation snapshot at100000th time step obtained by using parameters given in Table 4.2;

b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3 76
Figure 4.8: CDS results based on Laplacian scheme PK(D2Q9) given in equation (3.58); a) real space simulation snapshot at100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3 77
Figure 4.9: CDS results based on Laplacian scheme BV(D2Q9)$_{case1}$ given in equation (3.62); a) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3 78
Figure 4.10: CDS results based on Laplacian scheme BV(D2Q9)$_{case2}$ given in equation (3.63); a) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3 79
Figure 4.11: CDS results based on Laplacian scheme BV(D2Q9)$_{case3}$ given in equation (3.64); a) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3 79
Figure 4.12: CDS results based on Laplacian scheme SO(D3Q27); a) real space simulation snapshot at 100th time step; b) real space simulation snapshot at 10000th time step; c) real space simulation snapshot at 100000th time step 82
Figure 4.13: 3D CDS results based on Laplacian schemes D3Q7 and D3Q15. a) real space simulation snapshot at 100000th time step using Laplacian scheme D3Q7; b) real space simulation snapshot at 100000th time step using Laplacian scheme D3Q15. 83
Figure 4.14: 3D results of CDS in spherical morphology for A-B diblock copolymer systems. a) Real space simulation snapshot at 100000th time step obtained using Laplacian scheme D3Q19. b) Real space simulation snapshot at 100000th time step obtained using Laplacian scheme D3Q27. c) Real space simulation snapshot at 100000th time step obtained using Laplacian scheme PK(D3Q27). d) Real space simulation snapshot at 100000th time step obtained using Laplacian scheme BV(D3Q27). 84
Figure 5.1: Forward Euler method for two-dimensional CDS equations based on 5–point formula at different time steps. a) at t = 10; b) t = 100; c) t = 1000. 92
Figure 5.2: Backward Euler method for two-dimensional CDS equations based on 5– point formula at different time steps. a) at t = 10; b) t = 100; c) t = 1000. 95
Figure 5.3: CN scheme for two-dimensional CDS equations based on 5–point formula at different time steps. a) at t = 10; b) t = 100; c) t = 1000. 97
Figure 5.4: Explicit forward Euler method based on 5–point formula using periodic boundary conditions where (a) and (b) images are 100th and 10000th time steps 100
Figure 5.5: Implicit backward Euler method based on 5–point formula using periodic boundary conditions where images (a) and (b) are 100th and 10000th time steps. 100
Figure 5.6: CN method based on 5–point formula using periodic boundary conditions where images (a) and (b) are 100th and 10000th time steps respectively. 101
Figure 5.7: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 100th and 10000th time steps in images (a) and (b) respectively. Two lines of different colour show numerical values of $\psi(x)$ for two different methods, the forward Euler and the CN. The numerical values plotted here were obtained from the simulations shown in Figure 5.4 and 5.6 102

Figure 5.8: Explicit forward Euler method based on 9–point formula using periodic boundary conditions where (a) and (b) images are 100th and 10000th time steps respectively. 104
Figure 5.9: Implicit backward Euler method based on 9–point formula using periodic boundary conditions where images (a) and (b) are 100th and 10000th time steps respectively. 104
Figure 5.10: CN method based on 9–point formula using periodic boundary conditions where images (a) and (b) are 100th and 10000th time steps respectively.. 104
Figure 5.11: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 100th and 10000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler, backward Euler and the CN methods. The numerical values plotted here were obtained from the simulations shown in Figure 5.8, 5.9 and 5.10. ... 105
Figure 6.1: The diblock copolymer and homopolymer (A–B/C) systems. Images (a), (b) and (c) are 1000th, 4000th and 50000th time steps respectively.. 114
Figure 6.2: Hexagonal microdomains in A– B/C systems. Images (a) and (b) are 10000th and 50000th time steps respectively.. 116
Figure 6.3: The simulation images (a), (b), (c) and (d) are 1000th, 4000th, 10000th and 70000th time steps respectively. 117
Figure 6.4: The simulation images (a), (b), and (c) are 1000th, 10000th, and 50000th time steps respectively using Laplacian scheme SO(D3Q27)... 119
Figure 6.5: The simulation images (a) and (b) were obtained at 50000th time step using Laplacian schemes D3Q19 and BV(D3Q27) respectively.. 120
Figure 6.6: The images (a) and (b) are 1000th and 10000th time steps respectively obtained using the backward Euler method for two–order parameter systems. 125
Figure 6.7: The numerical values of order parameter $\phi(x,y)$ are plotted against the space (0 – 128) for 100th and 10000th time steps in images (a) and (b) respectively. Numerical values of $\phi(x,y)$ are shown for the forward Euler and backward Euler methods. The numerical values plotted here were obtained from the simulations shown in Figure 6.3 and 6.6........................ 126
Figure 6.8: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 100th and 10000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler and backward Euler methods. The numerical values plotted here were obtained from the simulations shown in Figure 6.3 and 6.6........................ 126
Figure 6.9: The images (a) and (b) are 10000th and 70000th time steps respectively obtained using the CN method for two–order parameter systems. 127
Figure 6.10: The numerical values of order parameter $\phi(x,y)$ are plotted against the space (0 – 128) for 100th and 70000th time steps in images (a) and (b) respectively. Numerical values of $\phi(x,y)$ are shown for the forward Euler and CN methods. The numerical values plotted here were obtained from the simulations shown in Figure 6.3 and 6.9. 128
Figure 6.11: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 100th and 70000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler and CN methods. The numerical values plotted here were obtained from the simulations shown in Figure 6.3 and 6.9. 128
Figure 7.1: Explicit forward Euler method based on 5–point formula (Laplacian scheme A(D2Q5)) using periodic boundary conditions where images (a) and (b) are at 1000th and 100000th time steps respectively. 136

Figure 7.2: Generalized ADI method based on 5–point formula (Laplacian scheme A(D2Q5)) using periodic boundary conditions where images (a) and (b) are at 1000th and 100000th time steps respectively. 137
Figure 7.3: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 1000th and 100000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler and generalized ADI methods. The numerical values plotted here were obtained from the simulations shown in Figure 7.1 and 7.2........ 137
Figure 7.4: Generalized ADI method based on 5–point formula using 256×256 grid size and time interval $\Delta t = 1.0$ where images (a) and (b) are at 1000th and 100000th time steps respectively. 139
Figure 7.5: Three-dimensional results using explicit method based on 7–point formula using $64 \times 64 \times 64$ grid size and time interval $\Delta t = 0.1$ where images (a) and (b) are at 1000th and 10000th time steps respectively. 141
Figure 7.6: Three–dimensional results using generalized ADI method based on 7–point formula using $64 \times 64 \times 64$ grid size and time interval $\Delta t = 1.0$ where images (a) and (b) are at 1000th and 10000th time steps respectively........ 141
Figure 7.7: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 1000th and 100000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler and generalized ADI methods. The numerical values plotted here were obtained from the 3D simulations shown in Figure 7.5 and 7.6........ 142
Figure 7.8: Hundsdorfer's ADI method based on 5–point formula (Laplacian scheme A) using periodic boundary conditions where images (a) and (b) at 1000th and 100000th time steps respectively. 145
Figure 7.9: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 1000th and 100000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler and Hundsdorfer's ADI methods. The numerical values plotted here were obtained from the simulations shown in Figure 7.1 and 7.8. 145

List of Tables

Table 3.1: Time step (Δt) values for all two– and three–dimensional Laplacian schemes obtained from stability analysis criteria 64
Table 4.1: Simulation parameters used for the different morphologies 67
Table 4.2: System parameters used in cell dynamical method for Lamellae morphology.......... 69
Table 4.3: System parameters used in cell dynamical method for binary blend 69
Table 4.4: 2D stencils along with their weights for utilisation in computer code and isotropic or anisotropic status 71
Table 4.5: 3D stencils along with their weights for utilisation in computer code and isotropic or anisotropic status. 80
Table 5.1: System parameters used in cell dynamical method for lamellae morphology 91
Table 7.1: System parameters used in cell dynamical method for lamellae morphology 136

Acknowledgements

I am indebted and grateful to many people for the support, encouragement, advice and friendship that they have offered during my studies at the University of Central Lancashire. It is a great pleasure to acknowledge them here.

I would like to first express my sincere gratitude to my supervisors, Dr Dung Ly and Professor Waqar Ahmed for their vital guidance, patience and encouragement over the past years. Many thanks for your confidence in me and your enthusiasm.

I would also like to thank Professor Andrei Zvelindovsky and Dr Marco Pinna for their help. Thanks go to all my colleagues and friends for their motivation and help, and for reminding me that to have fun occasionally is fine.

Finally, a huge and special thanks to my family for their support and love. This journey would have not been possible without their support. I dedicate this thesis to my sisters, brothers, wife and daughter.

Terminology

Nomenclature

f	Global volume fraction of *A* monomers in the diblock
N_A	Number of *A* monomers
N_B	Number of *B* monomers
$g(\psi)$	Map function
D	Positive constant for diffusion coefficient
F	Free energy functional
$\langle\langle X \rangle\rangle$	Laplacian on quantity *X*
K	Phenomenological mobility constant
$\mathbf{k}$	Wave factor
$F[\psi]$	Free energy functional
B	Chain length dependence
Δt	Time interval
M	Matrix

Greek Letters

ϕ_A	Local volume fraction
ϕ_B	Local volume fraction
ψ	Order parameter
ϕ	Order parameter
Δ	Differential operator

Abbreviations

MD	Molecular Dynamics
MC	Monte Carlo
DPD	Dissipative Particle Dynamics
BD	Brownian Dynamics
TDGL	Time-dependent Ginzburg-Landau
SCFT	Self-consistent Field Theory
CDS	Cell Dynamic Simulation
A–B/C	Mixture of *A*-*B* diblock copolymer and *C* homopolymer
BCPs	Block Copolymer Systems
CHC	Cahn-Hilliard Cook
PDEs	Partial Differential Equations
MesoDyn	Mesoscopic Dynamics
CN	Crank–Nicolson
ADI	Alternative Direction Implicit
FDT	Fluctuation Dissipation Theorem
FDTD	Finite Difference Time Domain
FD	Finite Difference
PDO	Partial Differential Operator
BTCS	Backward Time, Centred Space
FTCS	Forward Time, Centred Space
CG	Conjugate Gradient
D2Q5	Two–dimensional 5–point stencil
D2Q9	Two–dimensional 9–point stencil
D3Q19	Three–dimensional 19–point stencil
D3Q27	Three–dimensional 27–point stencil

Chapter One

1 Introduction

The studies of block copolymers in soft materials have been considered theoretically and experimentally over the last few decades and have attracted the attention of both material scientists and mathematicians. Block copolymers are materials that, due to their intrinsic property of microphase separation, can self–assemble into different nanostructures on a scale in the range of 10–100 nm. These structures are lamellae, spheres, packed cylinders and gyroids [1, 2]. Block copolymers are useful due to their ability to form regular nanometre–scale patterns. Experimental studies of these materials are time consuming, expensive and challenging. Therefore, computer simulation can be used in a computer–aided design to provide new insights into their characteristics; many computer simulation techniques have been designed to study block copolymer systems. A cell dynamics simulation (CDS) technique, based on solving partial differential equations (PDEs), is computationally very fast compared to other simulation methods.

This study focuses on mathematical modelling and computer simulations of diblock copolymers (two blocks per molecule) in lamellar forming systems. In this new contribution, the computer simulations of diblock copolymers were performed by employing CDS in a Cartesian coordinate system. A cellular automaton called the cell dynamics simulation (CDS) method was first proposed by Oono and Puri for the modelling of spinodal decomposition [3]. The CDS technique was used to study block copolymer structures in order to gain molecular information. Block copolymer structures are very sensitive to external influences. In computer simulation it manifests itself in sensitivity via discretization errors. In this study, several discretization schemes were investigated in order to improve CDS performance and a comparison between different discretization schemes was performed. The CDS simulation codes were developed in FORTRAN computer language for the evolution of order parameters.

1.1 Motivation of the study

In this study CDS has been employed because it offers a number of advantages, which include:

- The CDS is a fast method of simple Time Dependent Ginzburg–Landau (TDGL) type simulation in describing the phase separation dynamics in soft systems concerning the diffusive dynamics. For example, it was applied to Block Copolymer (BCP) systems to accurately describe very complex dynamical behaviour in large scale BCP systems [4]
- CDS can be efficiently used to model dynamic processes in block copolymers at the mesoscale level.
- The CDS method is proposed for obtaining numerical results in spatial decomposition considering the neighbouring points to minimize the calculation cost.
- It is a good compromise between computational speed and physical accuracy.
- CDS is less computationally intensive compared to the Self–Consistent Field (SCF) simulation method and it allows wider space parameters and a longer time for evolution [2].
- CDS is considered more efficient than discretised TDGL or Cahn–Hilliard Cook (CHC) equations, due to a much larger (effective) and stable time step.

However, CDS also suffers some drawbacks, including:

- The stability of CDS for larger (effective) time steps is still questionable.

- The finite difference Laplacian schemes (stencil operators) that lead to grid related artefacts (or discretization errors).
- The general implementation of CDS framework in the explicit finite difference method is not very stable or first-order accurate in time.

In order to enhance the CDS performance, the following were addressed:

- The CDS method needs further improvement in terms of stability analysis. It is the larger time step that recognizes the CDS's most efficient method of studying the microphase separation or spinodal decomposition [5]. Therefore, the stability analysis was performed for the CDS framework using different Laplacian schemes.
- The use of an isotropic discrete Laplacian operator in CDS is necessary to avoid grid related artefacts (anisotropies). To implement a strong grid–based simulation methodology, it is crucial to choose Laplacians with good isotropy and scaling behaviour. The achievement of proper structure factors in modelling methods of block copolymers is important and this process requires a considerable implementation of isotropy. Several discrete Laplacians were proposed in the literature [6, 7], especially for the Lattice–Boltzmann method. Here, the most frequent choices of discrete Laplacians will be considered and discussed, giving their properties and isotropic behaviour in detail.
- The general finite difference method applied to CDS has been the forward Euler method (explicit scheme). It is well known that the explicit scheme has some limitations, mainly regarding convergence and stability [8]. The forward Euler method is not very stable or conditionally stable and is first–order accurate in time and second–order accurate in space. Other implicit schemes, i.e. the Crank–Nicolson scheme (CN), must be implemented to replace the conventional

approach of the finite difference method. The implicit schemes are unconditionally convergent and stable (in terms of using time step value) and are second–order accurate in time and space. The Crank–Nicolson (CN) scheme is one of the implicit schemes, having less truncation errors and more stability than the forward Euler method [8]. In this work the CDS equations have been modified with implicit finite difference schemes which include backward Euler, CN and Alternating Direction Implicit (ADI) methods.

1.2 Aim and Objectives

This thesis is aimed at developing the analysis of partial differential equations (PDEs) involved in the CDS method for investigating phase separation in diblock copolymer systems.

The objectives of this research study were as follows:

- To investigate 2D and 3D discrete Laplacian operators for ensuring isotropy and to perform the stability analysis for time step value so that these isotropic discrete Laplacian operators can be employed in CDS for modelling of diblock copolymers.
- To investigate simulation results of *A*–*B* diblock copolymers systems using CDS by employing various 2D and 3D isotropic Laplacian schemes.
- To implement the Crank–Nicolson (CN) method for CDS based on one order parameter system. The CN is a finite difference method which is second-order accurate in time and space. The CN method is slow but more stable compared to the forward Euler method, which is fast but not stable.
- To implement the CN method for CDS based on a two order parameter system. The two order parameter systems are comprised of *A*–*B* diblock copolymer and *C*

homopolymer with incompressibility conditions for two independent variables. One of these variables describes the segregation of copolymers and the other describes order parameter in micro-phase separation. The new methodology for such systems will be helpful and useful to relieve the anisotropy of the system in the late stage of domain growth, which may arise from the discretization of the space.

- To implement the Alternating Direction Implicit (ADI) method for CDS based on one order parameter system. The ADI method is a finite difference method which is second-order accurate in time and space. The ADI is faster than CN. The implementation of the ADI method for CDS framework makes the CDS more stable and robust technique.

1.3 Original contributions in the thesis

The work presented in the thesis makes the following novel contributions:

- Although the two– and three–dimensional discrete Laplacian operators have been studied for their isotropy and scaling behaviour [6, 7, 9], in this study various stencils for Laplacian operators are quantitatively compared to measure their isotropy. Three new two-dimensional 9–point isotropic stencil operators (BV(D2Q9)) are derived from B.A.C. van Vlimmeren's method. The isotropy of these stencils is measured and compared with the existing 9-point isotropic discrete Laplacian OP(D2Q9). One of these stencils has been found to be more isotropic. The stability of the efficient CDS method was analysed by making use of special properties of the discrete Laplacian operator. It was found that the isotropic discrete Laplacians up to fourth–order on **k** become anisotropic for larger wave vectors, and up to second–order 'less isotropic' are slightly anisotropic on the whole **k** range. The two-dimensional 9–point PK(D2Q9) and three-dimensional 19–point (D3Q19) discrete Laplacians were found to be ideally isotropic.

- Computer codes in FORTRAN were developed for CDS based on 2D and 3D Laplacian schemes in order to investigate isotropic simulation results of *A*–*B* diblock copolymer systems.
- Generally, the explicit scheme (forward Euler method) has been implemented for CDS equations. In this work, the novel model of CDS has been developed by implementing implicit finite difference schemes which include backward Euler, Crank–Nicolson and Alternating Direction Implicit (ADI) methods. In CN methodology for CDS, a 9–point isotropic Laplacian operator was successfully employed and numerical results were obtained. The implicit schemes are more stable and second-order accurate in time and space. The implementation of these schemes makes the CDS stronger in terms of stability and consistency.
- To make the CN method more generalized for CDS method. The CN method was implemented for two order parameter systems using CDS model equations and the computer codes were also developed. The 9–point isotropic Laplacian operator was successfully employed and numerical results were obtained. The novel development of CDS for two order parameter systems in CN method helps to relieve the anisotropy of the system in the late stage of domain growth, which may arise from the discretization of the space.

1.4 Outline of thesis

This thesis is divided into eight chapters. **Chapter One** gives a brief introduction.

In **chapter two,** the review of mathematical models and a related literature review are given. In addition, the block copolymers and their applications are discussed, the importance of Laplacian operators is addressed and the finite difference schemes are explained.

In **chapter three,** the cell dynamics simulation model explains the one-order parameter system. The 2D and 3D Laplacian schemes and their properties are investigated. A stability analysis was undertaken for each Laplacian scheme to suggest the suitable time interval value.

In **chapter four**, the simulations are presented for 2D and 3D Laplacian schemes which are discussed in Chapter three. The isotropic results of the lamellar forming of *A*–*B* diblock copolymer for 2D simulations and the isotropic results of spherical morphology in 3D simulations are analysed using Laplacian schemes.

In **chapter five**, the matrix-based forward Euler, backward Euler and Crank–Nicolson methods are implemented for CDS, based on one–order parameter system model equations. Two different Laplacian schemes are employed in a newly developed model using the Crank–Nicolson method for CDS.

In **chapter six**, the cell dynamical model is explained for two–order parameter systems of *A*–*B* diblock copolymer and *C* homopolymer mixtures. The backward Euler and Crank–Nicolson method is implemented for two–order parameter systems.

In **chapter seven**, the Alternating Direction Implicit (ADI) method is implemented for the cell dynamical model of one–order parameter systems. Two different ADI methods are implemented.

In **chapter eight**, the conclusions are given for the findings and future work is elaborated.

Chapter Two

2 Literature review

2.1 Overview of Mathematical Contributions in Soft Materials

Soft materials research is now a highly demanding field which is transforming into a quantitative science. The collaboration between the mathematical sciences and materials sciences is increasing and researchers from both fields are working together and have developed a broad mathematical theory of materials. Materials science has been the main area of research for mathematics professionals. At the same time, the development in materials science has been the focus for industries such as aerospace, automotive, biomaterials, chemicals, electronics, energy, metals, and telecommunications [10, 11]. The important and emerging area in the field of materials science is polymers, where the problems of liquid flow are arising. Today, the field of materials science is vast and covers physical sciences, engineering and mathematics. The synthesis and manufacture of new materials, the modification of materials, and the understanding and prediction of materials and materials properties and their evolution over time, are all basic objectives of materials science. The mathematical sciences play a unique role by giving a quantitative description of the processes and phenomena of materials. The mathematical sciences are helpful in unifying force, revealing underlying structure, and developing computational modelling of processes and phenomena of soft materials. The mathematical methods solve significant problems in materials science and suggest further interesting research areas in mathematical sciences [12].

There are a number of aspects in atomic-scale theories which are modelled by mathematical simulation methods, e.g. the Monte Carlo (MC) [13, 14] simulation method which has been used for the computation of equilibrium atomic configurations and the

Molecular Dynamics (MD) [13] simulation method for the evolution of nonequilibrium atomic configurations at nonzero temperature [15]. In various other phenomena, the evolution of a system is calculated with the specification of external variables involved such as pressure, volume, or temperature etc. The studies of such systems include dislocation formation and motion, plastic flow, grain boundary sliding and strengthening, crack propagation, and chemical reactivity, etc. [16, 17]. The mathematical sciences contribute on a larger scale to the numerical implementation of the systems described above and examples of the mathematical techniques used are fast Fourier transforms, multidimensional integration, curved–space description, solution of nonlinear equations, nonlinear regression, conjugate gradient methods, and eigenvalue methods for large or sparse matrices.

Many mathematical methods still remain poorly understood, though broadly used, in the field of materials science. These include a number of factors that can be carefully considered in the process of approximations; such investigations may be: (i) the stability of methodology, (ii) the types of approximation schemes to be applied to predict the behaviour of particular models, and (iii) guidance to the solution of the integral equations that occur in such systems. While the theory of atomic models has been emerging rapidly during the past few decades [18], there remains much work to be done to understand the complexities in molecular systems. Molecular systems involve several or many atoms (as for polymers) covalently bonded together; they form different geometrical objects and the conventional graph theory best describes such molecular models [19]. As far as the improvement in theory of equilibrium interfaces gain rapid attention, the other research field of interfacial dynamics becomes open to mathematical researchers. The studies of interfacial dynamics are phenomenological and are based on the order parameters [20]. Mathematical researchers are making a grand contribution to this field by developing new algorithms and computational models to obtain the numerical solutions of equations involved in these systems, as materials are produced from phase–separated polymers and the strengths of materials are determined by interfacial strength and morphology. Specifically, investigations have been focused on interfaces in binary blends [21, 22]. In the fields of phase transformations and pattern formations, solidification is an important area which requires a considerable mathematical effort. For instance, it involves a strong interaction between materials scientists, mathematicians, and numerical analysts to develop a new theory and method of calculation for pattern formation during the processes of alloy solidification [23]. An example of such a method is the 'phase–field model', based on a set of phase field equations which are basically time–dependent parabolic partial differential equations. These equations describe the phase transition in which the interface is determined by setting a function called phase or order parameter [23]. Such methods are extended to be applied to the systems of binary alloys. While the model equations describe the formulation of alloy solidification elegantly, many mathematical challenges arise to obtaining these solutions. Some of the mathematical intricacies are highlighted here. The equations which involve small parameters multiplying highest–order derivative pose stiffness; simple finite difference methods with explicit time stepping are not sufficiently accurate and specifically address the stability of the solution with respect to mesh size, orientation, numerical noise and time step. The mechanical performance of materials is highly important for their application. Such applications are airframes of aircraft, automobile structures, the interconnections of microelectronics, and many others. The defects and their responses in many ways affect the mechanical behaviour of materials. Many defects, deformations and fracture problems create curiosities among materials scientists, structural analysts and applied mathematicians to address such problems. The mathematical methods, such as partial differential equations representing phase separation phenomena, singularity analysis, finite element methods and so on, have been applied and developed in this area [12].

The theoretical and experimental studies of spinodal decomposition are now attracting attention of both material scientists and mathematicians. The systems of binary mixtures, such as metal alloys and polymer blends, or diblock copolymers based on one scalar order parameter were simulated in order to recognize the technologically relevant fundamental information about kinetics and morphologies. However, there are number of challenges to face when dealing with the issues in the study of growth of order parameter through domain coarsening or phase ordering dynamics [24], and some of these are explained here. An experiment of phase ordering is a process of hours or days and it is still not certain whether the equilibrium state of the system is achieved; this is because the driving forces for coarsening are minute [25]. The theoretical studies of such processes, which use simple models, focus on limiting behaviour [24]. The use of computational methods provides knowledge for dynamic properties and nonequilibrium morphologies. The mathematical approaches should be able to simulate large volumes with proper statistics, minimize the role of boundary conditions, and choose between efficient methodologies which are responsible for the microscopic time and length scales. Considering all the aspects mentioned above, there is still much work to be done to develop computational models in order to sort out between experimental behaviour and simulation results to gain sufficient molecular information. On the other hand, spinodal decomposition often strives with internal factors such as impurities in alloys [26] and symmetry breaking by confinement in thin films [26, 27]. There are external factors which are applied as parameters to avoid defects and tailor overall structure orientation [4]. All these factors play an important role in developing theoretical studies for model description and the appropriate molecular information. Therefore, the computational study of such systems is still of considerable importance. The partial differential equations (PDEs), such as time–dependent Ginzburgh–Landau (TDGL) or Cahn–Hilliard Cook (CHC) equations, best describe phase ordering kinetics or phase separation processes. Computational models based on finite–difference approximation of these PDEs have expounded several characteristics of phase separation for a whole range of systems [28]. This research work deals with the mathematical models for phase separation in diblock copolymers.

2.2 Block Copolymers and their Applications

Polymers can be defined as a long chain of molecules which is formed by the reaction within the smaller units of molecules called monomers, and for two or more different chemical blocks the polymer may come into formation of a block copolymer [29]. The polymer based structures can be found in melts, blends or solutions in various ranges from nanometre scales to microns, millimetres, or even larger [13]. The chain of molecules that form block copolymers is composed of chemically different polymer blocks covalently bonded into one macromolecule. The formation of block copolymers depends on a number of different chemical blocks in the polymer [29]. The block copolymers have two distinct monomers, *A* and *B* form: *A*–*B* diblock, *A*–*B*–*A* triblock, or multiblock copolymers [30].

Block copolymers have many applications in the field of soft nanotechnology, such as templates for nanoelectronics [31], separation nonporous membranes [32], photonic crystals [33], and catalyst materials [34], nanoparticle synthesis, mechanical flow fields, electric fields, temperature gradients and others [35].

Moreover, the use of block copolymers in applications of electronic devices such as fuel cells, batteries, or optoelectronic devices relies on their existing properties. Their use in applications requires highly ordered and defect-free structures. Although the block copolymers have contributed to various applications ranging from drug delivery to structural materials, these have also made an extensive contribution to thin films. The block copolymer thin films are heavily focussed on because they gain two dimensional patterns at optimum level [2, 36]. Other structures, such as high density hard drives, which may be regarded as the smaller versions of current electronics, can be manufactured by employing polymer nano–domains. These patterned magnetic bits can have better performance rates than current optical lithographically patterned drives [36]. The block copolymers are also widely used in commercial products such as bottle stoppers, jelly candles, outer coverings of optical fibres, and artificial organ technology [37].

The applications of block copolymers are made possible in drug delivery and engineering by further discovering the formation of nanostructures [30]. To understand the complexities of these systems, computer technology is employed, for which mathematical models are developed. The microscopic observation of such systems is presented through computer models which help to study these systems, giving a broader view [4].

The block copolymers are not only investigated as a single aspect, but the studies are also undertaken to understand the phase behaviour of the mixtures or blends made up of block copolymers and other polymers of identical monomeric units (homopolymer) [4, 38]. Polymeric blends or mixtures exhibit the behaviour of phase separation and this behaviour encourages researchers to study associated mechanical, chemical and structural properties; for example, polymeric alloys such as brass formed as a solid phase from combining copper and zinc, and also the microstructured phases as formed in steel by the addition of carbon and other elements to iron.

2.3 Applications of Cell Dynamics Simulation (CDS) Method

Many computer simulation techniques have been developed to investigate block copolymers which include molecular dynamics (MD) [13], Monte Carlo (MC) [13, 14] dissipative particle dynamics (DPD) [13, 39], Brownian dynamics (BD) [40], the Time–Dependent Ginzburg–Landau (TDGL) method (including cell dynamics simulation (CDS)) [4, 13, 41], the self–consistent field theory (SCFT) [42] and the Lattice Boltzmann method [13, 43, 44]. The CDS is a fast method of simple TDGL type simulation method which has been applied to study the evolution of diffusive structure in binary blends of both polymers and alloys [3, 45-48] and diblock copolymers [49]. The modelling of reaction–diffusion systems of the Fischer type for studying chemical reactions and population dynamics has been implemented by using CDS technique [50]. Recently, CDS has been applied to study phase separation in diblock copolymers including additional factors, such as confinement, shear, and electric fields [4]. The use of CDS allows convenience in exploring the phase ordering process for examining various dynamical scaling hypotheses [51, 52]. The mesoscopic structure formation of diblock copolymer systems can best be analysed at large extent by incorporating CDS.

Shinozaki and Oono [3, 47] used CDS for modelling the phase–ordering dynamics of thermodynamically unstable phases. They have explained that for spinodal decomposition many analytical and numerical approaches have been applied. For example, Monte Carlo technique was considered the best numerical simulation technique for the problem of highly non–linear phase separation. To overcome the conventional analytical formulation in terms of partial differential equation they have used computationally efficient CDS. Bahiana and Oono [53] conducted a detailed study using CDS method which found out relatively a close connection between microphase separation of block copolymers and spinodal decomposition. The numerical study via

CDS done by them indicated that for very late time–ordering processes, the hydrodynamic effects are essentially important. Kodama and Doi [54] conducted a computer simulation for 2D system to analyse the structural changes of lamellae of block copolymer consisting of two blocks *A* and *B* under steady shear flow. In this work they have used CDS to solve the time evolution equation. Ren and Hamley [41] describe various applications of this powerful CDS method, e.g. simulation of microemulations, binary blends containing surfactants or hard particles, cross linked polymer blends and simulation of microphase separated structures in BCPs and the kinetics of microphase separation. In another study for shear orientation of lamellar phases in block copolymer BCP, the dynamic density functional method of mesoscopic dynamics (MesoDyn) has been given a little comparison with CDS where the kinetics of order parameter evolution are same. MesoDyn simulations have been considered slower than CDS [55].

The review of applications of CDS method given above present an insight into the importance of this method. The CHC equation and its Euler discretized CDS form have different roots but in the literature a close relation has been shown between them [46] that the CDS is an efficient finite difference approximation of these PDEs.

2.4 Two Order Parameter Systems

The study of phase separation in mixtures or blends of soft materials has been of great interest theoretically and experimentally [56-58]. The phase separation behaviour of systems has been the main focus in the ways of both the macrophase and the microphase separation for the past few decades. The phase separation is invoked in structures such as lamellar, cylindrical hexagonal, spherical and regular three-dimensional bicontinuous structures [59]. Two order parameter systems can be defined as systems where a mixture or a blend contains *A*–*B* diblock copolymer and *C* homopolymer. The phase separation triggers two different ways of macrophase and microphase separations. In the model of the system, one independent variable represents the microphase separation that takes place in *A*–*B* diblock copolymer, and the second independent variable represents the macrophase separation between *A*–*B* diblock copolymer and solvent *C* homopolymer. The two independent variables under the incompressibility condition are taken as $\phi = \phi_A - \phi_B$ and $\psi = \phi_A + \phi_B$, where ϕ_A, ϕ_B and ϕ_C are local volume fractions of monomers. The variable ψ describes the segregation of *A*–*B* diblock copolymer and *C* homopolymer, while ϕ represents the microphase separation in *A*–*B* diblock copolymer [59].

The block copolymers can "microphase separate" to form periodic nanostructures. The nanoscale structures created from block copolymers could potentially be used for creating devices for use in computer memory, nanoscale–templating and nanoscale separations. The 'macrophase separation' takes place between the diblock copolymer and the homopolymer mixture, and then 'microphase separation' takes place between the *A*–*B* diblock copolymer.

In a copolymer molecule, the chain of sequences of *A* and *B* monomers are incompatible with each other. If the two sequences of monomers are chemically connected at a junction point, then the macrophase separation cannot occur. For this reason, the phase separation is on macroscopic scale and microdomains of *A*–rich and *B*–rich regions occur. This incompatibility causes spatial segregation at low temperatures for copolymer melt and in thermal equilibrium these microdomains are regularly arranged [60].

T. Ohta and A. Ito [59] studied the dynamics and domain morphology of phase separation in diblock copolymer–homopolymer mixtures and kinetics of double phase separation. They investigated that there exists incompatibility of monomers because of mutual connectivity and this is the reason that phase separation causes spatially periodic equilibrium structures such as lamellar, cylindrical hexagonal and regular three-dimensional structures. There are two key parameters: the block ratio and the temperature, due to which ordered states change and cause a mesophase state. There is a short repulsive interaction between *A* and *B* monomers of a chain of each copolymer. The same repulsive interaction can be assumed between *C* monomers in homopolymer and *B* monomers in copolymer. It is shown how macrophase separation causes microphase separation for volume fraction and chain lengths and, because of this different domain, patterns appear.

2.5 Importance of Isotropic Laplacian Operators

From a mathematical point of view, the operators are integral or differential. The Laplacian (Δ operator) is the divergence of the gradient (or Nabla, ∇ operator) can be written as:

$$\nabla \cdot (\nabla \psi) = \Delta \psi = \sum_{i=1}^{N} \frac{\partial^2 \psi}{\partial x_i^2}, \qquad \text{for } x_i = \text{x,y,z} \ldots \text{ dimensions.} \tag{2.1}$$

When these operators are applied to specific problems for numerical solutions then these operators are discretized. There are various methods of discretizations where accuracy, stability, consistency and conservation of any condition are kept in full consideration for the method to be applicable. Generally, the discretizations face error terms which are anisotropic (artefacts) in a numerical solution of the PDEs [6, 61, 62]. The phenomenon of block copolymers simulated by the CDS method involves the PDEs, and therefore the mathematical operators such as Laplacian operators come into play. Isotropy plays an important role in discretization of the Laplacian operator. Here an example is presented from digital image processing about the importance of the isotropic Laplacian operator. If the discretized approximations of the Laplacian operator are not isotropic, then the Laplacian operator yields different values for two identical edges oriented at different angles. This produces anisotropy which affects outlining edges [63]. It is, therefore, the isotropic properties of various two– and three–dimensional operators that are thoroughly investigated in this study. These Laplacian schemes are then employed in CDS for numerical results. The original CDS technique uses forward Euler's method for solving PDEs [46]. Sumesh *et al*. [6] have proposed two– and three–dimensional Laplacian schemes for cell–dynamical and hybrid lattice Boltzman simulations. The discretization approaches have been used to overcome anisotropic Laplacians in larger stencils and to preserve isotropy up to leading order error. These schemes are completely based on the finite difference method for spatial discretization. They have presented one isotropic scheme in a 9-point Laplacian for two dimensions and all others for three dimensions. They have given a very good comparison of Laplacian operators by Fourier, transforming those at two different planes. The *DdQn* model is shown in Figure 2.1, where *d* is the number of dimensions and *n* is the number of points which follow the weights of the Laplacian schemes. In Figure 2.1, the ordering points on a cubic unit cell and energy shells *e*1, *e*2, and *e*3 with *e*0 on a cubic grid are shown. Sumesh *et al*. [9] present a numerical method for the solution of nonlinear stochastic partial differential model *H* equations for binary mixtures. These equations cannot have an analytical solution so they are dealt with by numerical methods.

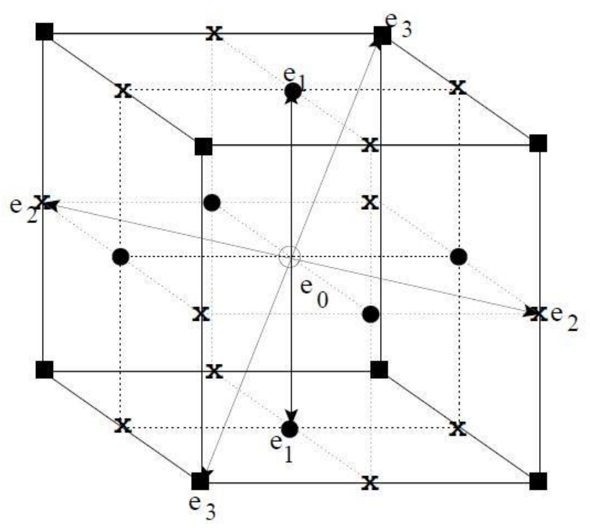


Figure 2.1: Energy shells form stencils on a cubic unit shell. This image is taken from [6].

Numerical methods which are used for discretization of equations of motion on a lattice ensure non–violation of the conservation laws and Fluctuation Dissipation Theorem (FDT). Otherwise, the naive discretization of both the momentum and the order parameter in model *H* can lead to violations of conditions on the lattice. They use two methods: the finite difference method and the finite volume method, for spatial discretization of the order parameter equation. The discretization of two-dimensional 9–point Laplacian operators through finite volume method proved to be less isotropic than Shinozaki and Onoo's Laplacian choice [3, 46]. The Yee algorithm approximates the Laplacian operator through strongly anisotropic 5–point, 9–point and 25–point Laplacian schemes in two dimensions. For implementation of the Yee algorithm, a finite difference time domain method (FDTD) is analysed with respect to the approximation of the two–dimensional Laplacian operator, associated with curl–curl operator. The Yee algorithm is a well–known finite difference time domain (FDTD) approximate to Maxwell's equations. This method assumes a staggered filed arrangement in both space and time, where first order partial derivatives are approximated through second order accurate finite differences. Efforts have been made to improve FDTD schemes. These schemes are based on the transverse Laplacian term associated with curl–curl operator and can be more isotropic, less dispersive and have a higher courant number than the Yee algorithm and its versions [64]. Fei Xiao *et al*. [65] have presented a 2–D isotropic finite difference time domain (FDTD) method, unlike the 1–D finite difference scheme, as the 1–D finite difference scheme approximates the spatial partial differential operator (PDO) in Maxwell's equations. To improve results they adopted the high–dimensional isotropic finite difference scheme in the FDTD to greatly decrease the numerical anisotropy by demonstrating the superiority and applicability of the method. This method was developed with the help of Fourier analysis, which greatly reduced the numerical anisotropy in comparison to that of the conventional Yee-FDTD method. Kumar [66] presented a new method called isotropic finite–difference method in order to preserve isotropy by numerical simulation of partial differential equations (PDEs). Conventional finite–difference methods were pointed out for discretizing PDEs which introduced anisotropy into the numerical scheme; that anisotropy comes from the directional bias of error terms in the discretization. Conclusively, a finite difference scheme is proposed, where the lowest order error terms are without directional bias. The analysis of error terms is necessary in our research study, particularly when we discretize PDEs into numerical simulation. Furthermore, Chow [67] presented discretization of the Laplacian operator on non-hyper–cubical lattices. He explained an implication of the result of an equation that one–shell discretization of the Laplacian operator always has second order errors in any lattice.

2.6 Finite Difference Methods

As in this study the CDS technique is implemented in different finite difference (FD) methods, in this section the FD methods are explained in a very basic way. The model heat equation is presented as an example in the context of numerical approximations for different FD methods.

The finite difference method approximates the solution in a generated grid and uses its schemes for obtaining numerical solutions to ordinary or partial differential equations. The PDEs or ODEs become a linear or non–linear system of algebraic equations. The numerical solution of partial differential equations can be achieved by the finite difference method, which is obtained by replacing the derivatives in the equation by appropriate numerical differentiation formulae [8, 68]. Discrete approximations replace the numerical solutions at some finite number of points in a physical domain. The set of finite points can be collectively known as mesh, where algebraic equations are constructed for discrete approximations, solved or evaluated for discrete unknowns. The fundamental idea of the finite difference method is to replace continuous derivatives with different formulae involving discrete approximations related to positions on mesh; these positions are called nodes. One important thing must be taken into consideration, which is that not all the finite difference approximations provide accurate numerical schemes, but some of them assure stability and convergence in order to satisfy reliable results from these methods [8].

The heat equation is an important partial differential equation which has many applications in different areas. The heat equation has its origins in physics and is studied from a number of perspectives. The form of this equation is commonly known as the diffusion equation. In this work, the two–dimensional heat equation is approximated numerically. The motivation of a heat equation is just thinking of a one-dimensional heated metal rod or thin metal bar of length L. For two-dimensional heat equations, suppose a thin square plate of size $L \times L$ is heated and heat enters or leaves the plate by means of conduction or radiation. It is assumed that the plate is insulated along its top and bottom.

Let:

$u(x, y, t) =$ temperature of the plate at position (x, y) and time t.

The basic differential equation of heat in two–dimensions is given below [8]:

$$\frac{\partial u}{\partial t} = D(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) = D\nabla^2 u, \ 0 \le x \le L, \ 0 \le y \le L \text{ and } 0 \le t \le T \qquad (2.2)$$

where D is a constant coefficient presenting thermal diffusivity and u is the dependent variable, equation (2.2) is a partial differential equation that is satisfied by u , subject to the initial condition:

$$u(x, y, 0) = f(x, y), \ (x, y) \in R \qquad (2.3)$$

where $R = [0, L] \times [0, L]$.

At the edges of the plate, the Dirichlet boundary conditions are imposed and given as follows:

$$\begin{aligned} u(0, y, t) &= u(L, y, t) = 0, \\ u(x, 0, t) &= u(x, L, t) = 0. \end{aligned} \qquad (2.4)$$

2.6.1 Forward Time, Centred Space

The generalization of finite difference formula in forward time, centred space (FTCS) or explicit forward Euler method for a two-dimensional heat equation (2.2) is:

$$u_{j,k}^{n+1} = u_{j,k}^{n} + (r_x \Delta_2^{(x)} + r_y \Delta_2^{(y)}) u_{j,k}^{n}, \tag{2.5}$$

including the following operators:

$$\Delta_2^{(x)} u_{j,k}^{n} = u_{j+1,k}^{n} - 2u_{j,k}^{n} + u_{j-1,k}^{n}, \tag{2.6}$$

$$\Delta_2^{(y)} u_{j,k}^{n} = u_{j,k+1}^{n} - 2u_{j,k}^{n} + u_{j,k-1}^{n}. \tag{2.7}$$

and $r_x = D\Delta t / \Delta x^2$, and $r_y = D\Delta t / \Delta y^2$ [69].

The stability of the Explicit scheme can be taken into consideration such that the discrete Fourier mode of term $u_{j,k}^{n}$ in equation (2.5) is [69]:

$$u_{j,k}^{n} = \xi^n e^{ijp\pi\Delta x} e^{ikq\pi\Delta y}, \tag{2.8}$$

and substituting equation (2.8) into homogenous equation (2.5), and dividing the resulting equation by ξ^{n+1} gives:

$$\xi = 1 - 4r_x \sin^2\left(\frac{p\pi\Delta x}{2}\right) - 4r_y \sin^2\left(\frac{q\pi\Delta y}{2}\right). \tag{2.9}$$

The discrete Neumann criterion $|\xi| \le 1$ implies that $r_x + r_y \le 1/2$.

If the uniform mesh, $\Delta x = \Delta y$ is used, such a condition requires $r_x = r_y \le 1/4$ and restricts to take a twice smaller Δt (or $\sqrt{2}$ – times larger Δx) to ensure stability.

In the FTCS scheme, the time derivative is discretized in forward difference and the Taylor expansion is given below [70, 71]:

$$\frac{\partial u}{\partial t} = \frac{u^{n+1} - u^n}{\Delta t} - \frac{\Delta t}{2}\left(\frac{\partial^2 u}{\partial t^2}\right) + \ldots + O(\Delta t)$$

where the dots show higher order terms. The time derivative has truncation error $O(\Delta t)$. The space derivatives are discretized in central difference so their truncation error is $O(\Delta x^2)+O(\Delta y^2)$. The discretization error of the FTCS scheme is: $O(\Delta t)+O(\Delta x^2)+O(\Delta y^2)$ which shows it is linear in time space and quadratic in step space [69].

2.6.2 Backward Time, Centred Space

The explicit forward Euler method approximates values of terms in n space on the right hand side of *diffusion equation* (2.2) for term in n+1 space, but in backward time, centred space (BTCS) or implicit backward Euler method it is reversed; the values in n+1 terms space are approximated from n space term. The finite difference in backward time, centred space for diffusion equation is given below:

$$u_{j,k}^{n+1}-(r_x\Delta_2^{(x)}+r_y\Delta_2^{(y)})u_{j,k}^{n+1}=u_{j,k}^{n}. \tag{2.10}$$

To approximate equation (2.10) numerically, it needs to bring in *M*x =*b* where *M* is the two dimensional matrix of size $(N_x-1)(N_y-1)\times(N_x-1)(N_y-1)$ and b is a one–dimensional matrix of size $(N_x-1)(N_y-1)$. After algebraic manipulation on equation (2.10) it becomes:

$$\underbrace{(1-r_x\Delta_2^{(x)}-r_y\Delta_2^{(y)})}_{A}\underbrace{u_{j,k}^{n+1}}_{x}=\underbrace{u_{j,k}^{n}}_{b} \tag{2.11}$$

Hence, a linear system must be solved to obtain $u_{j,k}^{n+1}$ from $u_{j,k}^{n}$. However, r_x and r_y are greater than zero, so the matrix *A* is positive definite and strictly diagonally dominant, as well as being *tri*–diagonal [69]. There are several iterative algorithms, i.e. the Crout Factorization method, the SOR algorithm, the LU decomposition method or the

Conjugate Gradient method. The linear system in equation (2.11) takes the form: $x = M^{-1}b$. The inverse of the matrix *M*, which involves the stiffness matrix of the Laplacian operator, is not easy in high dimensions. In the BTCS scheme, the time derivative is discretized in forward difference and space derivatives are discretized in central difference. The discretization error of this scheme is: $O(\Delta t) + O(\Delta x^2) + O(\Delta y^2)$ [69] which is linear in time space and quadratic in step space.

The stability of the implicit forward Euler method can be given as the eigenvalues of M^{-1} are reciprocals of those of *M*, the spectral radius $M^{-1}, \rho(M^{-1}) < 1$. This implies that M^{-1} is convergent. So the method is stable, independent of any choice of r_x or r_y [69]. The amplification factor for two–dimensional equation (2.11) is:

$$\xi = \frac{1}{1 + 4r_x \sin^2(p\pi\Delta x/2) + 4r_y \sin^2(q\pi\Delta y/2)} \tag{2.12}$$

The two–dimensional implicit backward Euler method is absolutely stable, since $|\xi| \le 1$.

2.6.3 Crank–Nicolson Scheme

The Crank–Nicolson (CN) scheme is popular in financial engineering and for approximating the *convection*–diffusion equation. The CN scheme was proposed by Crank and Nicolson [72, 73] and is a FD method which is unconditionally stable and is used to approximate partial differential equations (PDEs) numerically. It is known that both the FTCS and BTCS schemes have a discretization error of order $O(\Delta t + h^2)$, but the CN technique puts its efficiency into achieving second order in time and space $O(\Delta t^2 + h^2)$ [8, 69]. As the second order accuracy in time and space can be achieved through this technique, the CN method is regarded as numerically stable; however, oscillations may still occur. It is an implicit method and is a widely used FD method for the modelling of heat–diffusive problems in one or more dimensions. The CN scheme is made up of a half explicit FTCS step and a half implicit BTCS step, each with time step $\Delta t/2.$ An averaging at the *i*th and (*i*+1)th time step is given for the spatial difference. In other words, it is the average of the forward Euler and backward Euler method with a central difference in space. The derivation of the CN scheme for heat equation (2.2) is constructed in a way that, on the left hand side the prediction of values of *u* is done at $n+1$ time step, which is an implicit approach, and on the right hand side the prediction of values of *u* is done at *n* time step, which is an explicit approach; on the right side all the values are assumed to be known easily. The form of *diffusion equation* (2.2) using the CN scheme is given below:

$$\left(1-\frac{r_x}{2}\Delta_2^{(x)}-\frac{r_y}{2}\Delta_2^{(y)}\right)u_{j,k}^{n+1}=\left(1+\frac{r_x}{2}\Delta_2^{(x)}+\frac{r_y}{2}\Delta_2^{(y)}\right)u_{j,k}^{n}. \qquad (2.13)$$

The CN method can be written in matrices as:

$$M_1u_{i,j}^{n+1}=M_2u_{i,j}^{n}. \qquad (2.14)$$

The non–singular matrices M_1 and M_2 are positive definite and have the size $(N_x-1)(N_y-1)\times(N_x-1)(N_y-1).$ To solve systems of linear equations in equation (2.14), either the LU decomposition or Conjugate Gradient method can be used to obtain $u_{i,j}^{n+1}$ from $u_{i,j}^{n}.$ The discrete Neumann criterion for stability is again derived by Fourier analysis, as for the explicit scheme. The amplification factor for the CN method is [69]:

$$\xi=\frac{1-2r_x\sin^2(p\pi\Delta x/2)-2r_y\sin^2(q\pi\Delta y/2)}{1+2r_x\sin^2(p\pi\Delta x/2)+2r_y\sin^2(q\pi\Delta y/2)} \qquad (2.15)$$

where it is necessary to have $|\xi| \le 1,$ for all non–negative r_x and r_y, so the two–dimensional CN scheme is absolutely stable and has order of convergence $O(\Delta t^2 + h^2)$.

In order to analyze the second order accuracy of CN scheme, each side of equation (2.13) can be rearranged into a partial factored form as follows [74]:

$$\left(1-\frac{r_x}{2}\Delta_2^{(x)}\right)\left(1-\frac{r_y}{2}\Delta_2^{(y)}\right)u_{j,k}^{n+1}=\left(1+\frac{r_x}{2}\Delta_2^{(x)}\right)\left(1+\frac{r_y}{2}\Delta_2^{(y)}\right)u_{j,k}^{n} \tag{2.16}$$

The CN scheme (2.16) of heat equation (2.2) in factored form has been derived from equation (2.13) without the loss of accuracy. Doing some algebraic manipulation on equation (2.16) gives:

$$\left(1-\frac{r_x}{2}\Delta_2^{(x)}-\frac{r_y}{2}\Delta_2^{(y)}+\frac{r_x r_y}{4}\Delta_2^{(x)}\Delta_2^{(y)}\right)u_{j,k}^{n+1}=\left(1+\frac{r_x}{2}\Delta_2^{(x)}+\frac{r_y}{2}\Delta_2^{(y)}+\frac{r_x r_y}{4}\Delta_2^{(x)}\Delta_2^{(y)}\right)u_{j,k}^{n} \tag{2.17}$$

Let $r_x = \Delta t / h^2 = r_y$ and the equation (2.17) can be rewritten as follows:

$$\left(1+\frac{(\Delta t)^2}{4h^4}\Delta_2^{(x)}\Delta_2^{(y)}\right)\left(\frac{u_{j,k}^{n+1}-u_{j,k}^{n}}{\Delta t}\right)=\frac{1}{h^2}\left(\Delta_2^{(x)}+\Delta_2^{(y)}\right)\left(\frac{u_{j,k}^{n+1}+u_{j,k}^{n}}{2}\right). \tag{2.18}$$

The equation (2.18) shows that the discretization error of CN scheme is quadratic in time space as well as quadratic in step space $\left(O(\Delta t^2)+O(\Delta x^2)+O(\Delta y^2)\right)$. The CN scheme shows an obvious difference with implicit BTCS in terms of discretization error order. The BTCS and CN schemes have an almost identical algorithmic approach where the truncation error for the Crank–Nicolson scheme is significantly smaller than the temporal truncation error of the BTCS scheme [69].

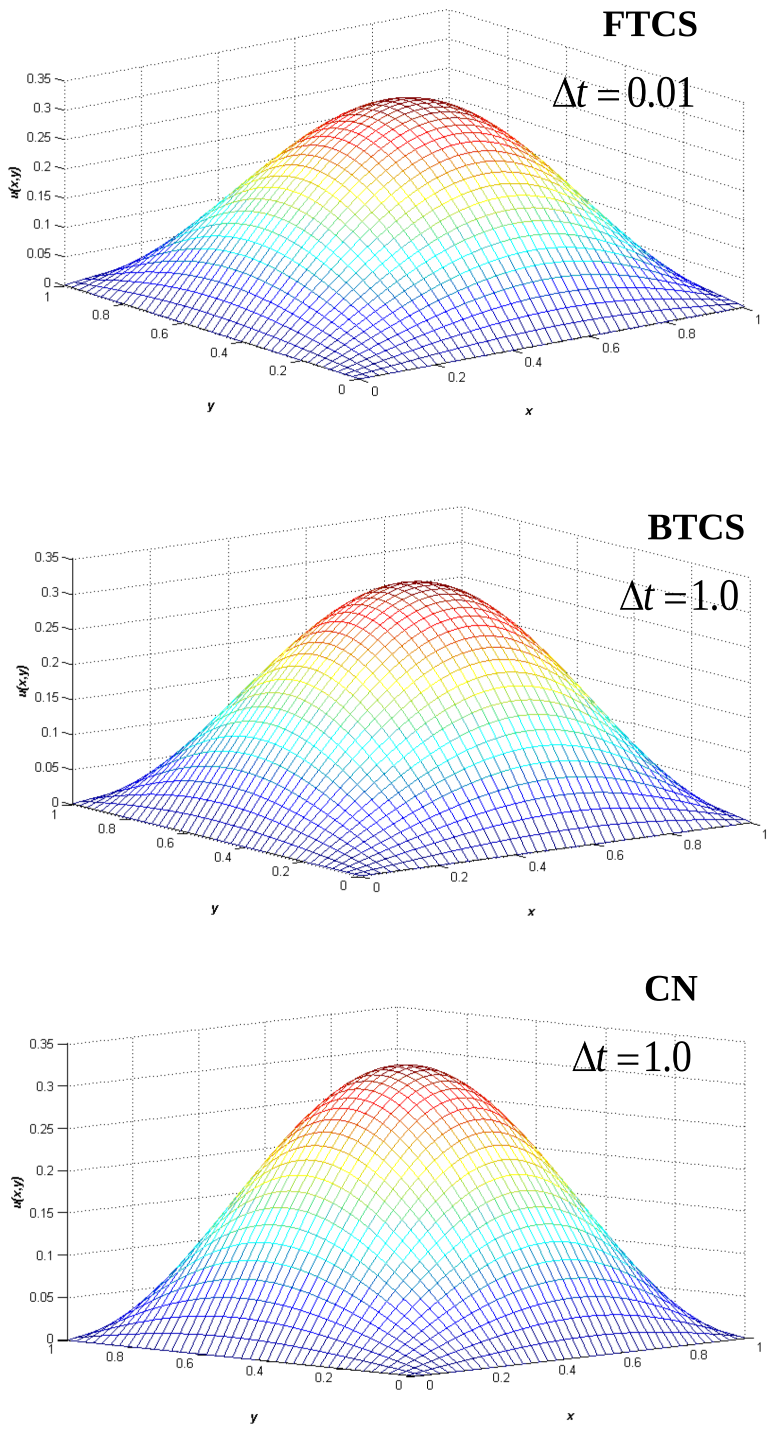


Figure 2.2: Three different schemes for a two-dimensional heat equation

Numerical simulations of heat equation (2.2) are presented by mesh plots in Figure 2.2 for three different methods. These simulatins were executed for 50x50 grid with space steps $\Delta x = \Delta y = 0.2$ up to 100 time steps. The boundary conditions were assigned by zeros and diffusion constant D was chosen 0.25. The time interval Δt was chosen 0.01 for forward Euler method because it is explicit method where time interval can be used conditionally. In Figure 2.2, the numerical solution $u(x,y)$ is shown against the x- and y-coordinates on a grid which was obtained same for three different methods but using different time interval Δt values. The images in Figure 2.2 are showing the state of the heat for the function $f(x, y) = \sin(\pi xy)$ initially assigned. It can be seen in Figure 2.2 that the images of the plots are identical for each scheme, where the explicit FTCS scheme was observed faster in execution compared to implicit schemes. The implicit methods CN and BTCS were found slower but allowed unconditional choice of using time interval Δt value.

2.6.4 Alternating Direction Implicit Method

The Alternating Direction Implicit (ADI) method is a finite difference method introduced by Peaceman and Rachford for the numerical solutions of heat flow equations [75]. The implicit Crank–Nicolson (CN) and backward Euler schemes are using the large matrices. The large matrices become huge, which require sufficient memory for processing and due to the huge matrices these methods work very slowly, especially for two–dimensional higher grids or for three dimensions [76]. The CN scheme is second order accurate in time and space and is unconditionally stable, but requires more operations per time step than the number of unknown variables [77]. To overcome this difficulty and complexity for obtaining numerical results, the operators are used in split format and the idea of using operator splitting is becoming more common day by day. The ADI scheme is based on the same concept of operator splitting [78]. The ADI scheme is computationally efficient and has the same properties as the CN scheme. It is second order accurate in time and space and, unlike the CN scheme, it requires the number of operations per time step that is proportional to the number of unknowns [77]. A variety of stable schemes are available in the ADI method which can be employed for the elliptical or parabolic partial differential equations. It all depends on the type of partial differential equation used; for example, any equation comprised of mixed derivative terms can be handled with a different scheme in the ADI method [79]. The finite difference methods for two–dimensional heat equations (2.1) and related spatial derivatives are discussed in previous sections. Here, the ADI method is elaborated for a two–dimensional heat equation (2.2); this ADI method employs the classic Peaceman and Rachford scheme [75, 76].

$$\left(1-\frac{r_x}{2}\Delta_2^{(x)}\right)u_{j,k}^{n+1/2}=\left(1+\frac{r_y}{2}\Delta_2^{(y)}\right)u_{j,k}^{n}, \tag{2.19}$$

$$\left(1-\frac{r_y}{2}\Delta_2^{(y)}\right)u_{j,k}^{n+1}=\left(1+\frac{r_x}{2}\Delta_2^{(x)}\right)u_{j,k}^{n+1/2}. \tag{2.20}$$

This methodology works in two steps; the first step is given in equation (2.19) in which the first half of the time step Δt the spatial derivative is computed implicitly in coordinate *x* and explicitly in coordinate *y*, and this process is reversed in the second step which is given in equation (2.20). In equation (2.19) the values are approximated for the intermediate solution for $u_{j,k}^{n+1/2}$ and in equation (2.20) the values are approximated for $u_{j,k}^{n+1}$ from the intermediate solution on the right hand side. For the numerical approximation of heat equation (2.2), the Peaceman and Rachford scheme is employed, as this scheme is unconditionally stable and is $O(\Delta t^2)+O(\Delta x^2)+O(\Delta y^2)$ (second–order accurate) [76, 77].

The steps given in equations (2.19) and (2.20) can be written in matrix form:

$$Mu_{j,k}^{n+1/2} = \left(1 + \frac{r_y}{2}\Delta_2^{(y)}\right)u_{j,k}^{n}, \tag{2.21}$$

$$Mu_{j,k}^{n+1} = \left(1 + \frac{r_x}{2}\Delta_2^{(x)}\right)u_{j,k}^{n+1/2}. \tag{2.22}$$

The matrix *M* on the left hand side of equations (2.21) and (2.22) is the tridiagonal matrix which simplifies the calculations, unlike the sparse matrices used in the CN scheme. The matrix *M* for a two–dimensional operator $\Delta_2^{(x)} = u_{j+1,k} - 2u_{j,k} + u_{j-1,k}$ is given as follows:

$$M = \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix}. \tag{2.23}$$

The specific form of the matrix *M* can depend on the type of boundary conditions used: either Dirichlet or periodic. As in the CN scheme, if the grid size is $n \times n$, the matrix becomes the $n^2 \times n^2$ size and the system of linear equations becomes difficult to solve. In ADI, if the grid size is $n \times n$,, the size of the matrix is $(n-1)\times(n-1)$ and is tridiagonal. The beauty of the ADI scheme is that it is using the same matrix *M* for both steps because the split operator is taken over one direction, either *x*–direction or *y*–direction. The operator $\Delta_2^{(y)}$ on the right side of equation (2.21) is not a matrix, but can be treated as the operations of subtraction and addition, simply in the manner of the forward Euler method. If it were a matrix, then the calculations would have been more complicated [77]. In the first step on the right hand side of equation (2.21), the $(M-1)$ equations are solved for the unknown vectors, and for the whole intermediate solution the tridiagonal matrix solves the $(M-1)\times(M-1)$ equations. After the intermediate solution is determined, the same procedure is carried out for the second step, which is given in equation (2.22); in this way the discretization for equation (2.2) approximates values by the so–called ADI method. To solve the linear system of equations, the LU decomposition or the Conjugate Gradient (CG) method can be used, as in the CN method. The Thomas algorithm is efficient for solving linear system of equations in the ADI method [80].

The stability of the ADI scheme is proved in Ref. [77]. To analyse the stability of the ADI scheme employed in equations (2.19) and (2.20) for heat equation (2.2), it can be written in one equation. If the value of $u_{j,k}^{n+1/2}$ from equation (2.19) is substituted into equation (2.20), then one gets the following:

$$\left(1-\frac{r_x}{2}\Delta_2^{(x)}\right)\left(1-\frac{r_y}{2}\Delta_2^{(y)}\right)u_{j,k}^{n+1}=\left(1+\frac{r_x}{2}\Delta_2^{(x)}\right)\left(1+\frac{r_y}{2}\Delta_2^{(y)}\right)u_{j,k}^{n}. \tag{2.24}$$

In discrete von Neumann stability analysis, the Fourier mode of term $u_{j,k}^{n}$ in equation (2.24) is [76, 77]:

$$u_{j,k}^{n}=\xi^{n}e^{ijp\pi\Delta x}e^{ikq\pi\Delta y}, \tag{2.25}$$

$$u_{j,k}^{n+1}=\xi.\xi^{n}e^{ijp\pi\Delta x}e^{ikq\pi\Delta y}. \tag{2.26}$$

Substituting the Fourier modes given in equations (2.25) and (2.26) in equation (2.24), the resulting equation becomes:

$$\xi=\frac{\left(1-2r_x\,sin^2\left(p\pi\Delta x/2\right)\right)\left(1-2r_y\,sin^2\left(q\pi\Delta y/2\right)\right)}{\left(1+2r_x\,sin^2\left(p\pi\Delta x/2\right)\right)\left(1+2r_y\,sin^2\left(q\pi\Delta y/2\right)\right)}. \tag{2.27}$$

The two–dimensional ADI scheme is stable since $|\xi|\le 1$.

The consistency of the ADI method is also discussed here. Adding two equations (2.19) and (2.20) together, the resulting equation becomes:

$$u_{j,k}^{n+1} - u_{j,k}^{n} = r_x \Delta_2^{(x)} u_{j,k}^{n+1/2} + \frac{r_y}{2} \Delta_2^{(y)} (u_{j,k}^{n+1} + u_{j,k}^{n}); \tag{2.28}$$

and if equation (2.19) is subtracted from equation (2.20) the following equation is obtained:

$$u_{j,k}^{n+1/2} = \frac{u_{j,k}^{n+1} + u_{j,k}^{n}}{2} - \frac{r_y}{2} \Delta_2^{(y)} \left(\frac{u_{j,k}^{n+1} - u_{j,k}^{n}}{2} \right). \tag{2.29}$$

By substituting equation (2.29) into equation (2.28) and $r_x = r_y = \Delta t / h^2$ gives:

$$\left(1 + \frac{(\Delta t)^2}{4h^4} \Delta_2^{(x)} \Delta_2^{(y)} \right) \left(\frac{u_{j,k}^{n+1} - u_{j,k}^{n}}{\Delta t} \right) = \frac{1}{h^2} \left(\Delta_2^{(x)} + \Delta_2^{(y)} \right) \left(\frac{u_{j,k}^{n+1} + u_{j,k}^{n}}{2} \right). \tag{2.30}$$

The equation (2.30) can be compared with equation (2.18) which shows that the discretization error of ADI scheme for heat equation (2.2) is second order accurate in both space and time, i.e., $O\left((\Delta t)^2 + h^2\right)$. The discretization error of time step is quadratic ($O(\Delta t)^2$) in equation (2.30), which makes this second–order accurate in time. It must be noted that the FTCS and BTCS methods have discretization error of $O\left((\Delta t) + h^2\right)$, which shows that the order error ($O(\Delta t)$) is not quadratic. Equation (2.18) is derived to show that the CN scheme is second order accurate in time and space and in the same way equation (2.30) shows that the ADI scheme is second order accurate in time and space.

So far, the two–dimensional ADI method employing the Peaceman and Rachford scheme is discussed, which is fast and its stability is the same as that of the CN method. The three–dimensional Peaceman and Rachford scheme is not unconditionally stable [77]. An alternative ADI method was proposed for the two- and three-dimensional heat equation, which is unconditionally stable [81-83]. This method is well known as the Douglas method or the Douglas–Gunn method [82] but it is not explained here in this chapter. The Douglas–Gunn method is mentioned here to highlight that there are various unconditional stable schemes available which can be employed in the ADI method for two- and three–dimensional PDEs. Extensive work on ADI methods has been undertaken for linear second and fourth order parabolic problems [84, 85].

The ADI method has been very helpful in numerical simulations of problems in fluid dynamics [86-88]. Eres *et al*. [89] present a three–dimensional numerical approximation for the mathematical model based on the flow of drying paint films on horizontal substrates. For the numerical implementation, they employ an implicit method by condemning the explicit method for the small time step, and mention that the explicit method is computationally inefficient. They also mention that the ADI methods used by Peaceman and Rachford [75] and Douglas [82] are unconditionally stable for heat flow problems, but are not unconditionally stable for the higher order partial differential equations which involve the biharmonic equation. In this study, the ADI method has been implemented for the cell dynamical simulation technique to make the CDS technique more stable and robust. The ADI method implementation for CDS equations is covered in chapter 7.

Chapter Three

3 Method and Model Equations (One order parameter)

In this chapter the cell dynamics simulation (CDS) method is elaborated. In CDS, the original averaging operator does not represent the discrete Laplacian and that the CDS method should not be analysed via the Time–Dependent Ginzburg–Landau (TDGL) equations. Therefore, the formations and properties of two– and three–dimensional Laplacian operators (averaging operators) are discussed in terms of stencil size and their isotropic behaviour, which ensure the stability of CDS results. The analysis of several discrete Laplacians in terms of isotropy has been performed on a more quantitative balance than previously considered [6]. The stability analysis is also undertaken for the Laplacian schemes for investigating their time step value that will be used in simulations based on CDS.

3.1 Cell dynamics simulation method

The CDS technique is used in a number of works in diblock copolymers and binary mixtures, i.e. metal alloys [3, 48, 53, 90]. The CDS is basically a discretization of the set of partial differential equations involved in the block copolymers for the purpose of obtaining numerical results [4]. The efficiency and the stability of the CDS was investigated by Oono and Puri [46]. In this chapter, the CDS model equations are presented for a one order parameter system in a lamellae forming of *A*–*B* diblock copolymer systems. In CDS, the value of an order parameter $\psi(t,i)$ is determined at time *t* in cell *i* of discrete lattice. Suitable choice of the order parameter of an *A*–*B* diblock copolymer is defined as [2]:

$$\psi = \phi_A - \phi_B + (1 - 2f), \tag{3.1}$$

where ϕ_A and ϕ_B are local volume fractions of A and B monomers, and $f = N_A/(N_A + N_B)$ is the volume fraction of the A monomer in the diblock. The evolution of the order parameter in a single cell is given by:

$$\psi(t+1,i) = g(\psi(t,i)) \tag{3.2}$$

where $g(\psi)$ is the so called map function [46, 91]. The cells need to be connected in order to observe the spatial cooperative interactions. The force on the order parameter $\psi(t,i)$ tends to have proportionality to its difference from the average of the order parameters in the neighbourhood cells. Considering the diffusive dynamics for spatial cooperative interaction between the connectivity of cells, the case of non–conserved order parameter for the time evolution order parameter with term of diffusive dynamics becomes [41],

$$\psi(t+1,i) = g(\psi(t,i)) + D\left[\langle\langle\psi(t,i)\rangle\rangle - \psi(t,i)\right] = \Gamma[\psi(t,i)] \tag{3.3}$$

where the sum is given by first term related to chemical potential gradients and second term that considers diffusive dynamics [41]. In equation (3.3), *D* is a positive constant which presents phenomenological diffusion constant, $\langle\langle X\rangle\rangle - X$, which is essentially an isotropized discrete Laplacian. The general definition $\langle\langle X\rangle\rangle$ on a two–dimensional square lattice is given by

$$\langle\langle\psi(t,i)\rangle\rangle = W_1\sum_{NN}\psi(i,t) + W_2\sum_{NNN}\psi(t,i) \tag{3.4}$$

where *NN* and *NNN* represent its nearest neighbours and next–nearest neighbours respectively, and *Ws* are weights, i.e. $W_1 = 1/6$ and $W_2 = 1/12$ [2, 46]. The inclusion of contributions from the surrounding cells is necessary in the case of a conserved order parameter. The isotropic behaviour of the CDS model in the case of a conserved order parameter model may be affected due to an exchange of order parameter values between a cell and its neighbouring cells. Therefore, to maintain isotropy, the net change of the order parameter should be avoided inside the neighbourhood surrounding the centre cell. Thus, for a conserved order parameter, the net gain of order parameter in a particular cell is given by $\Gamma[\psi(t,i)]-\psi(t,i)$ and in this way the discrete model for conserved order parameter of CDS becomes [41, 46]:

$$\psi(t+1,i)=\Gamma[\psi(t,i)]-\langle\langle\Gamma[\psi(t,i)]-\psi(t,i)\rangle\rangle. \tag{3.5}$$

An additional term, $-B\psi(t,i)$, is added to equation (3.5); this term comes from the contribution of long range ordering to the free energy [41] and equation (3.5) becomes:

$$\psi(t+1,i)=\Gamma[\psi(t,i)]-\langle\langle\Gamma[\psi(t,i)]-\psi(t,i)\rangle\rangle-B\psi(t,i). \tag{3.6}$$

For the dynamics of $\psi(t,r)$, the Cahn-Hilliard-Cook (CHC) equation is used which is given as follows [54, 92]:

$$\frac{\partial\psi}{\partial t}=\mathrm{K}\nabla^2\left(\frac{\delta F[\psi]}{\delta\psi}\right), \tag{3.7}$$

where K is a phenomenological mobility constant. This mobility constant is set to unity for the corresponding setting of the timescale for the diffusive process and $F[\psi]$ is the free energy functional and is given as [53]:

$$F[\psi(r)]=\int dr\left[H(\psi)+\frac{D}{2}|\nabla\psi|^2\right]+\frac{B}{2}\int dr\int dr' G(r-r')\psi(r)\psi(r'). \tag{3.8}$$

In equation (3.8), the first and second terms are the short– and long–range interactions respectively, the diffusion coefficient D is a positive constant, and the constant B represents the chain length dependence to the free energy and $H(\psi)$ is given as:

$$H(\psi) = \left[-\frac{\tau}{2} + \frac{A}{2}(1-2f)^2 \right]\psi^2 + \frac{v}{3}(1-2f)\psi^3 + \frac{u}{4}\psi^4, \tag{3.9}$$

where τ is a temperature and *A*, *v*, *u* are phenomenological constants [41]. All of these parameters represent the relation to molecular characteristics. The numerical evolution of equation (3.7) is given by [55, 93]:

$$\psi(t+1,i) = \psi(t,i) - \{\langle\langle \Gamma(t,i) \rangle\rangle - \Gamma(t,i) + B(t,i)\}, \tag{3.10}$$

where $\langle\langle X \rangle\rangle - X$ is an isotropized discrete Laplacian in square coordinates for quantity *X*, $i = (i_x, i_y)$ and

$$\Gamma(t,i) = g(\psi(t,i)) - \psi(t,i) + D[\langle\langle \psi(t,i) \rangle\rangle - \psi(t,i)], \tag{3.11}$$

where the so called map function is given by:

$$g(\psi) = \left[1 + \tau - A(1-2f)^2\right]\psi - v(1-2f)\psi^2 - u\psi. \tag{3.12}$$

3.2 Laplacian schemes

In this section the Laplacian schemes are discussed by elaborating their properties. The two-dimensional and three-dimensional Laplacian operators are discussed with their Fourier transforms. Each Laplacian scheme is incorporated in the form of equation (3.4) for implementing in simulations for obtaining numerical results.

The Laplacian (Δ operator) is the divergence of the gradient (or Nabla, ∇ operator) and can be written as:

$$\nabla \cdot (\nabla \psi) = \Delta \psi = \sum_{i=1}^{N} \frac{\partial^2 \psi}{\partial x_i^2} \quad \text{for } x_i = x, y, z \ldots \textit{ dimensions} \tag{3.13}$$

In Fourier space, the Laplacian operator is given by:

$$-\nabla \cdot \nabla \Leftrightarrow \mathbf{k}^2. \tag{3.14}$$

where **k** is the wave-vector in Fourier series expansion. The minus sign in $-\nabla \cdot \nabla$ makes this operator positive–semidefinite [94, 95]. The Laplacian operator is discretized on the grid by application of finite difference scheme:

$$\Delta = S + O(h^n). \tag{3.15}$$

where S represents a stencil and $O(h^n)$ is the truncation error of the order n due to finite mesh size h.

The mathematical operators have special importance when they describe the physical world around us. The isotropy is another aspect of discrete Laplacian that must be ensured to carry out proper numerical simulations and this aspect is focussed on in this study. The Laplacian operator is rotationally invariant but all its discretized approximations are not isotropic [63]. The term isotropy is used in several scientific disciplines and defines the certain properties of an object of the nature to be identical when quantified from any direction. The term anisotropy is the opposite of the isotropy and the definition of anisotropy is that it is a condition where different properties are obtained in different directions. The direction is the main factor that is contained in the differentiation of both the isotropic and anisotropic phenomena. Generally, the isotropy is a homogeneousness in all orientations and the anisotropy is a situation where properties vary systematically [96, 97].

Many mathematical techniques, e.g. compact schemes, Padé approximations etc., have been implemented to bolster the numerical accuracy in simulations of partial differential equations (PDEs). A numerical scheme is said to be isotropic if it does not have directional preference. Sometimes the use of conventional finite difference approach for discretizing the PDE causes anisotropy in the numerical scheme. The anisotropy in any numerical scheme is produced from directional bias of the error terms in the discretization [66].

The isotropic property is dealt with as a desirable feature in the discretization of the Laplacian operator. If the discretization is not isotropic then numerical values face artefacts (error terms) which result in anisotropy [6]. An isotropic Laplacian abrades the function equally in all directions [94]. The importance of using isotropic Laplacian operators can be known as an example of heat conduction, and the stability criterion of the numerical scheme becomes better by using the isotropic discretization of the Laplacian [66]. Hale [94] has presented computer simulation images of paintings using isotropic and anisotropic Laplacians. The poor approximations to the anisotropic Laplacians produce checkerboard patterns of the images due to the artefacts, and therefore improved approximations (isotropic Laplacians) are suggested to produce clear images.

3.2.1 Two dimensional Laplacian schemes

In this section, the various two-dimensional Laplacians schemes are discussed, including 9–point family stencils and various other stencils. These Laplacian schemes are obtained from a finite difference scheme for a two–dimensional Laplacian operator. The stencil shape for 9–point family Laplacians is given in Figure 3.1. The *DmQn* notation is used where *m* is the number of dimensions and *n* is the number of points to be calculated. The first scheme is obtained considering its nearest neighbours (NN) [63, 98, 99] as shown in equation (3.16), and the second scheme equation (3.17) is obtained considering its next nearest neighbours (NNN) [63, 99].

$$\Delta(\psi)_{A(D2Q5)} = \frac{1}{h^2}\left[\sum_{i=1}^{4}\psi_i^{(1)} - 4\psi^{(0)}\right]. \tag{3.16}$$

$$\Delta(\psi)_{D2Q5} = \frac{1}{2h^2}\left[\sum_{i=1}^{4}\psi_i^{(2)} - 4\psi^{(0)}\right]. \tag{3.17}$$

The subscript *D2Q5* in equations (3.16) and (3.17) represent the 5–point stencil in two dimensions. The letter *A* in the subscript of equation (3.16) is given to distinguish it from equation (3.17), and will be discussed frequently in the text. When both the NN and NNN are considered, a new model, suggested by Tomita [100] in equation (3.18), is presented which gives 9–point stencil operators on the grid:

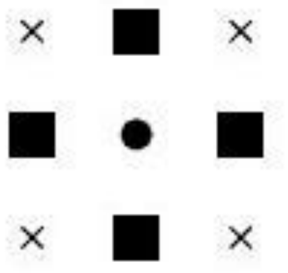


Figure 3.1: The 9–point stencil shape of Laplacian on 2D grid. The dark circle is the centre, the square boxes are its nearest neighbours (NN) and the crosses are its next nearest neighbours (NNN).

$$\Delta(\psi)_{D2Q9} = \frac{1}{1+2\alpha}\left\{\sum_{i=1}^{4}\psi_i^{(1)} + \alpha\sum_{i=1}^{4}\psi_i^{(2)} - 4(1+\alpha)\psi^{(0)}\right\}. \tag{3.18}$$

Varying values of α, with $\alpha = 1/4,$, $\alpha = 1/2,$, $\alpha = 3/4,$ and $\alpha = 1$ give finite difference schemes, which are listed below:

$$\Delta\psi_{PK(D2Q9)} = \frac{1}{6h^2}\left[4\sum_{i=1}^{4}\psi_i^{(1)} + \sum_{i=1}^{4}\psi_i^{(2)} - 20\psi^{(0)}\right]. \tag{3.19}$$

$$\Delta\psi_{OP(D2Q9)} = \frac{1}{2h^2}\left[\sum_{i=1}^{4}\psi_i^{(1)} + \frac{1}{2}\sum_{i=1}^{4}\psi_i^{(2)} - 6\psi^{(0)}\right]. \tag{3.20}$$

$$\Delta\psi_{D2Q9} = \frac{2}{5h^2}\left[\sum_{i=1}^{4}\psi_i^{(1)} + \frac{3}{4}\sum_{i=1}^{4}\psi_i^{(2)} - 7\psi^{(0)}\right]. \tag{3.21}$$

$$\Delta\psi_{BK(D2Q9)} = \frac{1}{3h^2}\left[\sum_{i=1}^{4}\psi_i^{(1)} + \sum_{i=1}^{4}\psi_i^{(2)} - 8\psi^{(0)}\right]. \tag{3.22}$$

The subscripts PK, SO and BK in equations (3.19), (3.20) and (3.22) stand for Patra and Karttunen, Oono and Puri and Behzad Kamgar respectively. Partra and Karttunen [7] have shown that equation (3.16) is anisotropic and equation (3.19) is isotropic. Shinozaki and Oono used equation (3.20) for cell dynamics simulations [3, 47], which is generally considered to be the best choice for isotropy by Tomita [100]. Equation (3.17) is shown to be anisotropic [95] and equation (3.22) is discussed in the literature but is not isotropic [63]. The scheme in equation (3.21) is found to be unstable and is not present in the literature. The isotropy of the discrete Laplacians considered above can best be analysed in Fourier space. The corresponding Fourier transforms of equations (3.16), (3.17) and (3.19–3.22) are given below:

$$\Gamma(k)_{A(D2Q5)} = \frac{2}{(\Delta x)^2}\left[\cos(k_x\Delta x) + \cos(k_y\Delta x) - 2\right]. \tag{3.23}$$

$$\Gamma(k)_{D2Q5} = \frac{1}{(\Delta x)^2}\left[\cos[(k_x + k_y)\Delta x] + \cos[(k_x - k_y)\Delta x] - 2\right]. \tag{3.24}$$

$$\Gamma(k)_{PK(D2Q9)} = \frac{2}{3(\Delta x)^2}\begin{bmatrix} 2\cos(k_x\Delta x) + 2\cos(k_y\Delta x) \\ +\frac{1}{2}\left[\cos[(k_x + k_y)\Delta x] - \cos[(k_x - k_y)]\Delta x\right] - 5 \end{bmatrix}. \tag{3.25}$$

$$\Gamma(k)_{OP(D2Q9)} = \frac{1}{2(\Delta x)^2}\begin{bmatrix} 2\cos(k_x\Delta x) + 2\cos(k_y\Delta x) + \cos[(k_x + k_y)\Delta x] \\ + \cos[(k_x - k_y)\Delta x] - 6 \end{bmatrix}. \tag{3.26}$$

$$\Gamma(k)_{D2Q9} = \frac{2}{5(\Delta x)^2}\begin{bmatrix} 2\cos(k_x\Delta x) + 2\cos(k_y\Delta x) \\ + \frac{3}{2}\left[\cos[(k_x + k_y)\Delta x] + \cos[(k_x - k_y)\Delta x]\right] - 7 \end{bmatrix}. \tag{3.27}$$

$$\Gamma(k)_{BK(D2Q9)} = \frac{1}{3(\Delta x)^2}\begin{bmatrix} 2\cos(k_x\Delta x) + 2\cos(k_y\Delta x) \\ + 2\cos[(k_x + k_y)\Delta x] + 2\cos[(k_x - k_y)\Delta x] - 8 \end{bmatrix}. \tag{3.28}$$

Expanding $\cos(x)$ in the above equations at around $x = 0$ provides:

$$\Gamma(k)_{A(D2Q5)_{NN}} = -(k_x^2 + k_y^2) + \frac{(\Delta x)^2}{12}(k_x^2 + k_y^2)^2 - \frac{(\Delta x)^2}{6}k_x^2 k_y^2 + O(k^6). \tag{3.29}$$

$$\Gamma(k)_{D2Q5_{NNN}} = -(k_x^2 + k_y^2) + \frac{(\Delta x)^2}{12}(k_x^2 + k_y^2)^2 + \frac{(\Delta x)^2}{3}k_x^2 k_y^2 + O(k^6). \tag{3.30}$$

$$\Gamma(k)_{PK(D2Q9)} = -(k_x^2 + k_y^2) + \frac{(\Delta x)^2}{12}(k_x^2 + k_y^2)^2 + O(k^6). \tag{3.31}$$

$$\Gamma(k)_{OP(D2Q9)} = -(k_x^2 + k_y^2) + \frac{(\Delta x)^2}{12}(k_x^2 + k_y^2)^2 + \frac{(\Delta x)^2}{12}k_x^2 k_y^2 + O(k^6). \tag{3.32}$$

$$\Gamma(k)_{D2Q9} = -(k_x^2 + k_y^2) + \frac{(\Delta x)^2}{60}(5k_x^4 + 18k_x^2 k_y^2 + 5k_y^4) + O(k^6). \tag{3.33}$$

$$\Gamma(k)_{BK(D2Q9)} = -(k_x^2 + k_y^2) + \frac{(\Delta x)^2}{12}(k_x^2 + k_y^2)^2 + \frac{(\Delta x)^2}{6}k_x^2 k_y^2 + O(k^6). \tag{3.34}$$

P.I.C. Teixeira and B.M. Mulder have made a little algebraic mistake when Fourier transforming equation (3.16); see equation (6) in Ref. [101]. They have presented the following equation:

$$\Gamma(k)_{A(D2Q5)} = -(k_x^2 + k_y^2) + \frac{(\Delta x)^2}{3}(k_x^2 + k_y^2)^2 - \frac{2(\Delta x)^2}{3}k_x^2 k_y^2 + O(k^6). \tag{3.35}$$

They also made another algebraic mistake for Fourier transforming the triangular stencil operator (3.36). For this operator, in the article they have presented the equation (3.37) after expanding $\cos(x)$ at around $x=0$; see equation (9) in Ref. [101]:

$$\Gamma(k)_{tri} = \frac{2}{3(\Delta x)^2}\left[\begin{array}{l} 2\cos(k_x\Delta x) + 2\cos\left(\frac{1}{2}k_x\Delta x + \sqrt{\frac{3}{2}}k_y\Delta x\right) \\ + 2\cos\left(\frac{1}{2}k_x\Delta x - \sqrt{\frac{3}{2}}k_y\Delta x\right) - 6 \end{array}\right] \tag{3.36}$$

$$= -k_x^2 + k_y^2 - \frac{(\Delta x)^2}{16}\left(k_x^2 + k_y^2\right)^2 + O(k^6). \tag{3.37}$$

After expansion of $\cos(x)$ in operator (3.36), the result is different; the resulting equation is presented below:

$$= -(k_x^2 + 2k_y^2) + \frac{(\Delta x)^2}{16}\left(k_x^2 + 2k_y^2\right)^2 + O(k^6). \tag{3.38}$$

The difference can be seen easily by comparing equation (3.29) with equation (3.35) and equation (3.37) with equation (3.38).

All equations from (3.29) to (3.34) are second order in *k* where equations (3.31) and (3.32) are isotropic and equation (3.33) has anisotropic term which can be observed in its second term.

To compare the properties of these two-dimensional discrete Laplacians in terms of isotropy, a measure of isotropy for the discrete Laplacians has been performed. In Figure 3.2, an analysis of isotropy and anisotropy of the discrete Laplacains has been shown after introducing cylindrical coordinates (r,ϕ) in reciprocal k space by plotting:

$$d(r) = \max_{\phi} \Gamma(r,\phi) - \min_{\phi} \Gamma(r,\phi). \tag{3.39}$$

Equation (3.39) calculates the difference between the maximum and minimum value of $\Gamma(k)$ for radius r. A clear difference between anisotropic and isotropic discrete Laplacians is shown in Figure 3.2. The isotropy measure *d(r)* with maximum deviation from radius r shows the maximum anisotropic behaviour (or maximum anisotropy) in comparison with the analytical expression $-k^2$, which does not show any deviation from radius r because it is considered to have maximum isotropy. All the discrete Laplacians A($D2Q5$), $D2Q5$, BK($D2Q9$) with solid lines seem to be anisotropic because of a noticeable deviation. The other two 9–point discrete Laplacians (PK($D2Q9$) and OP($D2Q9$)) with dotted lines, which are generally considered to be isotropic, seem to have less deviation from the radius r. The discrete Laplacian PK($D2Q9$) with dotted lines (light black colour) is isotropic for $r < 2$, but for larger r, it becomes anisotropic. The stencil OP($D2Q9$) of Oono–Puri is less isotropic than the PK($D2Q9$) for $1.5 < r < 2.5$, but more isotropic for $r > 2.5$. Surprisingly, the isotropy of OP($D2Q9$) is even better than that of PK($D2Q9$).

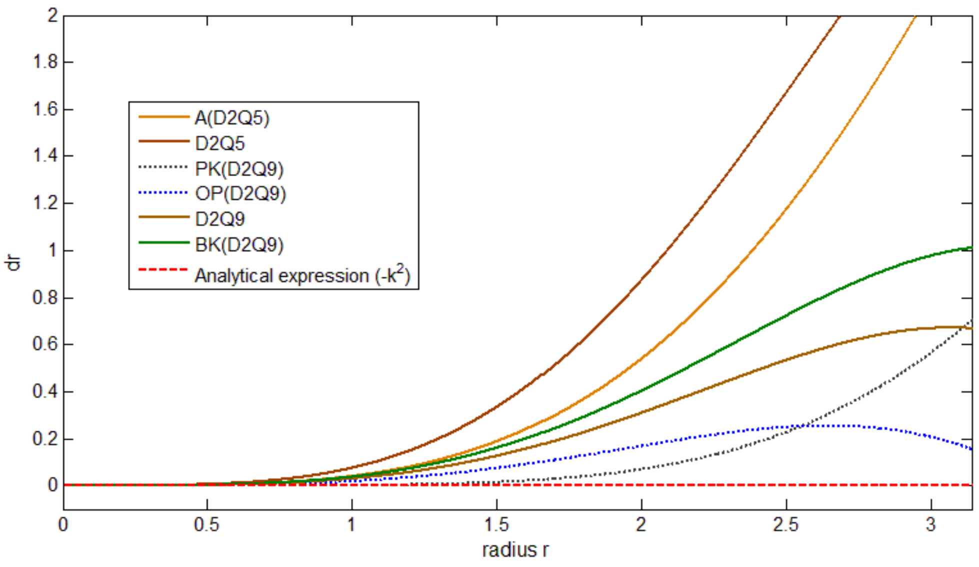


Figure 3.2: The value of a measure of the isotropy $d(r)$ for the actual Laplacian (red dashed line for $-k^2$) and other discrete Laplacians with r the radius.

There has been extensive research work on the investigation of isotropic stencil operators in two and three dimensions for complex dynamics systems. B.A.C. van Vlimmeren [102] has proposed a unique method to calculate the weights of a three–dimensional 27–point stencil operator. Fraaije and his co–workers [61, 103] re–evaluated the basic numerical aspects of the standard lattice models and calculated the weights of a 27–point isotropic stencil operator using the method of B.A.C. van Vlimmeren on the lattice to represent the linkage operator efficiently and accurately. Their work confirmed that isotropy and scaling conditions can be considered to calculate the values for the weights. To investigate the best isotropic stencil operators in 9–point stencil in this study, this method is applied in 2D. The following method calculates the values for the weights d_α in a 9–point isotropic stencil. The half point difference scheme is such that:

$$D_\alpha(f)(x) = \sum_{\alpha=1}^{m=9} d_\alpha \frac{f(x+\frac{h}{2}r_\alpha) - f(x-\frac{h}{2}r_\alpha)}{\|hr_\alpha\|}. \tag{3.40}$$

In Fourier space the discrete half point derivative operator D_α in direction α is:

$$D_\alpha = \frac{2i}{\|hr_\alpha\|}\sin\left(\frac{hkr_\alpha}{2}\right), \tag{3.41}$$

where r_α is a lattice direction in positive half–space:

$$\begin{aligned} r_\alpha &= \left\{\begin{matrix}(1,0)\\(0,1)\end{matrix}\right\} d_{10}. \\ r_\alpha &= \left\{\begin{matrix}(1,1)\\(1,-1)\end{matrix}\right\} d_{11}. \end{aligned} \tag{3.42}$$

The vector length $|k|$ is considered for positive half space to be $k = \pi$ and the directions are $(|k|,0)$ and $(|k|/\sqrt{2},|k|/\sqrt{2})$. In Fourier space, ∇^2 is $-k \cdot k$, with the corresponding discrete *S*(*k*):

$$-q^2 \rightarrow S(k) = \sum_{\alpha} d_\alpha D_\alpha D_\alpha \tag{3.43}$$

The values of the weights d_α are found by invoking the following two conditions:

$$\frac{\partial^2 S}{\partial k_{x_i}^2} = -2 \text{ where } x_i = x, y \tag{3.44}$$

and:

$$S(\pi,0) = S\left(\frac{\pi}{\sqrt{2}},\frac{\pi}{\sqrt{2}}\right) \tag{3.45}$$

The first equation is for a scaling condition, whereas the second condition is for an isotropy condition, which result in weights $d_{10} = 0.53015$ and $d_{11} = 0.469849$. The stencil is obtained as follows:

$$\begin{aligned} S(k) = 0.530151&\left(-4\sin\left[\frac{k_x \Delta x}{2}\right]^2 - 4\sin\left[\frac{k_y \Delta x}{2}\right]^2\right) \\ +0.469849&\left(-2\sin\left[\frac{(k_x + k_y)\Delta x}{2}\right]^2 - 2\sin\left[\frac{(k_x - k_y)\Delta x}{2}\right]^2\right). \end{aligned} \tag{3.46}$$

Other different vector lengths are also chosen and different stencil weights are obtained. Details are given as follows by categorizing three different cases for the different vector choices:

Case 1: In this case, the first vector choice is $S(\pi,0)=S\left(\pi/\sqrt{2},\pi/\sqrt{2}\right),$ which is given above, and then the stencil in equation (3.46) is obtained with weights $d_{10}=0.53015$ and $d_{11}=0.469849$.

Case 2: In this case, the second vector choice is $S(\pi/2,0)=S\left(\pi/2\sqrt{2},\pi/2\sqrt{2}\right)$ and for this vector choice, the obtained weights are $d_{10}=0.63778$ and $d_{11}=0.362218$. The choice of any vector gives different weights but the terms in stencil remain the same as in equation (3.38). For the second vector choice, the stencil is given below:

$$\begin{aligned} S(k) &= 0.637782\left(-4\sin\left[\frac{k_x\Delta x}{2}\right]^2-4\sin\left[\frac{k_y\Delta x}{2}\right]^2\right) \\ &+0.362218\left(-2\sin\left[\frac{(k_x+k_y)\Delta x}{2}\right]^2-2\sin\left[\frac{(k_x-k_y)\Delta x}{2}\right]^2\right) \end{aligned} \tag{3.47}$$

Case 3: In this case, the third vector choice is $S(3\pi/4,0)=S\left(3\pi/4\sqrt{2},3\pi/4\sqrt{2}\right)$ and for this vector choice the obtained weights are $d_{10}=0.59713$ and $d_{11}=0.402869$. For the third vector choice the stencil is given below:

$$\begin{aligned} S(k) &= 0.597131\left(-4\sin\left[\frac{k_x\Delta x}{2}\right]^2-4\sin\left[\frac{k_y\Delta x}{2}\right]^2\right) \\ &+0.402869\left(-2\sin\left[\frac{(k_x+k_y)\Delta x}{2}\right]^2-2\sin\left[\frac{(k_x-k_y)\Delta x}{2}\right]^2\right) \end{aligned} \tag{3.48}$$

The stencils obtained in equations (3.46–3.48) are in **k** domain. For the simulations in cell dynamics (CDS), these equations need to be transformed into real space analogue. The transformation is given below in equations (3.49–3.51) for equations (3.46–3.48) respectively.

$$S(k)_{BV(D2Q9)_{case1}} = \frac{1}{(\Delta x)^2}\left\{\begin{matrix} 0.530151(2\cos(k_x\Delta x)+2\cos(k_y\Delta x)) \\ +0.234925(2\cos[(k_x+k_y)\Delta x] \\ +2\cos[(k_x-k_y)\Delta x]-3.0603 \end{matrix}\right\}. \tag{3.49}$$

$$S(k)_{BV(D2Q9)_{cas2}} = \frac{1}{(\Delta x)^2}\left\{\begin{matrix} 0.637782(2\cos(k_x\Delta x)+2\cos(k_y\Delta x)) \\ +0.181109(2\cos[(k_x+k_y)\Delta x] \\ +2\cos[(k_x-k_y)\Delta x]-3.27556 \end{matrix}\right\}. \tag{3.50}$$

$$S(k)_{BV(D2Q9)_{case3}} = \frac{1}{(\Delta x)^2}\left\{\begin{matrix} 0.597131(2\cos(k_x\Delta x)+2\cos(k_y\Delta x)) \\ +0.201435(2\cos[(k_x+k_y)\Delta x] \\ +2\cos[(k_x-k_y)\Delta x]-\ 3.19426 \end{matrix}\right\}. \tag{3.51}$$

The subscript BV in equations (3.49–3.51) indicates that these stencils are obtained by following B.A.C. van Vlimmeren’s method. It is clear from Figure 3.3 that all the BV(D2Q9) stencils are isotropic in comparison to PK(D2Q9). The two stencils BV(D2Q9)$_{\text{case2}}$ and BV(D2Q9)$_{\text{case3}}$ are equally isotropic for $r < 2.3$ and seem to be better than PK(D2Q9), but these are anisotropic for larger *r*. The deviation of lines from radius *r* show that the green line for the BV(D2Q9)$_{\text{case1}}$ behaves in almost the same way as OP(D2Q9), which is less isotropic than the other two cases for $1.5 < r < 2.5,$ and more isotropic for $r > 2.5.$ It must be noted that the isotropy of the BV(D2Q9)$_{\text{case1}}$ stencil is better than all the other discrete Laplacians in the 9–point family.

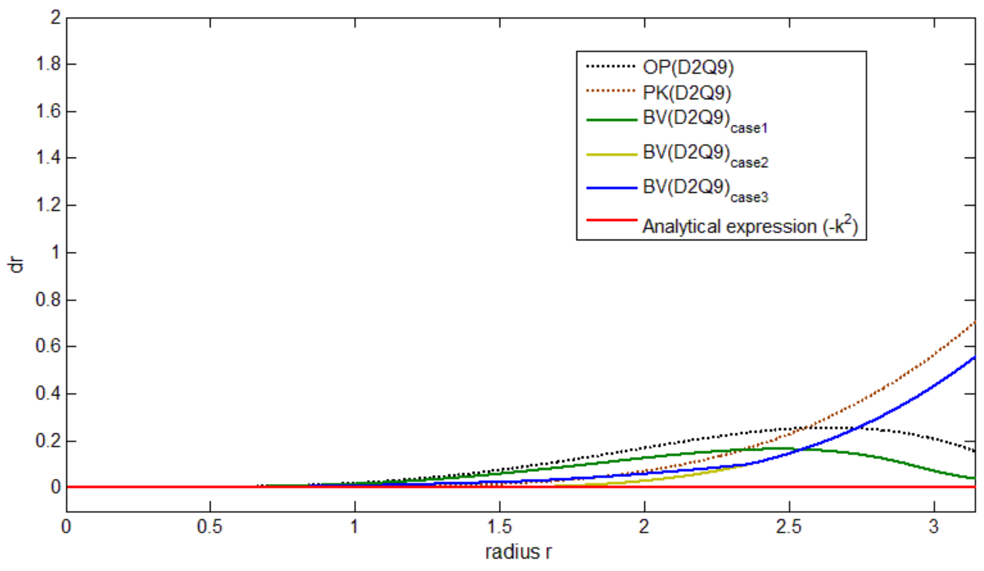


Figure 3.3: The value of a measure of the isotropy $d(r)$ for the actual Laplacian (red solid line for $-k^2$) and other discrete Laplacians against the radius *r*. The BV(*D2Q9*) are newly derived stencils.

After the investigation of 9–point family Laplacian schemes for error order $O(h^2)$, the stencil shapes are analysed for error order $O(h^4)$. These shapes include a 9–point star stencil and a 17–point stencil. The Laplacian operator is then derived from the finite differences and all these are discussed here. The 9–point star and 17–point Laplacians have the error order term $O(h^4)$. Stencil shapes for these three Laplacians are shown in Figure 3.4 (a) and (b) respectively. The dark circles in each part of Figure 3.4 are the points for forming Laplacians schemes. The Laplacian schemes obtained from the 9–point star and17–point stencils and the corresponding Fourier transforms are given as follows:

$$\Delta(\psi)_{D2Q9_{star}} = \frac{1}{12h^2}\left[16\sum_{i=4}^{4}\psi^{(1)} - \sum_{i=1}^{4}\psi^{(2)} - 60\psi^{(0)}\right]. \qquad (3.52)$$

$$\Gamma(k)_{D2Q9_{Star}} = \frac{1}{6(\Delta x)^2}\begin{bmatrix} 16(\cos(k_x\Delta x)+\cos(k_y\Delta x)) \\ -(\cos(2k_x\Delta x)+\cos(2k_y\Delta x))-30 \end{bmatrix}. \qquad (3.53)$$

$$\Delta(\psi)_{D2Q17} = \frac{1}{120h^2}\left[128\sum_{i=4}^{4}\psi^{(1)} + 16\sum_{i=1}^{4}\psi^{(2)} - 8\sum_{i=4}^{4}\psi^{(3)} - \sum_{i=4}^{4}\psi^{(4)} - 540\psi^{(0)}\right]. \qquad (3.54)$$

$$\Gamma(k)_{D2Q17} = \frac{1}{60(\Delta x)^2}\begin{bmatrix} 128\left[\cos(k_x\Delta x) + \cos(k_y\Delta x)\right] \\ +16\left[\cos\left[(k_x + k_y)\Delta x\right] + \cos\left[(k_x - k_y)\Delta x\right]\right] \\ -8\left[\cos(2k_x\Delta x) + \cos(2k_y\Delta x)\right] \\ -\left[\cos\left[(2k_x + 2k_y)\Delta x\right] + \cos\left[(2k_x - 2k_y)\Delta x\right]\right] - 270 \end{bmatrix}. \qquad (3.55)$$

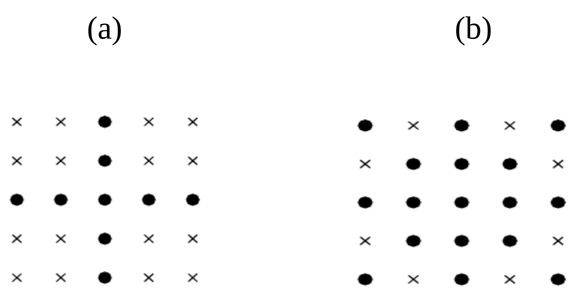


Figure 3.4: Stencil shapes of Laplacian on 2D grid. a) 9–point star stencil; b) 17–point stencil.

Equations (3.53) and (3.55) are the corresponding Fourier transforms of equations (3.52), and (3.54) respectively. The 9–point star discrete Laplacian is reported as anisotropic and the 17–point are reported as isotropic [7]. In Figure 3.5, the isotropy is measured for these fourth-order discrete Laplacians along with the isotropy measure of second order PK(D2Q9), discrete Laplacian and the analytical expression (actual Laplacian $-k^2$). It is obvious from Figure 3.5 that the 9–point star stencil deviates more from radius r and therefore becomes completely anisotropic. The isotropy of the 17–point stencil (D2Q17) with green dashed line is not much better than the discrete Laplacian PK(D2Q9) with the blue solid line. The 17–point stencil cab be observed to have maximum divergence, therefore, it cannot be considered to be isotropic.

In the way the Laplacian schemes are used in cell dynamics simulation technique, all the two-dimensional Laplacian schemes are modified a little by calculating the average weights for the nearest neighbours (NN), next nearest neighbours (NNN), next–next nearest neighbours (NNNN) and next–next–next nearest neighbours (NNNNN), in accordance with the simulation requirements; these are given in the form of $\langle\langle X \rangle\rangle$.

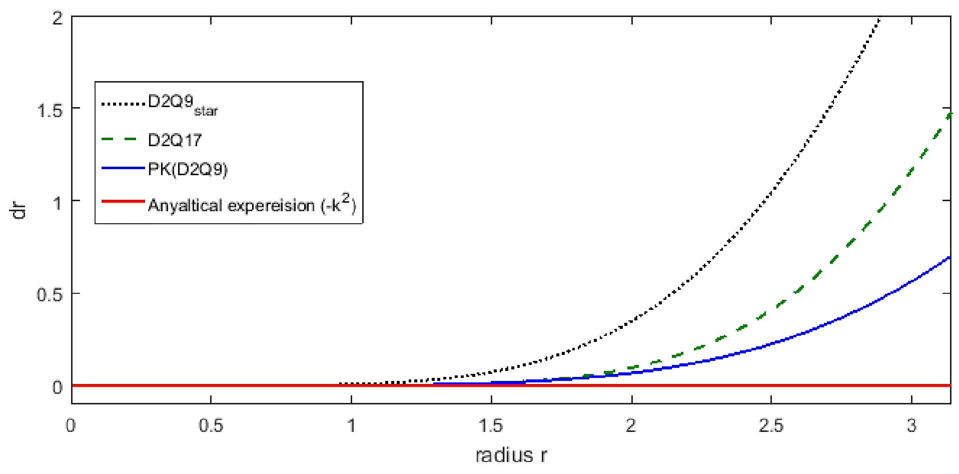


Figure 3.5: The value of a measure of the isotropy $d(r)$ for the actual Laplacian (red solid line for $-k^2$) and other discrete Laplacians with r the radius.

The two-dimensional Laplacian schemes are listed with alphabetical titles so that they may be easily referred to throughout the thesis**.** The equations with small modifications listed below are for equations (3.16, 3.17, 3.19–3.22, 3.46–3.48, 3.52 and 3.54) respectively:

$$\langle\langle \psi \rangle\rangle_{A(D2Q5)} = \frac{1}{4}\sum_{NN}\psi \tag{3.56}$$

$$\langle\langle \psi \rangle\rangle_{D2Q5} = \frac{1}{4}\sum_{NNN}\psi \tag{3.57}$$

$$\langle\langle \psi \rangle\rangle_{PK(D2Q9)} = \frac{1}{5}\sum_{NN}\psi + \frac{1}{20}\sum_{NNN}\psi \tag{3.58}$$

$$\langle\langle\psi\rangle\rangle_{OP(D2Q9)} = \frac{1}{6}\sum_{NN}\psi + \frac{1}{12}\sum_{NNN}\psi \tag{3.59}$$

$$\langle\langle\psi\rangle\rangle_{D2Q9} = \frac{1}{7}\sum_{NN}\psi + \frac{3}{28}\sum_{NNN}\psi \tag{3.60}$$

$$\langle\langle\psi\rangle\rangle_{BK(D2Q5)} = \frac{1}{8}\sum_{NN}\psi + \frac{1}{8}\sum_{NNN}\psi \tag{3.61}$$

$$\langle\langle\psi\rangle\rangle_{BV(D2Q9)_{case1}} = 0.173235\sum_{NN}\psi + 0.076765\sum_{NNN}\psi \tag{3.62}$$

$$\langle\langle\psi\rangle\rangle_{BV(D2Q9)_{case2}} = 0.194709\sum_{NN}\psi + 0.055291\sum_{NNN}\psi \tag{3.63}$$

$$\langle\langle\psi\rangle\rangle_{BV(D2Q9)_{case3}} = \mathit{0.186939}\sum\psi + \mathit{0.063061}\sum_{NNN}\psi \tag{3.64}$$

$$\langle\langle\psi\rangle\rangle_{D2Q9_{star}} = \frac{4}{15}\sum_{NN}\psi - \frac{1}{60}\sum_{NNN}\psi \tag{3.65}$$

$$\langle\langle\psi\rangle\rangle_{D2Q17} = \frac{32}{135}\sum_{NN}\psi + \frac{4}{135}\sum_{NNN}\psi - \frac{2}{135}\sum_{NNNN}\psi - \frac{1}{540}\sum_{NNNNN}\psi \tag{3.66}$$

3.2.2 Three–dimensional Laplacian schemes

Three–dimensional Laplacian schemes are presented in this section. The isotropic and anisotropic three–dimensional Laplacian schemes are investigated to be employed in CDS to analyse the numerical results of one order parameter evolution in a Lamellae morphology system of block copolymers.

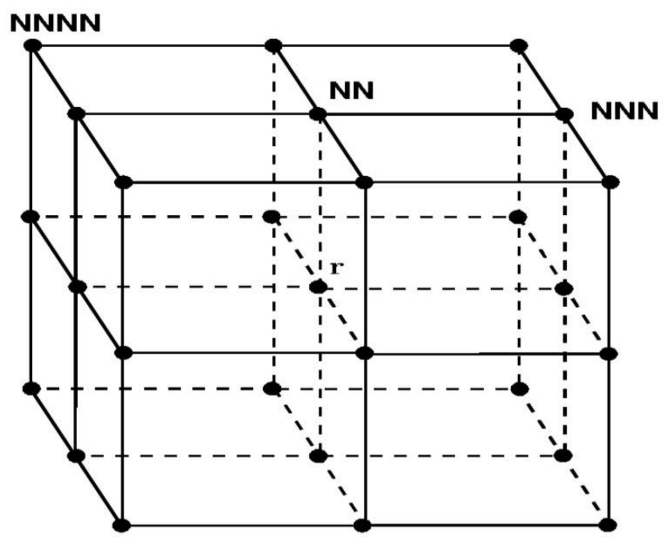


Figure 3.6: The stencil for three-dimensional Laplacian schemes, where NN, NNN, and NNNN are the nearest neighbours, next–nearest neighbours and next–next nearest neighbours to the point r. This image is taken with the permission of [2].

The Laplacian schemes are derived through finite difference considering the nearest neighbours (NN) and the next nearest neighbours (NNN) with error order $O(h^2)$. The stencil points can be seen in Figure 3.6. All three-dimensional Laplacian schemes are listed below:

$$\Delta(\psi)_{D3Q7} = \frac{1}{h^2}\left[\sum_{i=1}^{6}\psi^{(1)} - 6\psi^{(0)}\right]. \tag{3.67}$$

$$\Delta(\psi)_{D3Q15} = \frac{1}{12h^2}\left[8\sum_{i=1}^{6}\psi^{(1)} + \sum_{i=1}^{8}\psi^{(3)} - 56\psi^{(0)}\right]. \tag{3.68}$$

$$\Delta(\psi)_{D3Q19} = \frac{1}{6h^2}\left[2\sum_{i=1}^{6}\psi^{(1)} + \sum_{i=1}^{12}\psi^{(2)} - 24\psi^{(0)}\right]. \tag{3.69}$$

$$\Delta(\psi)_{D3Q27} = \frac{1}{36h^2}\left[16\sum_{i=1}^{6}\psi^{(1)} + 4\sum_{i=1}^{12}\psi^{(2)} + \sum_{i=1}^{8}\psi^{(3)} - 152\psi^{(0)}\right]. \tag{3.70}$$

$$\Delta(\psi)_{PK(D3Q27)} = \frac{1}{30h^2}\left[14\sum_{i=1}^{6}\psi^{(1)} + 3\sum_{i=1}^{12}\psi^{(2)} + \sum_{i=1}^{8}\psi^{(3)} - 128\psi^{(0)}\right]. \tag{3.71}$$

$$\Delta(\psi)_{SO(D3Q27)} = \frac{1}{22h^2}\left[6\sum_{i=1}^{6}\psi^{(1)} + 3\sum_{i=1}^{12}\psi^{(2)} + \sum_{i=1}^{8}\psi^{(3)} - 80\psi^{(0)}\right]. \tag{3.72}$$

The subscripts PK and SO stand for Patra Kartunnen and Shinozaki–Oono respectively. The 7–point Laplacian (D3Q7) scheme in equation (3.67) is based on its nearest neighbours (NN) only and is obtained from the simple central finite difference method. The D3Q7 is reported as anisotropic [6, 7]. Laplacian schemes for 15–point (D3Q15) and 19–point (D3Q19) stencils given in equations (3.68) and (3.69) are obtained from general finite difference derivations and are described as isotropic [6, 7]. There are various schemes for the 27–point Laplacian operator, which are discussed in the literature. Sumesh *et. al.* [6] discussed various three–dimensional 27–point Laplacian schemes. The 27–point Laplacian scheme (D3Q27) in equation (3.59) is based on its nearest neighbours (NN), next nearest neighbours (NNN) and the next–next nearest neighbours (NNNN) and is described as isotropic [6]. The Laplacian scheme PK(D3Q27) given in equation (3.71) has been systematically derived by imposing conditions of rotational invariance and isotropy [7]. The Laplacian scheme SO(D3Q27) given in equation (3.72) has been used in CDS as an averaging operator to maintain isotropy, but is not the optimal choice for achieving isotropy [49].

The isotropy of the three–dimensional discrete Laplacians discussed above can also be analysed in Fourier space. The corresponding Fourier transforms of Laplacian schemes in equations (3.66–3.72) are given below respectively:

$$\Gamma(k)_{D3Q7} = \frac{2}{(\Delta x)^2}\left[(\cos(k_x\Delta x) + \cos(k_y\Delta x) + \cos(k_z\Delta x)) - 3\right]. \tag{3.73}$$

$$\Gamma(k)_{D3Q15} = \frac{1}{6(\Delta x)^2}\begin{bmatrix} 8(\cos(k_x\Delta x)+\cos(k_y\Delta x)+\cos(k_z\Delta x)) \\ +4(\cos(k_x\Delta x)\cos(k_y\Delta x)\cos(k_z\Delta x))-28 \end{bmatrix}. \tag{3.74}$$

$$\Gamma(k)_{D3Q19} = \frac{1}{3(\Delta x)^2}\begin{bmatrix} 2(\cos(k_x\Delta x)+\cos(k_y\Delta x)+\cos(k_z\Delta x)) \\ +4(\cos(k_x\Delta x)\cos(k_y\Delta x)+\cos(k_x\Delta x)\cos(k_z\Delta x) \\ +\cos(k_y\Delta x)\cos(k_z\Delta x))-12 \end{bmatrix}. \tag{3.75}$$

$$\Gamma(k)_{D3Q27} = \frac{1}{9(\Delta x)^2}\begin{bmatrix} 8(\cos(k_x\Delta x)+\cos(k_y\Delta x)+\cos(k_z\Delta x)) \\ +4(\cos(k_x\Delta x)\cos(k_y\Delta x)+\cos(k_x\Delta x)\cos(k_z\Delta x) \\ +\cos(k_y\Delta x)\cos(k_z\Delta x)) \\ +2(\cos(k_x\Delta x)\cos(k_y\Delta x)\cos(k_z\Delta x))-38 \end{bmatrix}. \tag{3.76}$$

$$\Gamma(k)_{PK(D3Q27)} = \frac{1}{15(\Delta x)^2}\begin{bmatrix} 14(\cos(k_x\Delta x)+\cos(k_y\Delta x)+\cos(k_z\Delta x)) \\ +6(\cos(k_x\Delta x)\cos(k_y\Delta x)+\cos(k_x\Delta x)\cos(k_z\Delta x) \\ +\cos(k_y\Delta x)\cos(k_z\Delta x)) \\ +4(\cos(k_x\Delta x)\cos(k_y\Delta x)\cos(k_z\Delta x))-64 \end{bmatrix}. \tag{3.77}$$

$$\Gamma(k)_{SO(D3Q27)} = \frac{1}{22(\Delta x)^2}\begin{bmatrix} 12(\cos(k_x\Delta x)+\cos(k_y\Delta x)+\cos(k_z\Delta x)) \\ +12(\cos(k_x\Delta x)\cos(k_y\Delta x)+\cos(k_x\Delta x)\cos(k_z\Delta x) \\ +\cos(k_y\Delta x)\cos(k_z\Delta x)) \\ +8(\cos(k_x\Delta x)\cos(k_y\Delta x)\cos(k_z\Delta x))-80 \end{bmatrix}. \tag{3.78}$$

The resulting equation after expanding cos(x) around the origin in Fourier transforms (3.74 – 3.77) of discrete Laplacians D3Q15, D3Q19, D3Q27 and PK(D3Q27) is obtained the same for all, which is given below:

$$\Gamma(k) = -k^2 + \frac{k^4}{12} + O(k_\alpha^6), \tag{3.79}$$

and the expansions of cos(x) around origin for D3Q7 and SO(D3Q27) are as follows:

$$\Gamma(k)_{D3Q7} = -k^2 + \left[\frac{k^4}{12} - \frac{k_x^2k_y^2 + k_x^2k_z^2 + k_y^2k_z^2}{6}\right] + O(k_\alpha^6). \tag{3.80}$$

$$\Gamma(k)_{SO(D3Q27)} = -k^2 + \left[\frac{k^4}{12} + \frac{2[k_x^2 k_y^2 + k_x^2 k_z^2 + k_y^2 k_z^2]}{33} \right] + O(k_\alpha^6). \qquad (3.81)$$

All the stencils from (3.74) to (3.77) are observed to be isotropic up to fourth order in *k*. The 7–point stencil D3Q7 can be expected to be anisotropic, due to using the few stencil points in its construction. The Laplacian scheme SO(D3Q27) use all 27 points but in spite of that it is isotropic at leading order in error.

Sumesh *et al*. [6] gave a good comparison of various three–dimensional discrete Laplacians for isotropy with the discrete operator used in the original study of Shinozaki and Oono [47]. They compare Laplacians via isocontour plots along the $k_z = 0$ and $k_z = \pi$ planes, but such analogy only provides qualitative information. For quantitative analysis, the isotropy of the three–dimensional discrete Laplacians may be measured by calculating *d* and then observing the deviation of the Laplacian curve along the radius *r*. Such measurement of isotropy can be performed after introducing spherical coordinates (r, θ, ϕ) in k space by plotting:

$$d(r) = \max_{\theta,\phi} \Gamma(r,\theta,\phi) - \min_{\theta,\phi} \Gamma(r,\theta,\phi). \qquad (3.82)$$

The results for discrete Laplacians D3Q15, D3Q19, D3Q27 and PK(D3Q27) are shown in Figure 3.7. All four of these discrete Laplacians belong to the same class, as can be observed from equation (3.79). However, as the *r* increases, the Laplacian becomes anisotropic and therefore the proper isotropy can be found for small *r*. It can be concluded that the D3Q19 (19–point stencil in equation (3.69)) is most isotropic.

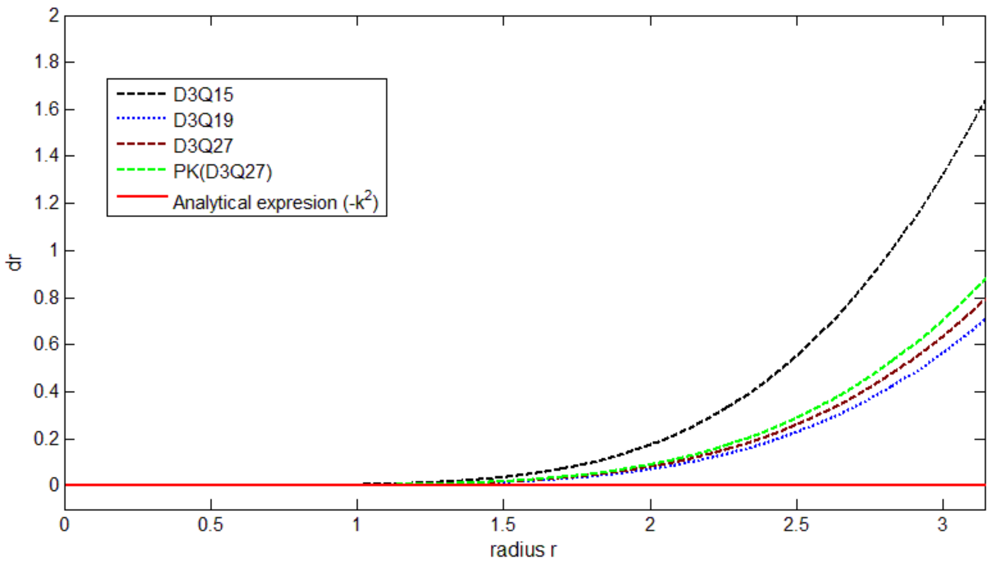


Figure 3.7: The value of a measure of the isotropy $d(r)$ for the actual Laplacian (red solid line for $-k^2$) and other discrete Laplacians with r the radius.

Another three–dimensional 27–point isotropic stencil operator is given below, obtained by the method of B.A.C. van Vlimmeren [102], and the same method has also been explained by Fraaije and co–workers [61, 103]:

$$\frac{\partial^2 S}{\partial k_{x_i}^2} = -2 \text{ where } x_i = x, y, z \qquad \text{(Scaling condition)}$$

$$S(\pi,0,0) = S\left(\frac{\pi}{\sqrt{2}},\frac{\pi}{\sqrt{2}},0\right) = S\left(\frac{\pi}{\sqrt{3}},\frac{\pi}{\sqrt{3}},\frac{\pi}{\sqrt{3}}\right) \qquad \text{(Isotropy condition)}$$

The Laplacian scheme in equation (3.83) is obtained from the above method and is presented in real analogue. The weights are generated in the Fourier domain (3.84) so the equation has been modified in real analogue:

$$\Delta(\psi)_{BV(D3Q27)} = \frac{1}{h^2}\begin{bmatrix} 0.294726\sum_{i=1}^{6}\psi^{(1)} + 0.1177012\sum_{i=1}^{12}\psi^{(2)} \\ +0.0586061\sum_{i=1}^{8}\psi^{(3)} - 3.64975\psi^{(0)} \end{bmatrix} \qquad (3.83)$$

$$\Gamma(k)_{BV(D3Q27)} = \frac{1}{(\Delta x)^2}\begin{bmatrix} 0.589451(\cos(k_x\Delta x)+\cos(k_y\Delta x)+\cos(k_z\Delta x)) \\ +0.4708(\cos(k_x\Delta x)\cos(k_y\Delta x)+\cos(k_x\Delta x)\cos(k_z\Delta x) \\ +\cos(k_y\Delta x)\cos(k_z\Delta x)) \\ +0.4688(\cos(k_x\Delta x)\cos(k_y\Delta x)\cos(k_z\Delta x))-3.64975 \end{bmatrix}. \quad (3.84)$$

$$\Gamma(k)_{BV(D3Q27)} = -k^2 + \left[\frac{k^4}{12} + 0.8191[k_x^2k_y^2 + k_x^2k_z^2 + k_y^2k_z^2]\right] + O(k_\alpha^6). \quad (3.85)$$

The isotropy of the three–dimensional discrete Laplacians BV(D3Q27), D3Q7 and SO(D3Q27) are shown in Figure 3.8. It is obvious from equations (3.80), (3.81) and (3.85) that these three Laplacians have isotropic error up to second order in **k**. The discrete Laplacian D3Q7 is anisotropic because it is showing maximum deviation along the radius *r*. Overall, the less common BV(D3Q27) and the usual Shinozaki and Oono's choice SO(D3Q27) Laplacians are slightly anisotropic, whereas the BV(D3Q27) performs better than SO(D3Q27) for large *r* due to the specific conditions for which it was derived.

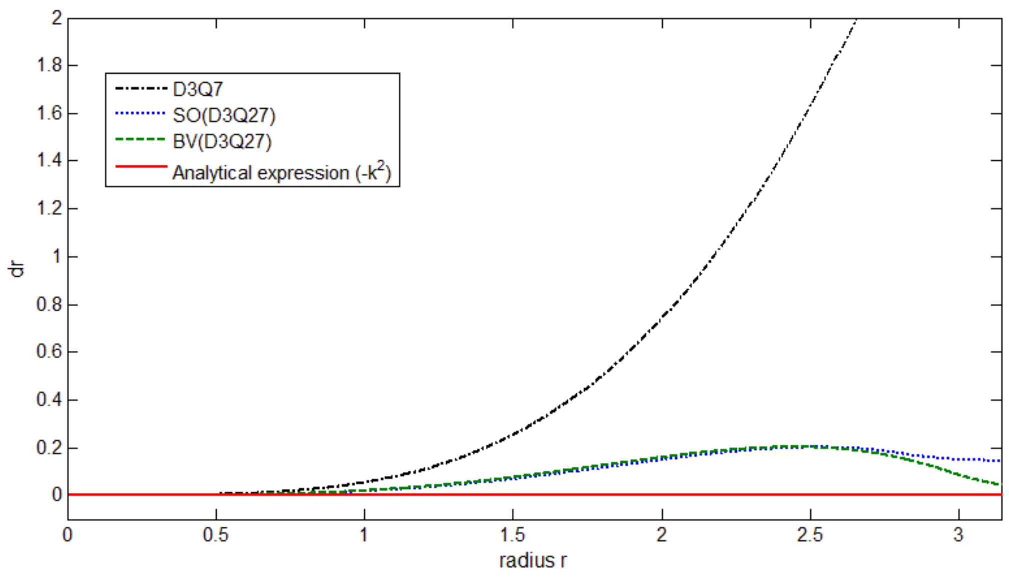


Figure 3.8: The value of a measure of the isotropy $d(r)$ for the actual Laplacian (red solid line for $-k^2$) and other discrete Laplacians with r the radius.

Generally, the discrete Laplacian D3Q19 seems the best choice among all the considered discrete Laplacians. It is fourth order in k and is using only 19 points on the grid. It is isotropic in the low $|\mathrm{k}|$ range and slightly anisotropic for larger $|\mathrm{k}|$.

In the way the Laplacian schemes are used for the simulations in CDS, like two–dimensional Laplacian schemes, all the three–dimensional Laplacian schemes are a little modified by calculating the average weights for the nearest neighbours (NN), next nearest neighbours (NNN), next–next nearest neighbours (NNNN) and next–next–next nearest neighbours (NNNNN), in accordance with the simulation requirements; these are given in the form of $\langle\langle X \rangle\rangle$. The three–dimensional Laplacian schemes are listed with alphabetical titles so that these may be easily referred to throughout the thesis. All Laplacian schemes are listed below for 7–point, 15–point, 19–point and 27–point stencils:

$$\langle\langle \psi \rangle\rangle_{D3Q27} = \frac{1}{6}\sum_{NN}\psi \tag{3.86}$$

$$\langle\langle \psi \rangle\rangle_{D3Q15} = \frac{1}{7}\sum_{NN}\psi + \frac{1}{56}\sum_{NNNN}\psi \tag{3.87}$$

$$\langle\langle \psi \rangle\rangle_{D3Q19} = \frac{1}{12}\sum_{NN}\psi + \frac{1}{24}\sum_{NNN}\psi \tag{3.88}$$

$$\langle\langle \psi \rangle\rangle_{D3Q27} = \frac{16}{152}\sum_{NN}\psi + \frac{4}{152}\sum_{NNN}\psi + \frac{1}{152}\sum_{NNNN}\psi \tag{3.89}$$

$$\langle\langle \psi \rangle\rangle_{PK(D3Q27)} = \frac{14}{128}\sum_{NN}\psi + \frac{3}{128}\sum_{NNN}\psi + \frac{1}{128}\sum_{NNNN}\psi \tag{3.90}$$

$$\langle\langle \psi \rangle\rangle_{SO(D3Q27)} = \frac{6}{80}\sum_{NN}\psi + \frac{3}{80}\sum_{NNN}\psi + \frac{1}{80}\sum_{NNNN}\psi \tag{3.91}$$

$$\langle\langle\psi\rangle\rangle_{BV(D3Q27)} = 0.0807524\sum_{\mathrm{NN}}\psi + 0.0322491\sum_{NNN}\psi + 0.0160576\sum_{NNNN}\psi \tag{3.92}$$

3.3 Stability analysis

For stability considerations, the linearized CDS equation (3.93) [101] can be derived with *tanh* map from equation (3.10) by ignoring the term $-B\psi(i,t)$.

$$\frac{\partial\psi}{\partial t} = \nabla^2\left[(1-A)\psi - D\nabla^2\psi\right] \tag{3.93}$$

The Fourier transform of order parameter linearized around the homogeneous fixed point gives equation (3.94), see Ref. ([104], equation (3.7)).

$$\delta\xi_k(t+\Delta t) = \sum_k H_{k,k'}\delta\xi_{k'} \tag{3.94}$$

where:

$$H_{k,k'} = [(1-A)\Delta t\Gamma(k) - D\Delta t\Gamma(k)^2]\delta_{k,k'} \tag{3.95}$$

It is essential to maintain $|H_{k,k}| < 1$ for the stability of the successive iterations of Fourier modes. Thus, two different instability conditions of bifurcation can be dealt with; one is tangential bifurcation, where $H_{k,k} > 1$ may occur, and the second is sub-harmonic bifurcation where $H_{k,k} < -1$ can occur. The condition of tangential bifurcation gives equation (3.96), which results in smaller growth of k modes [104].

$$-\Gamma(k) < 1 \tag{3.96}$$

The condition of sub–harmonic bifurcation creates constraints on time step Δt, which is dependent on mesh size [101, 104]. The condition of sub–harmonic bifurcation gives the following equation:

$$(1-A)\Delta t\Gamma(k) - D\Delta t\Gamma^2(k) < -2 \tag{3.97}$$

where Δt is the time step. The sub–harmonic bifurcation can be avoided for all *k* modes for the Laplacian operators by forming the following inequalities. Here a few Laplacian schemes are selected to give just an idea of how the formation of inequalities can be formed for the stability from Fourier transforms of Laplacian schemes. After substitution of the Laplacian operator $\Gamma(k)$ in equation (3.97) from Fourier transform equation (3.23) for 5–point stencil, the following equation (3.98) is obtained:

$$\Delta t < \frac{(\Delta x)^4}{32D - 4(A-1)(\Delta x)^2} \tag{3.98}$$

where *A*=1.5, *D* =0.7, and $\Delta x = 1.0$ are the parameters which are used for the numerical simulation in CDS code. After substitutions of the values for *A*, *D*, and Δx in the above inequalities, the time step values are obtained for all the Laplacian schemes and are given in Table 3.1.

Table 3.1: Time step (Δt) values for all two– and three–dimensional Laplacian schemes obtained from stability analysis criteria

S.No.	Time step	Corresponding Laplacians	S.No.	Time step	Corresponding Laplacians
1.	$\Delta t < 0.049$	5–point NN A(D2Q5)	11.	$\Delta t < 0.038$	17–pont D2Q17
2.	$\Delta t < 0.046$	5–point NNN D2Q5	12.	$\Delta t < 0.021$	7–point D3Q7
3.	$\Delta t < 0.072$	9–point PK(D2Q9)	13.	$\Delta t < 0.035$	15–point D3Q15
4.	$\Delta t < 0.09$	9–point SOP(D2Q9)	14.	$\Delta t < 1.111$	19–point D3Q19
5.	$\Delta t < 0.10$	9–point D2Q9	15.	$\Delta t < 0.097$	27-point D3Q27
6.	$\Delta t < 0.11$	9–point BK(D2Q9)	16.	$\Delta t < 0.085$	27-point PK(D3Q27)
7.	$\Delta t < 0.086$	9–point BV(D2Q9)$_{case1}$	17.	$\Delta t < 0.2173$	27-point SO(D3Q27)
8.	$\Delta t < 0.074$	9–point BV(D2Q9)$_{case2}$	18.	$\Delta t < 0.1698$	27-point BV(D3Q27)
9.	$\Delta t < 0.078$	9–point BV(D2Q9)$_{case3}$			
10.	$\Delta t < 0.030$	9–point Star (D2Q9)			

3.4 Conclusions

In this chapter, the CDS model has been explained for one order parameter systems for the evolution of lamellar forming of *A*–*B* diblock copolymer systems. The overall study in this chapter was conducted for the analysis of isotropic Laplacian operators to be used in CDS. The stencil (computational molecule) plays a very important role in the quality of the evolution of the order parameter, and keeping this point in consideration, several different stencil operators for CDS have been investigated.

Fourier analysis is also undertaken for the Laplacian operators. Isotropic properties of two– and three–dimensional Laplacian operators were analysed in detail. The two–dimensional 9–point Laplacians have been discussed with order error $O(h^2)$. The two–

dimensional 9–point star and 17–point Laplacian operators with order error $O(h^4)$ have been discussed and formatted for employing in CDS. In the same way, three–dimensional Laplacian schemes have been investigated and isotropic schemes have been notified. Following the method of B.A.C. van Vlimmeren for a three–dimensional isotropic stencil operator, the two–dimensional 9–point isotropic stencil operators (BV(D2Q9) in three cases) were derived and discussed along with three different vector choices. These are novel isotropic stencil operators which are presented in this study and are more efficient.

The 9–point family Laplacians, the stencils PK(D2Q9), BV(D2Q9)$_{case2}$ and BV(D2Q9)$_{case3}$ in 2D were found to be isotropic, and among these stencils the BV(D2Q9)$_{cas2}$ is optimally good in isotropy. In 3D, the 19–point stencil has been found to be more isotropic and it is more stable because it allows a larger time step value for Δt. The other stencils OP(D2Q9), BV(D2Q9)$_{case1}$, SO(D3Q27) and BV(D3Q27), have been found to be slightly anisotropic on the whole range k, but enabling larger time steps can be valid alternatives.

From the analysis of averaging operator (Laplacian) in CDS, it is clear that the original averaging operator does not represent the discrete Laplacian and that the CDS method should not be analysed via the TDGL equations. However, the original averaging operator can be replaced by a discrete representation of Laplacian via considering the stencil size and isotropic behaviour, which ensure the stability of CDS results. The investigation of several isotropic discrete Laplacian operators provides an alternative to use more isotropic Laplacian operators, which can helpful to resolve the grid related artefacts (anisotropies) in CDS results.

Chapter Four

4 Simulation Results (One–Order Parameter)

Numerous techniques have been used for modelling diblock copolymers over the last few decades; hence, a method is required that takes into account both accuracy and speed, taking into account a closer relationship between the real world and the laboratory by modelling the behaviour of diblock copolymers on a large scale and preventing the size effect problem. A coarse–grained discretization CDS scheme was identified as a promising candidate to compute and define the mesoscopic self–assembled structure of diblock copolymers [4]. The objective of the study in this chapter is to employ various 2D and 3D Laplacian operators in CDS for *A*–*B* dibblock copolymer systems in order to investigate isotropic simulation results. The Laplacian schemes employed for the simulations here were discussed in Chapter 3. To achieve this objective, initially a description has been given on the application of CDS to simulate microphase separation, front propagation and evolution of order parameter in *A*–*B* dibblock copolymer systems. The CDS technique is implemented in Fortran 90 programming language. The system specifications were used Linux 3.7 desktop Opensuse 12.3 with Intel(R) Xeon(R) CPU with IFORT compiler.

The discretized CDS version of Time–Dependent Ginzburg–Landau (TDGL) equation in the forward Euler method were simulated in Fortran program to carry out 2D and 3D simulations. The steps for the program were set up based on discrete equations (3.10), (3.11) and (3.12), which are given as follows:

1. Assigning random initial values to a variable representing order parameter $\psi.$;
2. Setting the periodic boundary conditions, i.e. x, y and z directions;
3. Calculation of first discrete Laplacian for order parameter ψ, i.e. $\langle\langle\psi\rangle\rangle-\psi$;

4. The multiplication of the result of step 3 by diffusion constant *D*, i.e. $D[\langle\langle\psi\rangle\rangle - \psi]$;
5. Calculation of the map function (3.12) based on the order parameter with initial random values and other constants with specific values;
6. Combining steps 4 and 5; see equation (3.11);
7. Calculation of second (outer) discrete Laplacian for the result of step 6, as follows:

$$\langle\langle\Gamma(t,i)\rangle\rangle - \Gamma(t,i)\,;$$

8. Calculation of order parameter based on new obtained values with respect to time evolution, see equation (3.10).

4.1 Two–dimensional simulations

A morphology is mainly concerned with the shape evolution of microphase separation of *A*–*B* diblock copolymers in different time steps. Several different morphologies were found: lamellae, cylinder, bicontinuous, and spheres, as well as the coexistence of spheres and cylinders [2]. The constants in equations (3.12) are treated as parameters for deciding a specific morphology. The parameter values for different morphologies are given in Table 4.1 [2].

Table 4.1: Simulation parameters used for the different morphologies

τ	f	u	v	B	D	A	**Morphology**
0.36	0.48	0.38	2.3	0.02	0.7	1.5	Lamellae
0.33	0.44	0.38	2.3	0.02	0.5	1.5	Bicontinuous
0.30	0.40	0.38	2.3	0.02	0.4	1.5	Cylinders
0.20	0.40	0.38	2.3	0.01	0.5	1.5	Spheres

The 2D simulations are presented here for lamellae morphology (lamellar forming) of the *A*–*B* diblock copolymer systems. For all the 2D simulations given in this section, the grid size was set at 128×128 with grid spacing $h = \Delta x = \Delta y = 1$ and these simulations were started from an initial random disordered state, i.e. $\psi = \pm 0.3$. It should be noted that snapshots of 2D simulations are presented without any specific time scale and these snapshots show numerical values of ψ. The source–code for the implementation of CDS presented in Figure 4.1 is given in Appendix A.

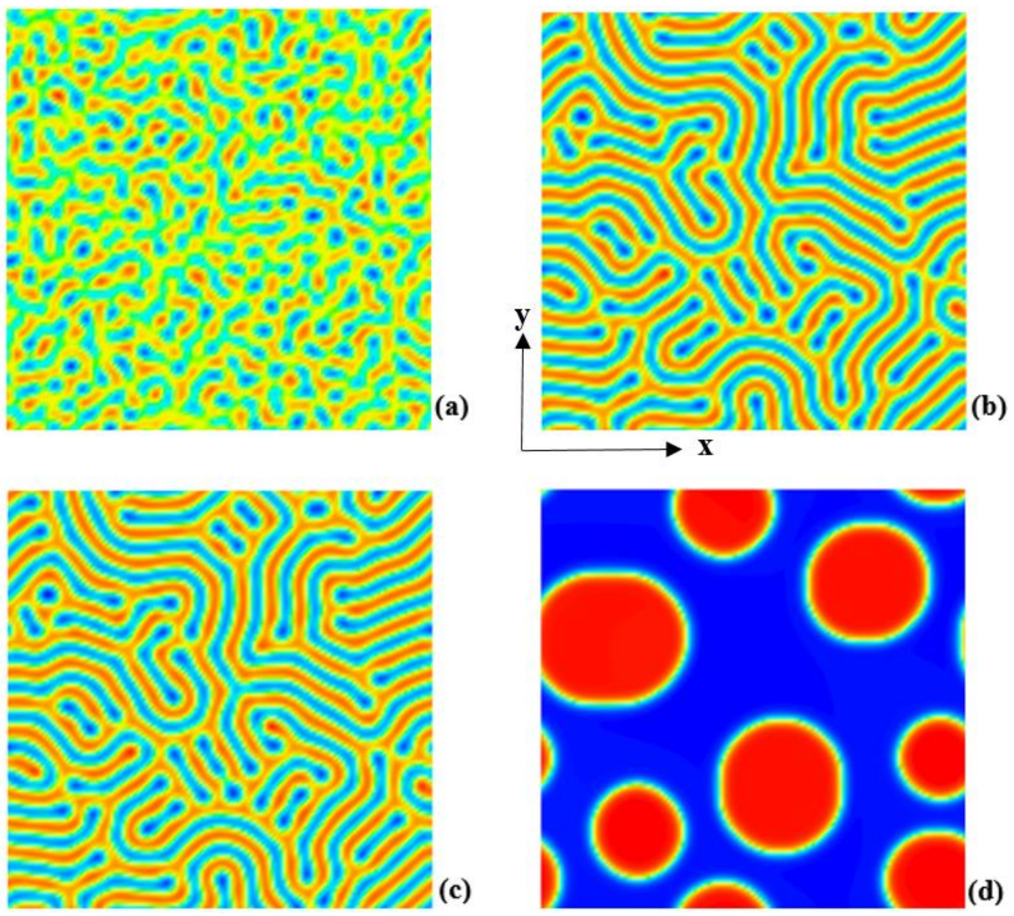


Figure 4.1: Results of CDS based on OP(D2Q9) 9–point stencil, equation (3.59). Real space simulation snapshots in (a), (b) and (c) are for 100th, 10000th and 100000th time steps respectively obtained by using parameters given in Table 4.2. Real space simulation snapshot in (d) is for 100000th time step obtained by using parameters given in Table 4.3.

Table 4.2: System parameters used in cell dynamical method for Lamellae morphology

CDS Parameters	τ	f	u	v	B	D	A	Initial random values
Lamellae Morphology	0.36	0.48	0.38	2.3	0.02	0.7	1.5	$\psi_i = \pm 0.3$

Table 4.3: System parameters used in cell dynamical method for binary blend

CDS Parameters	τ	f	u	v	B	D	A	Initial random values
Lamellae Morphology	0.36	0.48	0.38	2.3	0.0	0.7	1.5	$\psi_i = -0.3 \pm 0.01$

In the CDS method of *A*–*B* diblock copolymers, the compositional order parameter in terms of local and global volume fraction is defined by the following relation [4]:

$$\psi = \phi_A - \phi_B + (1 - 2f) \tag{4.1}$$

where ϕ_A, ϕ_B are the local volume fractions of the A and B monomers respectively. The volume fraction of A monomers is defined by the relation $f_A = N_A / (N_A + N_B)$, and similarly the volume fraction of B monomers is defined by the relation $f_B = N_B / (N_A + N_B)$, where N_A represents the number of monomers of block A, and N_B represents the number of monomers of block B. The constant *f* in both Tables 4.2 and 4.3 represents this ratio in the diblock copolymer system. For example, the constant *f* = 0.5 implies that the ratio of *A* and *B* monomers is equal in a mixture.

The simulation results presented in Figure 4.1 were obtained by using the 9–point isotropic operator of Oono and Puri's choice (equation (3.59)). In Figure 4.1 (a), (b) and (c), the images are shown for different stages of evolution of lamellae in a lamellar forming system of *A*–*B* diblock copolymers at different time step values. The initial stage of lamellae can be observed in Figure 4.1 (a) and the microphase separation starts to take place with respect to time and well-aligned lamellae can be seen in Figure 4.2 (b). In snapshots (b) and (c) in Figure 4.1, the lamellae coloured red along with an interfacial yellow colour can be seen microphase–separated in the diblock copolymer system representing either *A* or *B* block. In Figure 4.2 (b), the image is shown for a 10000th time step where the lamellar forming system becomes stable, which means the microphase separation has been completed and there is no further change in the lamellae evolution. Due to this, it is clear from Figure 4.1 (c) for the 100000th time step that lamellae are in a similar pattern as in Figure 4.1 (b) for a 10000th time step. Simulation results of microphase separation in a lamella forming system shown in Figure 4.1 (a), (b) and (c) were obtained by employing the parameter values given in Table 4.2.

The snapshot in Figure 4.1 (d) shows the simulation of a binary blend at a 100000th time step which was obtained by using a CDS parameter system (Table 4.3) for lamella forming system except (B=0). In this case, instead of microphase separation, a macrophase separation occurred in the pore system. In the pore system shown in Figure 4.1 (d), the domain is divided into two subdomains where the clearly visible yellow interfacial regions macrophase–separate A–rich subdomains in a red colour, and B-rich subdomains in a blue colour. It should be noted that the simulation result of a binary blend shown in Figure 4.1 (d) was obtained by using Laplacian scheme OP(D2Q9) and the red circular regions occur due to the isotropy of this scheme. The well aligned lamellae formations in Figure 4.1 (b) and (c) show that this Laplacian scheme is isotropic. Therefore, in this work, the Laplacian scheme OP(D2Q9) is termed as the default CDS averaging operator and the simulation results obtained using this scheme are termed as the default CDS results. All the 2D simulation results based on other Laplacian schemes will be compared with these default results.

Table 4.4 gives the complete information of Laplacian operators discussed in Chapter 3; these stencils are shown with the titles (alphabetic letters A–L) and along their weights *c1*, *c2*, *c3* and *c4* for their nearest neighbours (NN), and next–nearest neighbours (NNN), NNNN and NNNNN respectively.

Table 4.4: 2D stencils along with their weights for utilisation in computer code and isotropic or anisotropic status

Schemes	**Weights**				**Isotropic /Anisotropic**	**Equations**
	c1	*c2*	*c3*	*c4*		
A(D2Q5)	1/4	0	0	0	Anisotropic	(3.56)
D2Q5	0	1/4	0	0	Anisotropic	(3.57)
PK(D2Q9)	1/5	1/20	0	0	Isotropic	(3.58)
OP(D2Q9)	1/6	1/12	0	0	Isotropic	(3.59)
D2Q9	1/7	1/28	0	0	Anisotropic	(3.60)
BK(D2Q9)	1/8	1/8	0	0	Anisotropic	(3.61)
BV(D2Q9)$_{\text{case1}}$	0173235	0.076765	0	0	Isotropic	(3.62)
BV(D2Q9)$_{\text{case2}}$	0.194709	0.055291	0	0	Isotropic	(3.63)
BV(D2Q9)$_{\text{case3}}$	0.186939	0.063061	0	0	Isotropic	(3.64)
D2Q9$_{\text{star}}$	4/15	-1/60	0	0	Anisotropic	(3.65)
D2Q17	32/135	4/135	-2/135	-1/540	Isotropic	(3.66)

The format for the averaging operator used in the computer program is shown below:

$$DmQn = c1\sum_{NN} + c2\sum_{NNN} + c3\sum_{NNNN} + c4\sum_{NNNNN} \tag{4.2}$$

where *m* is the number of dimensions and *n* is the number of points in a stencil.

4.1.1 2D Simulation results based on anisotropic Laplacian schemes

In this section, the simulation results are presented for anisotropic schemes which are compared with the isotropic default CDS results shown in Figure 4.1.

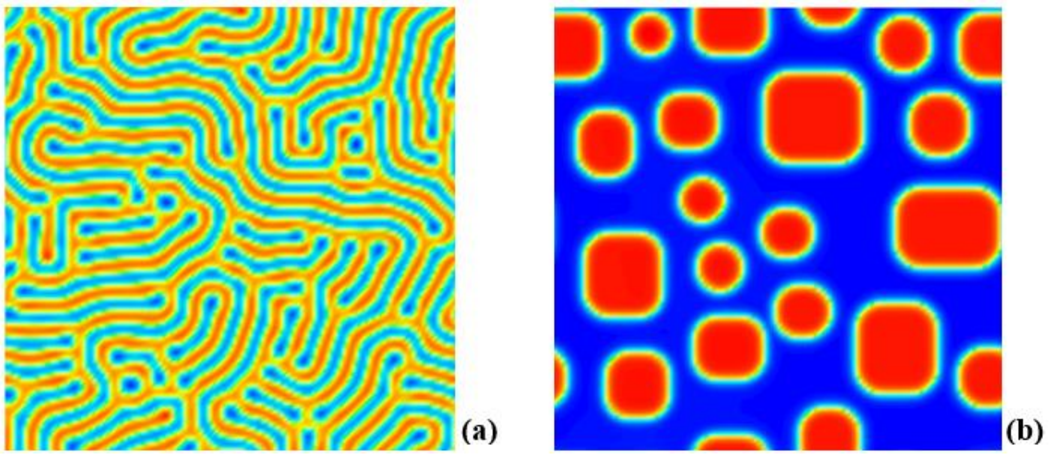


Figure 4.2: CDS results based on Laplacian scheme A(D2Q5); a) real space simulation snapshot at100000th time step by using parameters given in Table 4.2; b) real space simulation snapshot of binary blend at 100000th time step by using parameters given in Table 4.3.

The simulation snapshots shown in Figure 4.2 were obtained by using Laplacian scheme A(D2Q5). In Figure 4.2 (a), the parameters were employed from Table 4.2 and it can be observed that the lamellae seem to be short, not well aligned and do not form lamellar chain lengths compared to Figure 4.1 (c). The simulation snapshots in Figure 4.2 (b) were obtained by employing the parameters given in Table 4.3. In the pore system of binary blend shown in Figure 4.2 (c), the subdomains coloured red can be seen by the rectangular shapes. Compared to the red circular shapes formed for the subdomains in CDS default results shown in Figure 4.1 (d), the rectangular shapes of subdomains in Figure 4.2 (b) show that the scheme did not perform well for the simulation based on the parameters of binary blend in Table 4.3. The Laplacian scheme OP(D2Q9) produced isotropic results for both sets of parameter values in Table 4.2 and Table 4.3; however, scheme A(D2Q5), due to its anisotropy, could not accommodate two different sets of parameter values.

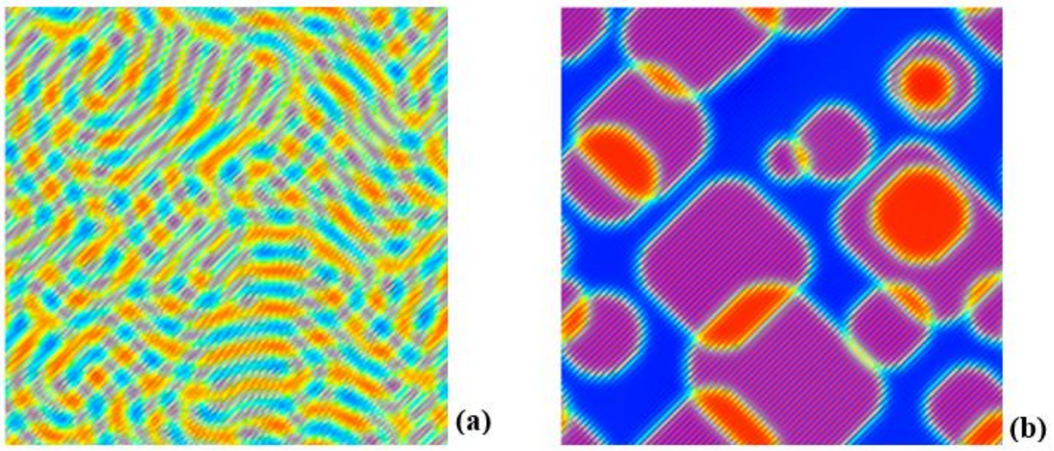


Figure 4.3: CDS results based on Laplacian scheme D2Q5 given in equation (3.57); a) real space simulation snapshot at100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3.

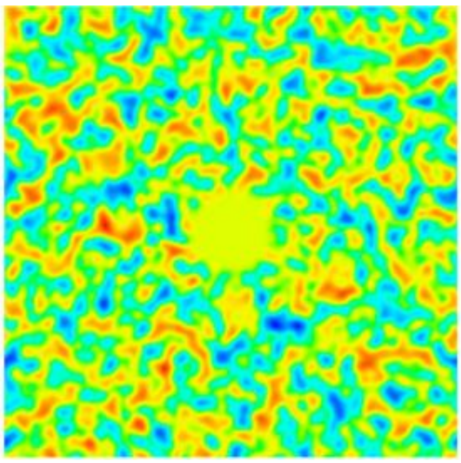


Figure 4.4: CDS results based on Laplacian scheme D2Q9 given in equation (3.60). The real space simulation snapshot at 100th time step obtained by using parameters given in Table 4.2.

The simulation results in Figure 4.3 were obtained using Laplacian D2Q5 and the simulation results in Figure 4.3 (a) and (b) were based on two different sets of parameter values given in Table 4.2 and Table 4.3 respectively. It can be seen in Figure 4.3 (a) that microphase separation in *A*–*B* diblock copolymer cannot be analysed or observed properly and the lamellae formations are not visible; also the macrophase separation in

Figure 4.3 (b) is not well defined compared to the default CDS simulation results given in Figure 4.1 (d). This is because of the anisotropy of the Laplacian scheme D2Q5.

The simulation snapshots given in Figure 4.4 were obtained using the Laplacian scheme D2Q9 given in equation (3.60). The parameters for this simulation were those given in Table 4.2. The simulation based on Laplacian scheme D2Q9 did not produce any meaningful results. Deformations can be observed and no state of microphase separation can be identified in Figure 4.4. The simulation based on this scheme did not run for longer time steps, and the numerical values were diverged immediately after the 100th time step. Both simulation results in Figure 4.3 and Figure 4.4 can be compared with the default results given in Figure 4.1. It must be noted that the source–code for simulations using schemes A(D2Q5), D2Q5 and D2Q9 were the same as those given in Appendix A for the default CDS using scheme OP(D2Q9); only the values of the weights (*c1, c2*) were changed. The simulation results given in Figure 4.5 were obtained using Laplacian scheme BK(D2Q9) and both snapshots show that this scheme did not produce well-defined lamellae formations.

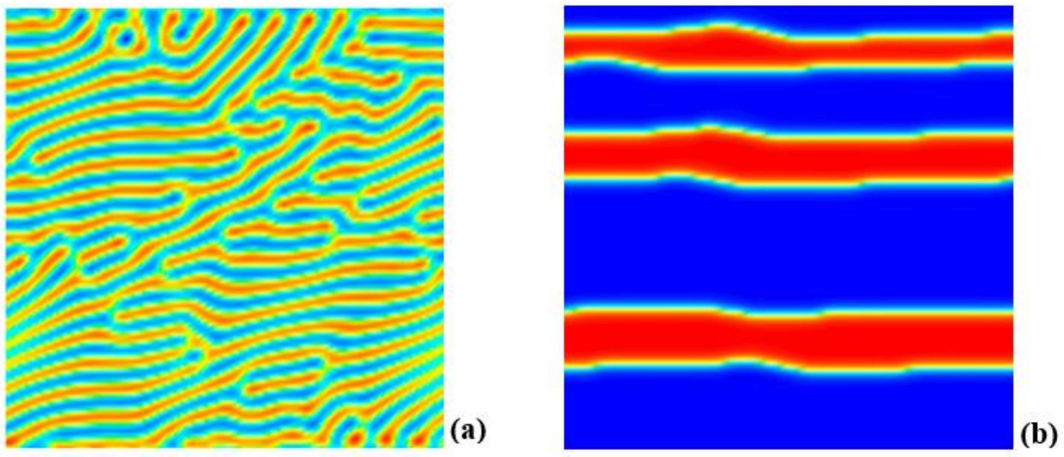


Figure 4.5: CDS results based on Laplacian scheme BK(D2Q9) given in equation (3.61); a) real space simulation snapshot at100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3.

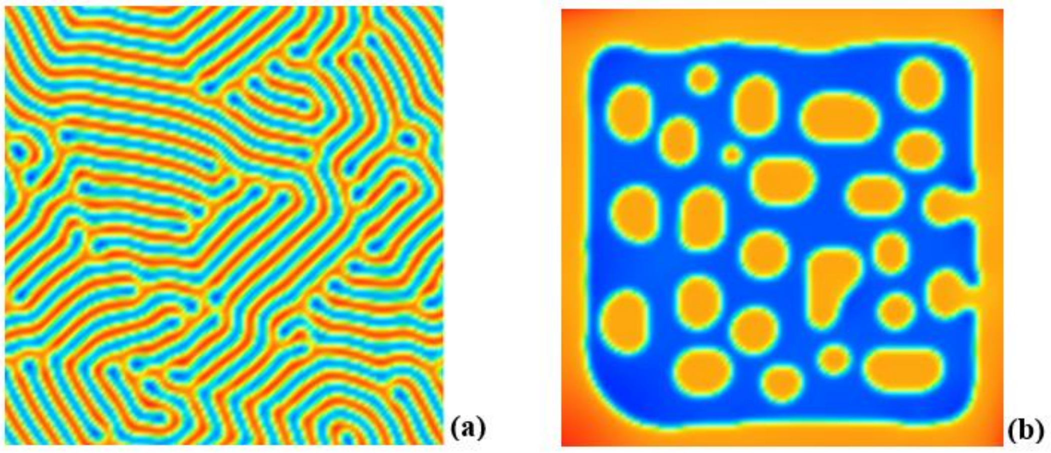


Figure 4.6: CDS results based on Laplacian scheme D2Q9$_{star}$; a) real space simulation snapshot at100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3.

Two different simulations were performed using Laplacian scheme BK(D2Q9) by employing the sets of parameters given in Table 4.2 and Table 4.3. In Figure 4.5 (a), the lamellae can be seen to be parallel with the horizontal axis and their formations are not in agreement with the default CDS results given in Figure 4.1 (c). Due to the anisotropy of scheme BK(D2Q9), the simulation result does not have any circular subdomains in Figure 4.5 (b) compared to those shown in Figure 4.1 (d) for default CDS results. The source code used for simulation based on scheme BK(D2Q9) is given in Appendix A. The Laplacian weights were only changed in the code of scheme OP(D2Q9).

In Figure 4.6, the simulation results are presented which were obtained by using 9–point 'star' Laplacian scheme D2Q9$_{star}$ equation (3.65). As with other simulations, the simulation based on scheme D2Q9$_{star}$ was also carried out for two different sets of parameter values given in Tables 4.2 and 4.3. It can be observed in Figure 4.6 (a), the lamellae are formed straight in one direction which is parallel to the horizontal axis. It can be observed from Figure 4.6 (b) that the simulation of binary blend based on Laplacian scheme D2Q9$_{star}$ is far different from the default CDS results; this is due to the anisotropy of the stencil 9–point 'star'. The source code used for scheme D2Q9$_{star}$ is given in Appendix B. The source code was modified, especially the code segment for writing Laplacian, because of the different scaling of scheme D2Q9$_{star}$ compared to other 9–point stencil formulas.

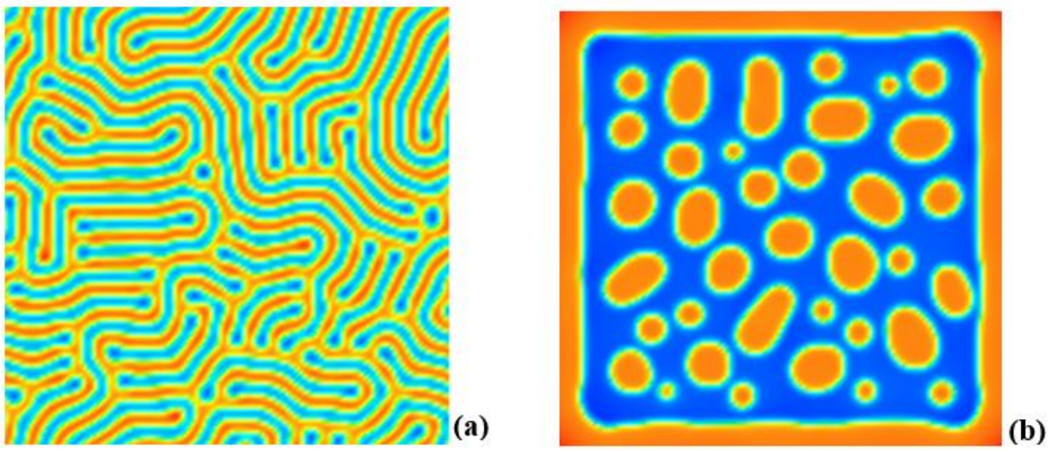


Figure 4.7: CDS results based on Laplacian scheme D2Q17 given in equation (3.66); a) real space simulation snapshot at100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3.

The simulation results shown in Figure 4.7 were obtained using Laplacian scheme D2Q17 (17-point stencil). In Figure 4.7 (a), it can be observed that the lamellae formations are well formed, but in Figure 4.7 (b), the simulation snapshot of the binary blend does not show a defined macrophase separation compared to the default CDS results in Figure 4.1 (d). The periodic boundary conditions do not seem to be preserved and therefore the orange colour can be seen on the boundaries. Due to the anisotropy of the Laplacian scheme D2Q17, it cannot be recommended for the simulations. The isotropy measure of this stencil can be seen in Figure 3.5 in Chapter three. The source code used for scheme D2Q17 is given in Appendix C. The source code given in Appendix A was modified for Laplacian D2Q17 because of the different scaling of scheme D2Q17 compared to other 9–point stencil formulas.

4.1.2 2D Simulation results based on isotropic Laplacian schemes

In this section the CDS results are presented based on isotropic Laplacian schemes. The simulations were run by employing two different sets of parameter values, as given in Tables 4.2 and 4.3, for each Laplacian scheme. The purpose of using two sets of parameter values is to investigate the isotropic behaviour of the Laplacian for the evolution of order parameter to investigate the phase separations in two different conditions. It must be noted that anisotropic Laplacians did not produce well–defined simulation results.

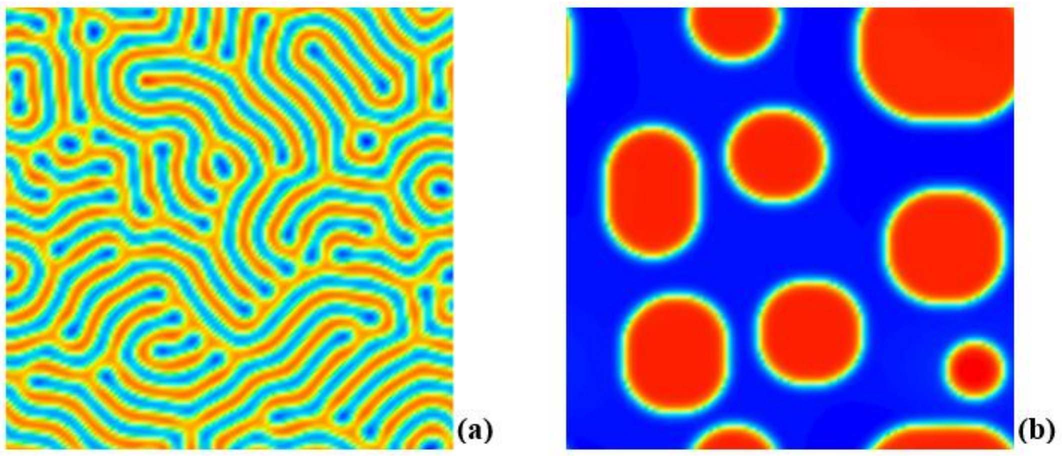


Figure 4.8: CDS results based on Laplacian scheme PK(D2Q9) given in equation (3.58); a) real space simulation snapshot at100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3.

The CDS results given in Figures 4.8, 4.9, 4.10 and 4.11 were obtained using Laplacian schemes OP(DQ9), BV(D2Q9)$_{case1}$, BV(D2Q9)$_{case2}$, and BV(D2Q9)$_{case3}$, respectively. The lamellae formations in these Figures seem to be well aligned and well defined and also no deformed or unidirectional chains of lamellae can be seen. It can be observed from the (a) snapshots of these Figures that the Laplacian schemes OP(D2Q9), BV(D2Q9)$_{case1}$, BV(D2Q9)$_{case2}$ and BV(D2Q9)$_{case3}$ yield isotropic results for microphase separation in *A*–*B* diblock copolymers. In the (b) snapshots of these Figures, the circular shapes of red subdomains in a binary blend can also be observed. The simulation results based on Laplacian schemes OP(D2Q9), BV(D2Q9)$_{case1}$, BV(D2Q9)$_{case2}$ and BV(D2Q9)$_{case3}$ can be compared with the default CDS results and it can be concluded that these Laplacians performed well enough for the simulations. The isotropy measure of the stencil of Laplacian scheme BV(D2Q9)$_{case1}$ was determined to be closely similar to that of Laplacian scheme OP(D2Q9); therefore the CDS results presented in Figure 4.9 are almost similar to those given in Figure 4.1 (c) and (d). It must be noted that in Table 4.4, the Laplacian schemes listed alphabetically in the right-hand column correspond to the stencils shown in the left-hand columns. The stencils of Laplacian schemes OP(D2Q9), BV(D2Q9)$_{case2}$, and BV(D2Q9)$_{case3}$ were determined to be isotropic for low **k** range in a similar way and among these, the stencil of Laplacian scheme BV(D2Q9)$_{case2}$ was determined with less divergence from radius *r* compared to OP(D2Q9) and BV(D2Q9)$_{case3}$. The source code for simulations using scheme OP(D2Q9), BV(D2Q9)$_{case1}$, BV(D2Q9)$_{case2}$ and BV(D2Q9)$_{case3}$ were used the same as that given in Appendix A for the default CDS using scheme OP(D2Q9); only the values of weights (*c1*, *c2*) were changed.

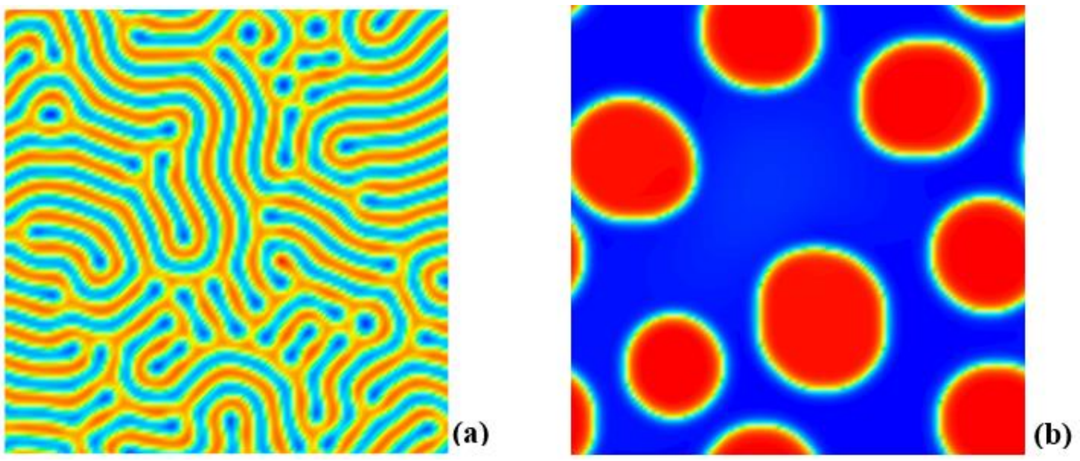


Figure 4.9: CDS results based on Laplacian scheme BV(D2Q9)$_{case1}$ given in equation (3.62); a) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3.

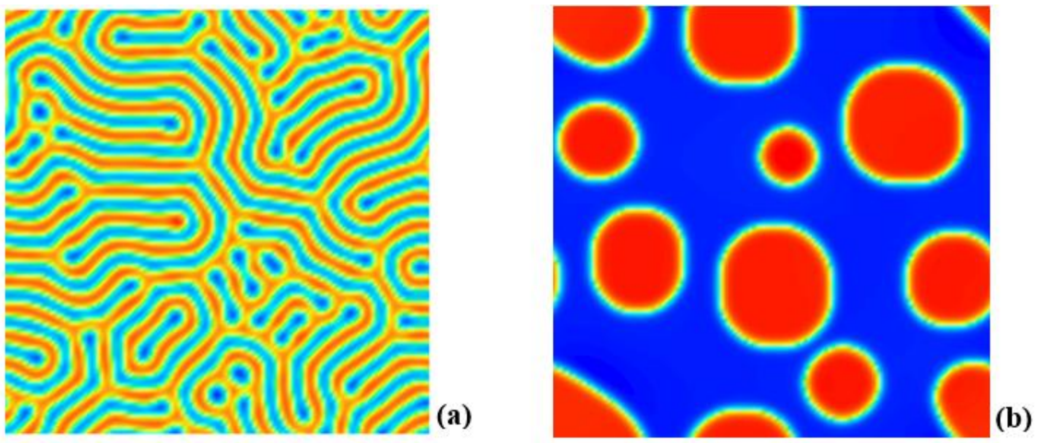


Figure 4.10: CDS results based on Laplacian scheme BV(D2Q9)$_{case2}$ given in equation (3.63); a) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3.

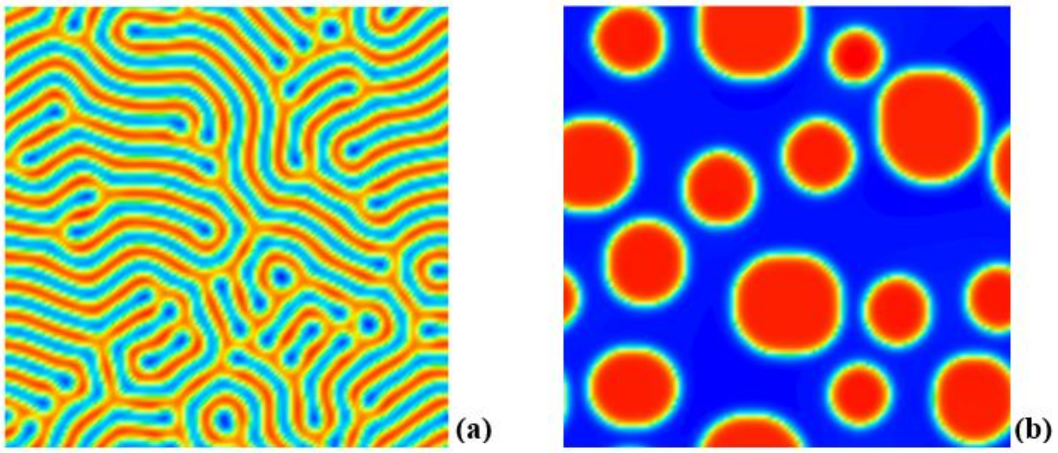


Figure 4.11: CDS results based on Laplacian scheme BV(D2Q9)$_{case3}$ given in equation (3.64); a) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.2; b) real space simulation snapshot at 100000th time step obtained by using parameters given in Table 4.3.

4.2 Three–dimensional simulations

In this section 3D Laplacian schemes were employed in CDS in order to investigate their isotropic or anisotropic behaviour in simulations. Here, all the 3D simulations were presented for spherical phase morphology of the *A*–*B* diblock copolymer systems. This morphology was chosen to closely observe the isotropic simulations clearly in 3D. The spherical morphology was investigated using a dynamic self–consistent field (SCF) simulation and it was found that a spherical phase morphology transforms into a hexagonal cylindrical phase [105]. This morphology, under external fields, i.e. shear flow and electric fields, was investigated by Pinna [2] using CDS and due to the computational efficiency of CDS it was possible for them to perform a larger parameter search, and simulate larger boxes for longer time steps than in previous work undertaken with SCF or Molecular Dynamics [105, 106].

Table 4.5 is presented below, before the discussion of 3D simulation based on Laplacian schemes, containing information about the weights used for Laplacian schemes and with isotropic or anisotropic status.

Table 4.5: 3D stencils along with their weights for utilisation in computer code and isotropic or anisotropic status.

Schemes	**Weights**			**Isotropic /Anisotropic**	**Equations**
	c1	*c2*	*c3*		
D3Q7	1/6	0	0	Anisotropic	(3.86)
D3Q15	1/7	0	1/56	Isotropic	(3.87)
D3Q19	1/12	1/24	0	Isotropic	(3.88)
D3Q27	16/152	4/152	1/152	Isotropic	(3.89)
PK(D3Q27)	14/128	3/128	1/128	Isotropic	(3.90)
SO(D3Q27)	6/80	3/80	1/80	Isotropic	(3.91)
BV(D3Q27)	0.0807524	0.0322491	0.0160576	Isotropic	(3.92)

In this section, all 3D simulations were run on a grid of size $75 \times 75 \times 50$ for up to 100000 time steps with grid spacing $h = \Delta x = \Delta y = 1$. These simulations were started from an initial random disordered state (ψ was a random number within the range ± 0.3). It should be noted that snapshots of 3D simulations are presented without any specific time scale. The source code for the implementation of 3D CDS results presented in Figure 4.12 is given in Appendix D.

In Figure 4.12, the images (a), (b), (c) and (d) are shown for different stages of evolution of 3D spheres (spherical particles) in spherical morphology of *A*-*B* diblock copolymers at different time step values. The initial stage of evolution can be observed in Figure 4.12 (a) where the microphase separation takes place. The parameter values for map function have been employed from Table 4.1 for spherical morphology. The simulations in Figure 4.12 were obtained by employing the 3D 27–point Laplacian operator of Shinozaki and Oono’s choice (Laplacian scheme SO(D3Q27)). In this work, the 3D averaging operator scheme SO(D3Q27) is termed as the default CDS operator and the simulation results given in Figure 4.12 based on it are termed as the default CDS results.

It can be observed in Figure 4.12 (a) that the spherical particles start to microphase–separate in a light green colour for one of the components, either *A* or *B* block, in diblock copolymer. Figure 4.12 (c) shows the microphase separation for the spherical morphology in its complete state.

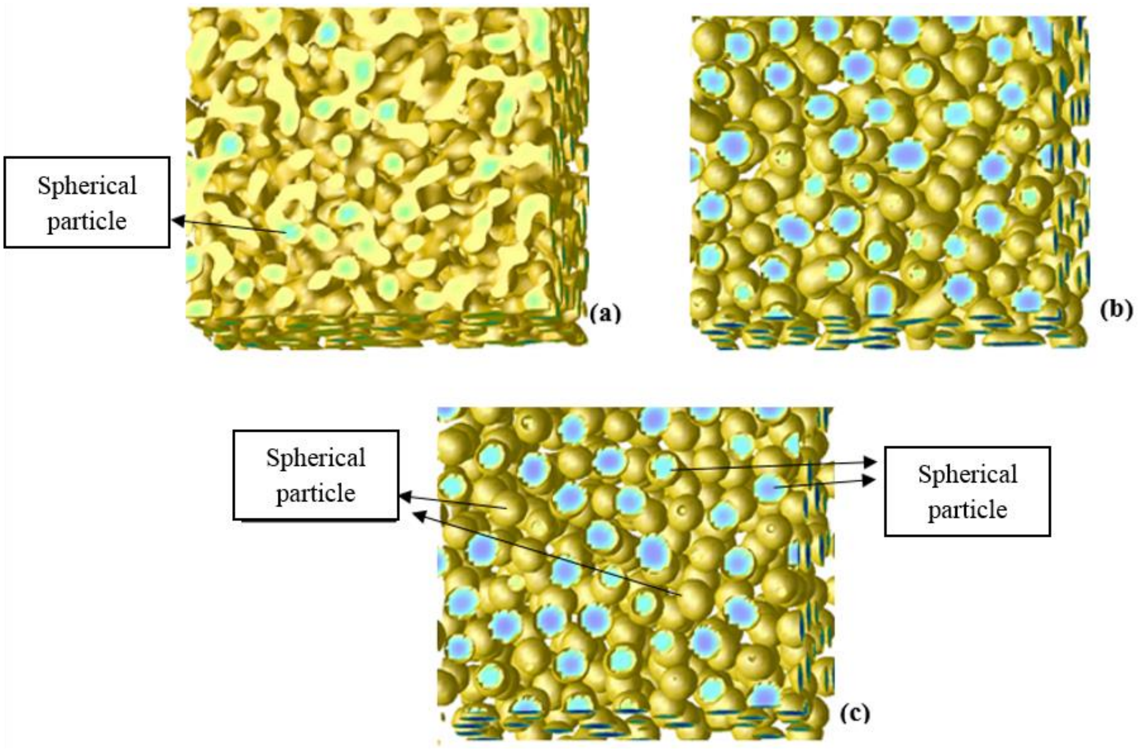


Figure 4.12: CDS results based on Laplacian scheme SO(D3Q27); a) real space simulation snapshot at 100th time step; b) real space simulation snapshot at 10000th time step; c) real space simulation snapshot at 100000th time step.

The simulation results given in Figures 4.13 (a) and (b) were obtained using 3D Laplacian schemes D3Q7 and D3Q15 respectively. The snapshot in Figure 4.13 (a) at the 100000th time step shows that the 3D 7–point Laplacian scheme D3Q7 did not perform well for the simulation of spherical morphology, although the microphase separation can be observed, but the required shapes of the spheres have not been formed perfectly. In Figure 4.13 (b), the defects can be observed in terms of mixed particles and rectangular shapes. It should be noted that a perfect system is one that is stable and does not have any defects as in the CDS default system which is given in Figure 4.12 (c). The simulation result in Figure 4.13 (a) can be compared with the default CDS results given in Figure 4.12. The 3D snapshot (b) of Figure 4.13 at the 100000th time step shows that the 15–point Laplacian scheme D3Q15 also did not perform well for the simulation. The shapes of rectangles can be observed rather than spheres and also the mixed particles can be seen which are considered to be defects in simulations.

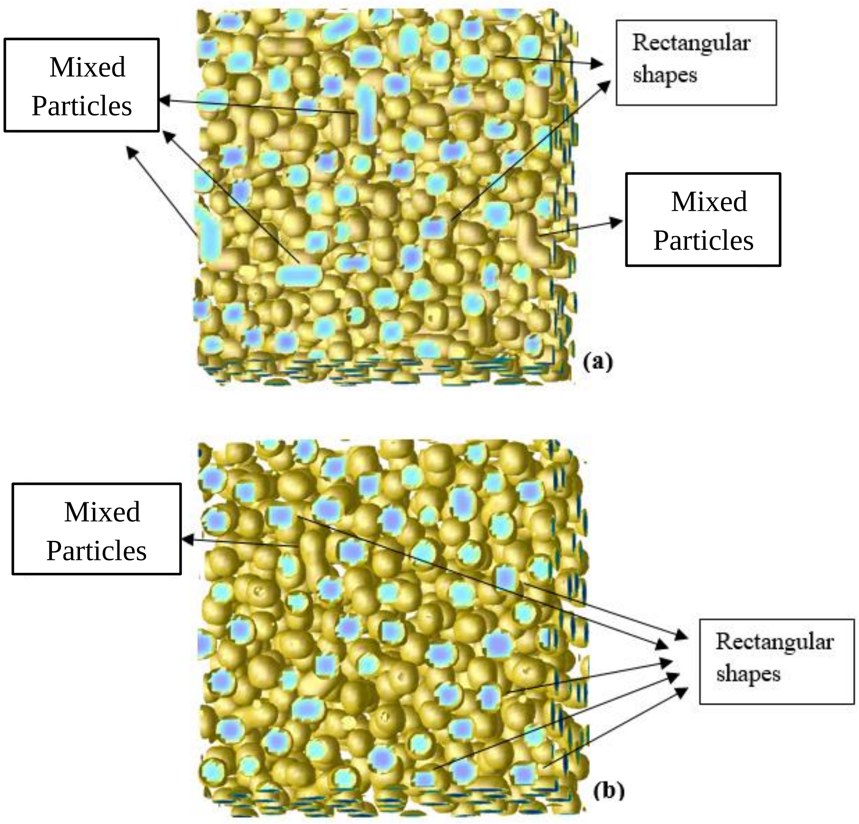


Figure 4.13: 3D CDS results based on Laplacian schemes D3Q7 and D3Q15. a) real space simulation snapshot at 100000th time step using Laplacian scheme D3Q7; b) real space simulation snapshot at 100000th time step using Laplacian scheme D3Q15.

The Laplacian schemes D3Q7 and D3Q15 cannot be considered to be isotropic because the simulation results obtained using these schemes were found to be poor compared to the default CDS 3D results given in Figure 4.12. The source code for the simulations using schemes D3Q7 and D3Q15 were the same as those given in Appendix D for Laplacian scheme SO(D3Q27), just the values for weights were changed.

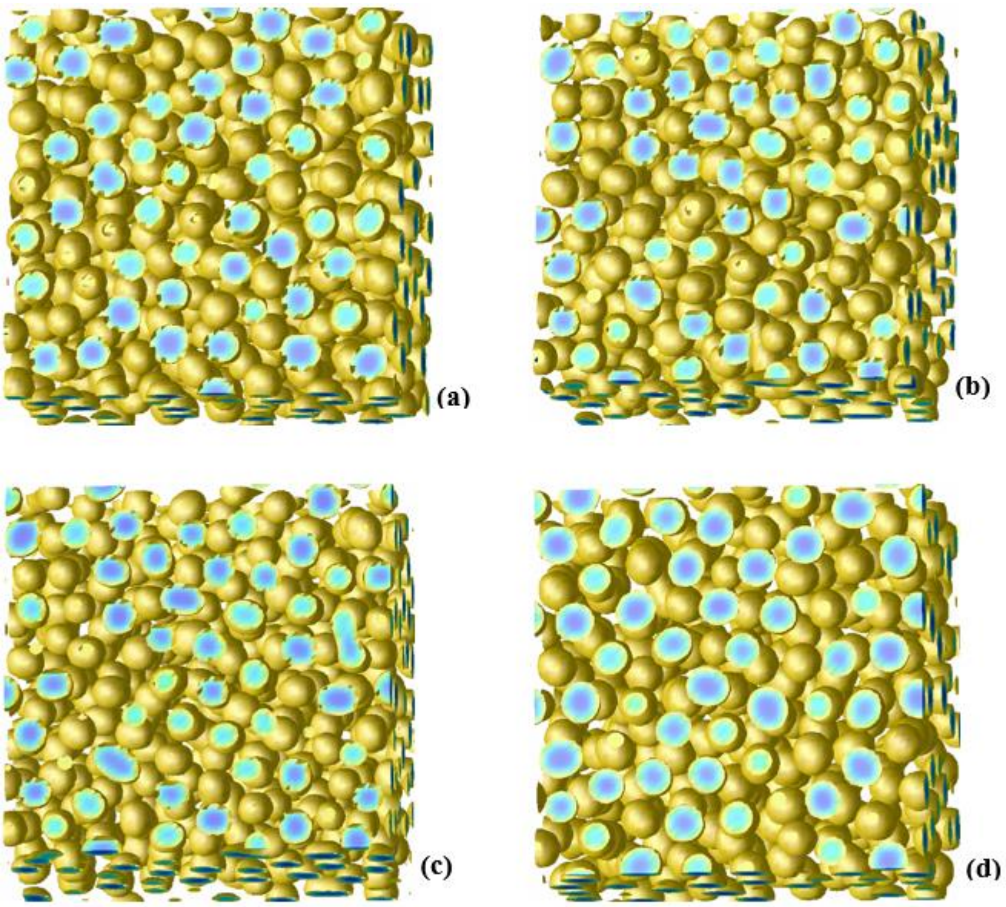


Figure 4.14: 3D results of CDS in spherical morphology for *A-B* diblock copolymer systems. a) Real space simulation snapshot at 100000th time step obtained using Laplacian scheme D3Q19. b) Real space simulation snapshot at 100000th time step obtained using Laplacian scheme D3Q27. c) Real space simulation snapshot at 100000th time step obtained using Laplacian scheme PK(D3Q27). d) Real space simulation snapshot at 100000th time step obtained using Laplacian scheme BV(D3Q27).

The simulation results given in Figure 4.14 (a), (b), (c) and (d) were obtained using 3D Laplacian schemes D3Q19, D3Q27, PK(D3Q27) and BV(D3Q27) respectively. The snapshot in Figure 4.14 (a) at the 100000th time step shows that the 3D 19–point Laplacian scheme D3Q19 performed well for the simulation of spherical morphology; the microphase separation can be observed where the shapes of spheres for blocks are very obvious and can be seen without defects. The snapshot in Figure 4.14 (b) shows perfect sphere shapes without obvious defects; the stencil Laplacian scheme D3Q27 used for this simulation was found to be isotropic. The snapshot in Figure 4.14 (c) shows that the Laplacian scheme performed well overall, but small defects can be observed. It is very clear from Figure 4.14 (d) that the Laplacian scheme BV(D3Q27) performed very well; perfect shapes of spheres can be observed and no defects can be found. The simulation results in Figure 4.14 can be compared with the 3D default CDS results given in Figure 4.12 (c). It must be noted that the 3D Laplacian schemes which were mentioned as isotropic schemes in Chapter 3 have yielded isotropic results. The source code for the schemes D3Q19, D3Q27, PK(D3Q27) and BV(D3Q27) was used the same as that given in Appendix D for Laplacian scheme SO(D3Q27), just the values of weights were changed.

4.3 Conclusions

Simulation results have been presented for the 2D and 3D Laplacian schemes. The snapshots show that Laplacian schemes D2Q5, D2Q9 and BK(D2Q9) are unstable for simulations. The anisotropic two–dimensional 5–point Laplacian A(D2Q5) and 3D 7–point Laplacian D3Q7 did not perform well as compared to isotropic Laplacians. The simulation snapshots obtained by using 2D 9–point Laplacian PK(D2Q9) depicted perfect lamellae formations. The simulations of binary blend using PK(D2Q9) and OP(D2Q9) schemes were found to be similar. The 9–point isotropic stencil operators derived from the B.A.C. van Vlimmeren's method performed similarly to Laplacian scheme OP(D2Q9) for two different parameter systems.

The simulations results obtained by using 2D 9–point star Laplacain scheme $(D2Q9)_{star}$ and 17–point D2Q17 were found badly anistropic for the macrophase sepration. The simulations based on these stencils took longer time for executions compared to isotropic 9–point family Laplcains.

In the 3D simulation results based on 15–point Laplacian scheme D3Q15, the rectangular shapes and mixed particles were found to be as defects for spherical morphology and due to the behaviour of scheme D3Q15, the results were considered anisotropic. The 3D 19–point Laplacian scheme D3Q19 is considered more compact due to the fewer stencil points and it was found with optimal isotropy. The Laplacian scheme D3Q19 performed well for the simulation of spherical morphology; the shapes of the spheres were found to be perfect. In three-dimensional simulations, 27–point based Laplacian schemes produced good results for the evolution of order parameter for spherical morphology. The Laplacian scheme BV(D3Q27) obtained by the method used by B.A.C. van Vlimmeren produced the isotropic results compared to the original CDS Laplacian scheme SO(D3Q27).

Chapter Five

5 Implementation of the Crank–Nicolson method for CDS equations

In this chapter, the main objective is to achieve the implementation of the CN method for the CDS equations. Originally, the CDS technique was employed using the forward Euler method in the literature [2, 4, 41, 49]. However, the forward Euler method is not stable, whereas the CN method is more stable and second–order accurate in time and space. Three different Finite Difference (FD) methods are analysed and implemented for the CDS equations. All the FD methods, including explicit and implicit schemes, are incorporated for modelling the lamellar forming of *A*–*B* diblock copolymer systems. The computer codes of all three FD methods for CDS were developed by following the same algorithm of eight steps, which was given at the beginning of Chapter 4. The matrix based approach has been adopted for the forward Euler method in the computer program, which is different from the conventional approach. Initial programs of FD methods for CDS are discussed without incorporating the boundary conditions and are based on the basic 5–point formula of Laplacian operator. The techniques are then extended to include boundary conditions and a better isotropic Laplacian scheme based on a 9–point stencil. The results obtained from the backward Euler and CN methods are compared with those of the forward Euler method.

5.1 Implementation of the Crank–Nicolson (CN) Scheme in cell dynamics

Before going to the implementation of CN for CDS, two other methods are implemented for CDS: the matrix-based forward Euler method and the backward Euler method. The description of these methods including CN method is given in section 2.6 of Chapter two for the model heat diffusion equation.

Shown here are the steps needed to solve the time–dependent Ginzburg–Landau (TDGL) equation, based on a finite difference technique. First, equation (3.10) can be re–written in a simple form of PDE as [2]:

$$\frac{\partial \psi}{\partial t} = -\{\nabla^2 (g(\psi) + D\nabla^2 \psi) + B\psi\}. \tag{5.1}$$

with ψ the spatial order parameter, *t* the time, ∇^2 the Laplacian of function of free energy, and *D* as a diffusive parameter. Equation (5.1) is non–linear and fourth order, including the *bi*–Laplacian or biharmonic operator ∇^4.

5.1.1 Matrix based forward Euler and backward Euler methods for CDS

At the first step, equation (5.1) is approximated in the matrix-based explicit forward Euler method without considering the periodic boundary conditions; later the periodic boundary conditions are considered. The derivation of equation (5.1) is carried out in the form of $x = Mb$, where *M* is a symmetric and positive definite matrix. This experiment is carried out for the two–dimensional equation (5.1). Here equation (5.1) is rewritten as:

$$\frac{\partial \psi}{\partial t} = -\nabla^2 g(\psi) - \nabla^2 (D\nabla^2 \psi) - B\psi\,, \tag{5.2}$$

where $g(\psi)$ is the so-called map function as given by [2]:

$$g(\psi) = \left[1 + \tau - A(1-2f)^2\right]\psi - v(1-2f)\psi^2 - u\psi. \tag{5.3}$$

Writing equation (5.2) in the form of $n+1$ and *n* space, the resulting equation is given as:

$$\psi_{j,k}^{n+1} = \underbrace{\psi_{j,k}^{n} - \Delta t D\nabla^2(\nabla^2)\psi_{j,k}^{n}}_{part1} - \underbrace{\Delta t B\psi_{j,k}^{n} - \Delta t\nabla^2 g(\psi_{j,k}^{n})}_{part2}. \tag{5.4}$$

Equation (5.4) is a non–homogenous partial differential equation where the homogenous terms are given in part 1 and the non–homogenous term is given in part 2. The homogenous part is brought into matrix M and is used to evaluate the independent variable $\psi^{n}_{j,k}$ in the first instance. The last non–homogeneous term in equation (5.4) containing $g(\psi)$ is evaluated at a second stage and in this way the values for $\psi^{n+1}_{j,k}$ are approximated.

The five–point Laplacian operator ∇^2 [98] is used for the evaluation of the map function which is also used in earlier results of an order parameter. The five–point formula is given below:

$$\nabla^2\psi^{n}_{j,k} = \frac{1}{h^2}\left[\psi^{n}_{j+1,k} + \psi^{n}_{j-1,k} + \psi^{n}_{j,k+1} + \psi^{n}_{j,k-1} - 4\psi^{n}_{j,k}\right] + O(h^2). \tag{5.5}$$

To apply the five–point formula in cell dynamics equations, according to the averaging conditions $\langle\langle X\rangle\rangle - X$ and the five–point formula, it takes the following form:

$$\nabla^2\psi^{n}_{j,k} = \frac{1}{4}\left[\psi^{n}_{j+1,k} + \psi^{n}_{j-1,k} + \psi^{n}_{j,k+1} + \psi^{n}_{j,k-1}\right] - \psi^{n}_{j,k}\,. \tag{5.6}$$

$$\nabla^2\psi^{n}_{j,k} = \frac{1}{4}\sum_{NN}\psi^{n}_{j,k} - \psi^{n}_{j,k}\,. \tag{5.7}$$

Now equation (5.7) is given a matrix form without periodic boundary conditions, say for a $n\times n$ grid.

$$M1 = \begin{bmatrix} -1 & 1/4 & 0 & \cdots & 0 \\ 1/4 & -1 & 1/4 & \ddots & 0 \\ 0 & 1/4 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1/4 \\ 0 & \dots & 0 & 1/4 & -1 \end{bmatrix}, \quad M2 = \begin{bmatrix} 1/4 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1/4 \end{bmatrix},$$

$$M3 = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad M = \begin{bmatrix} M1 & M2 & M3 & \cdots & M3 \\ M2 & M1 & M2 & \ddots & M3 \\ M3 & M2 & \ddots & \ddots & M3 \\ \vdots & \ddots & \ddots & \ddots & M2 \\ M3 & \cdots & M3 & M2 & M1 \end{bmatrix}.$$

The block matrix *M* is a symmetric and has a positive definite of size $n^2 \times n^2$ and can be generalized for any grid size and also the sub–matrices. In equation (5.4), there is a biharmonic operator $\nabla^2(\nabla^2) = \nabla^4$ in the second term on the right hand side which can be discretized in the thirteen–point stencil [98], so let $C = \nabla^4$ and *C* is taken as $C = M \times M$. The block matrix *C* is also symmetric and positive definite of size $n^2 \times n^2$ comprising sub–matrices as with those of *M* and can be generalized for any grid size. The two-dimensional thirteen–point stencil formula for the biharmonic operator formatted in *C* is given below [98, 107, 108]:

$$\nabla^4(\psi_{j,k}) = \frac{\partial^4}{\partial x^4}(\psi_{j,k}) + 2\frac{\partial^4}{\partial x^2 \partial y^2}(\psi_{j,k}) + \frac{\partial^4}{\partial y^4}(\psi_{j,k}) + O(h^2) + O(h^2), \tag{5.8}$$

where the spatial discretization is given as follows:

$$\frac{\partial^2}{\partial x^2}(\psi_{j,k}) = \frac{\psi_{j+1,k} + \psi_{j-1,k} - 2\psi_{j,k}}{h^2} + O(h^2), \tag{5.9}$$

$$\frac{\partial^4}{\partial x^4}(\psi_{j,k}) = \frac{\psi_{j+2,k} - 4\psi_{j+1,k} + 6\psi_{j,k} - 4\psi_{j-1,k} + \psi_{j-2,k}}{h^4} + O(h^2), \tag{5.10}$$

and similarly for $\frac{\partial^2}{\partial y^2}(\psi_{j,k})$ and $\frac{\partial^4}{\partial y^4}(\psi_{j,k})$.

Equation (5.4) takes the following form:

$$\psi_{j,k}^{n+1} = \underbrace{(I - \Delta tBI - \Delta tDC)\psi_{j,k}^{n}}_{Part1} - \underbrace{\Delta tAf(\psi_{j,k}^{n})}_{Part2} \ . \tag{5.11}$$

In part 1 of the equation (5.11), *I* is the identity matrix containing 1s on its main diagonal of size *M* matrix and $\psi_{j,k}^{n}$ is treated as the vector, having initial random values. First the homogenous part 1 of equation (5.11) is evaluated and then the non–homogenous part 2 of equation (5.11) is evaluated, which is the map function. Thus, the order parameter is evaluated as a whole.

For the explicit forward Euler method, the CDS technique was implemented in Fortran 77 programming language. All simulations were performed for a lamellar forming diblock copolymer system of block copolymers using the parameters for the map functions given in Table 5.1. These parameters are suggested for a lamellar forming system of diblock copolymers [2] and all the two-dimensional simulations in this chapter are based on these parameters.

Table 5.1: System parameters used in cell dynamical method for lamellae morphology

CDS Parameters	τ	f	u	v	B	D	A
Lamellae Morphology	0.36	0.48	0.38	2.3	0.02	0.7	1.5

The simulation results given in Figure 5.1 were obtained based on the following specifications:

- The grid size chosen was 64×64 with grid spacing $\Delta x = \Delta y = 1$;
- The total time of the simulations was up to 10000 time steps with the time interval $\Delta t = 0.1$;

- The simulations were run without periodic boundary conditions;
- A matrix-based approach was used for calculating the Laplacian;
- The simulations were started from an initial random disordered state $\psi = \pm 0.3$.

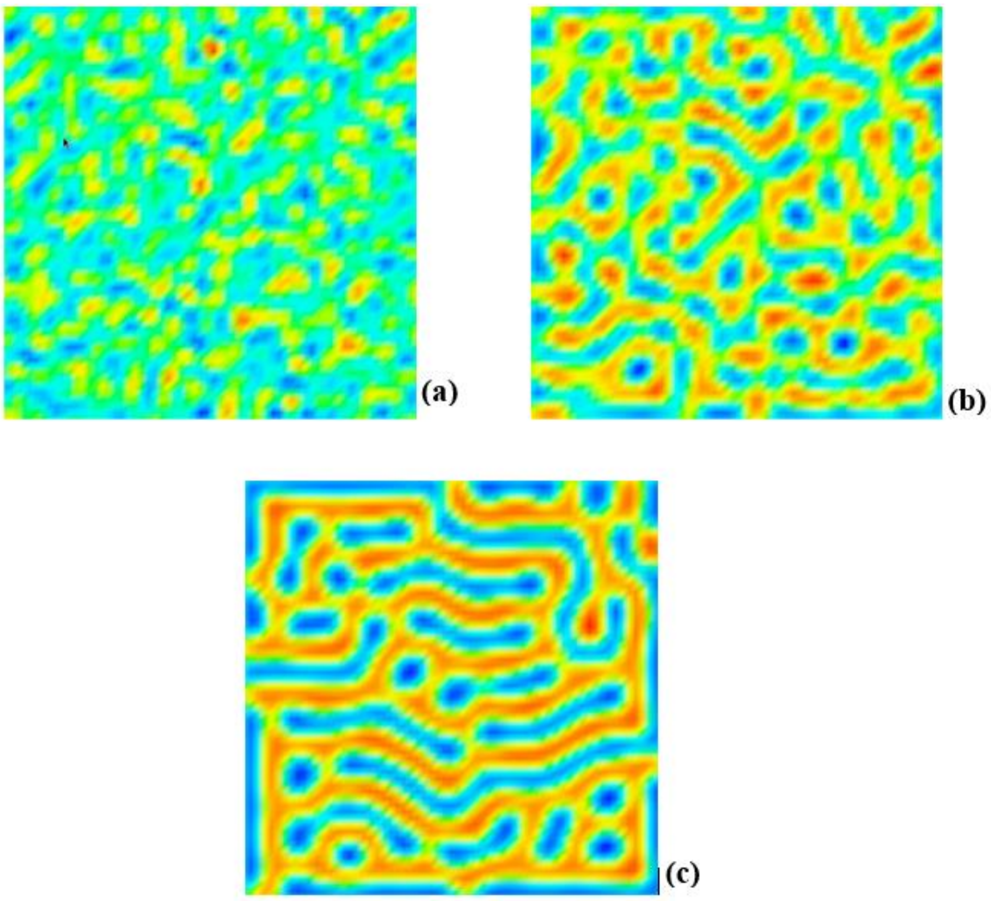


Figure 5.1: Forward Euler method for two-dimensional CDS equations based on 5–point formula at different time steps. a) at t = 10; b) t = 100; c) t = 1000.

In Figure 5.1 (a), (b) and (c), the images are shown for different stages of evolution of lamellae in a lamellar morphology of *A*-*B* diblock copolymer systems at different time steps. The order parameter evolution takes place successfully, which can be observed from the simulation results given in Figure 5.1 (a) and (b). In Figure 5.1 (c), the microphase separation can be observed and the lamellae formations can be seen but the absence of boundary conditions is clearly noticeable. The boundaries are covered with a blue colour, which means that the lamellae do not appear from the other side. Conclusively, the simulation results shown in Figure 5.1 were obtained without employing boundary conditions for the explicit forward Euler method; this method worked initially for the minimum specifications. In the next step the same results are produced using the implicit backward Euler method in order to proceed to the CN method.

Let us write equation (5.4) in the implicit BTCS method:

$$\psi_{j,k}^{n+1} + \Delta t D \nabla^2 (\nabla^2) \psi_{j,k}^{n+1} + \Delta t B \psi_{j,k}^{n+1} + \Delta t \nabla^2 f(\psi_{j,k}^{n+1}) = \psi_{j,k}^{n} \tag{5.12}$$

$$\underbrace{(I + \Delta t B I + \Delta t D C)\psi_{j,k}^{n+1}}_{Part1} + \underbrace{\Delta t A f(\psi_{j,k}^{n+1})}_{Part2} = \psi_{j,k}^{n} \tag{5.13}$$

Equations (5.12) and (5.13) are written in the implicit backward Euler method [8, 69] which is well described in section 2.6.2 of Chapter two for the model heat diffusion equation. The terms are shown on the left hand side to evaluate in $n+1$ space, see equation (2.11). The calculation is carried out in a way that part 1 (homogenous) in the equation (5.13) is evaluated first to identify the values of $\psi_{j,k}^{n+1}$ from $\psi_{j,k}^{n}$ and then part 2 (non–homogenous) is evaluated with new values of $\psi_{j,k}^{n+1}$. Thus, the order parameter is obtained.

To solve the linear system of equations given in equation (5.13), iterative methods need to be used for which LU decomposition was tried. LU decomposition worked very slowly and could be better for lower grids. The Conjugate Gradient (CG) method was used to get faster results. CG outperforms Jaccobi, Gauss–Seidel and Successive Over Relaxation (SOR) for large systems. Good results can be obtained in $\sqrt{N}$ steps of iteration [69]. The solution of the PDE is related to a solution $Mx = b$ system, that is $x = M^{-1}b$, where x is an unknown vector, *b* is a known vector and *M* is known. The lemma in the case of CG is that $Ax = b$ is equivalent to a quadratic minimization problem of the form.

$$f(x) = \frac{1}{2} x^T M x - x^T b \tag{5.14}$$

with $f \in R^n$. The minimum is reached when $x = M^{-1}b$. The method is effective for symmetric positive systems [109]. Memory usage is low with this method since only a small number of vectors are required. For the iterates, the residual vectors are also the gradients of a quadratic functional, the minimization of which is equivalent to solving the linear system [109]. Per iteration, on the matrix–vector product, three vector updates and two inner products were solved [109]. The CG algorithm is given in Appendix E.

The simulation results in Figure 5.2 were obtained using the implicit backward Euler method. The parameters and specifications are the same as those which were used for the explicit forward Euler method for simulation in Figure 5.1. It can be observed from Figure 5.2 that the evolution of an order parameter was successful using the implicit backward Euler method. The simulation results in Figure 5.2 show the same modelling of diblock copolymers as in Figure 5.1. The microphase separation can be clearly observed in Figure 5.2 (c). In section 2.6.2 of Chapter two, it is shown that this method is unconditionally stable. Comparatively, the implicit backward Euler method is preferable because the time interval Δt does not need to have any specific choice for its value. This property of the backward Euler method makes this scheme more preferable to use compared to the explicit forward Euler method.

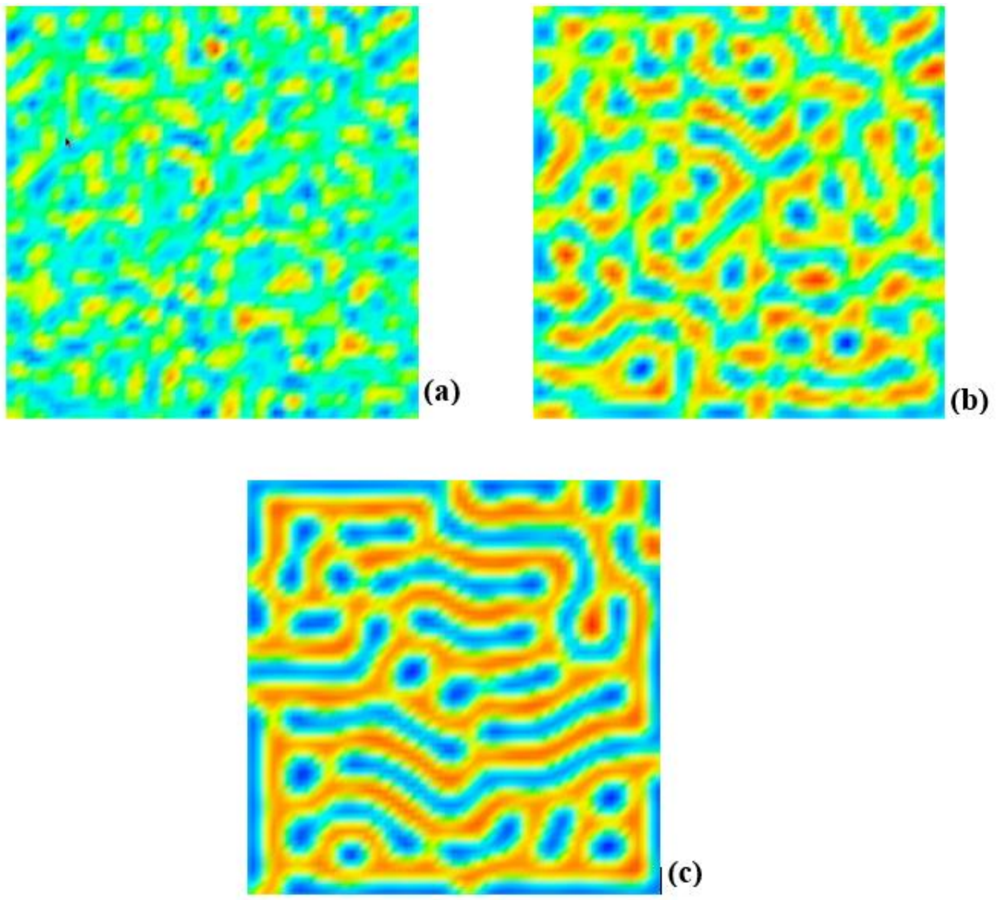


Figure 5.2: Backward Euler method for two-dimensional CDS equations based on 5–point formula at different time steps. a) at t = 10; b) t = 100; c) t = 1000.

5.1.2 Crank–Nicolson method for CDS

The CN method is convenient, particularly for one or two dimensions; however for two or more dimensions, the Alternating Direction Implicit method (ADI) [84] is favoured due to the simpler equations needing to be solved and hence faster results are obtained. The ADI method splits the finite difference equation into two implicit equations that result in a system of symmetric and tri–diagonal equations suitable for e.g. Cholesky decomposition or LU decomposition.

The main objective is to implement the CN scheme and to acquire results for a partial differential equation (5.4). It was discussed in section 2.6.3 of Chapter two that the CN method is the average of the explicit forward Euler and the implicit backward Euler methods and is also unconditionally stable. So accordingly, equation (5.4) is written in the CN method and is given as follows:

$$\begin{aligned}&\psi_{j,k}^{n+1}+\frac{\Delta tD}{2}\nabla^2(\nabla^2)\psi_{j,k}^{n+1}+\frac{\Delta tB}{2}\psi_{j,k}^{n+1}+\frac{\Delta t}{2}\nabla^2 f(\psi_{j,k}^{n+1})\\&=\psi_{j,k}^{n}-\frac{\Delta tD}{2}\nabla^2(\nabla^2)\psi_{j,k}^{n}-\frac{\Delta tB}{2}\psi_{j,k}^{n}-\frac{\Delta t}{2}\nabla^2 f(\psi_{j,k}^{n})\end{aligned}. \qquad (5.15)$$

$$\begin{aligned}&\underbrace{\left(I+\frac{\Delta tB}{2}I+\frac{\Delta tD}{2}C\right)\psi_{j,k}^{n+1}}_{Part1}+\underbrace{\frac{\Delta t}{2}Af(\psi_{j,k}^{n+1})}_{Part2}\\&=\underbrace{\left(I-\frac{\Delta tB}{2}I-\frac{\Delta tD}{2}C\right)\psi_{j,k}^{n}}_{Part1}-\underbrace{\frac{\Delta t}{2}Af(\psi_{j,k}^{n})}_{Part2}\end{aligned}. \qquad (5.16)$$

Equation (5.16) is a CN scheme for equation (5.4) where the terms on the right hand side are evaluated by the explicit forward Euler method in *n* space. The left hand side is evaluated from the right hand side values at *n*+1 space for part 1 using the CG method which was used in the implicit backward Euler method. After obtaining the values for an order parameter $\psi_{j,k}^{n+1}$, the nonhomogeneous part 2 of the map function on the left hand side is evaluated based on the new values of the order parameter. Thus, the order parameter is finally obtained. The results are presented in Figure 5.3, but as an experiment these results are without periodic boundary conditions. The images in Figure 5.3 are at the 10th, 100th and 1000th time steps. The simulation results given in Figure 5.3 were obtained using the same parameters and specifications that were used for the simulation of the explicit forward Euler method in this chapter. The simulation results in Figure 5.3 show the modelling of block copolymers in the same way as in Figures 5.1 and 5.3.

To analyse the stability of the CN scheme using the biharmonic operator, consider the homogenous part 1 of equation (5.16) excluding the term with coefficient *B* as follows:

$$\begin{aligned}&\psi_{j,k}^{n+1}+\frac{\Delta tD}{2}\left(\frac{\partial^4}{\partial x^4}+2\frac{\partial^4}{\partial x^2\partial y^2}+\frac{\partial^4}{\partial y^4}\right)\psi_{j,k}^{n+1}\\&=\psi_{j,k}^{n}-\frac{\Delta tD}{2}\left(\frac{\partial^4}{\partial x^4}+2\frac{\partial^4}{\partial x^2\partial y^2}+\frac{\partial^4}{\partial y^4}\right)\psi_{j,k}^{n}.\end{aligned} \tag{5.17}$$

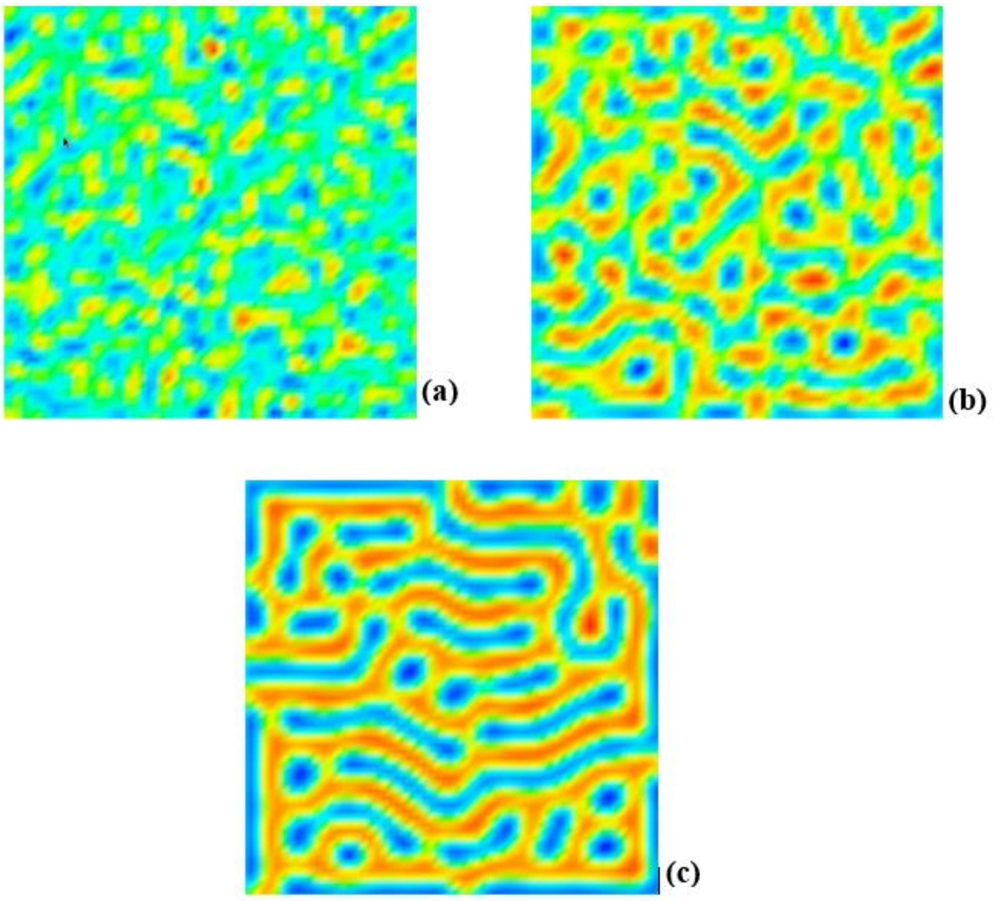


Figure 5.3: CN scheme for two-dimensional CDS equations based on 5–point formula at different time steps. a) at t = 10; b) t = 100; c) t = 1000.

To derive a sufficient condition for stability, the discrete Neumann stability criterion was applied, the discrete Fourier mode [69, 84] in two dimensions,

$$\psi_{j,k}^{n} = \xi^{n} e^{ijp\pi\Delta x} e^{ikq\pi\Delta y}, \tag{5.18}$$

and by inserting equation (5.18) in equation (5.17) and dividing the resulting equation by ξ^{n+1}, gives:

$$\xi^{n} = \frac{\begin{bmatrix} 1-8D\Delta t\sin^{4}(p\pi\Delta x/2)-8D\Delta t\sin^{4}(q\pi\Delta y/2) \\ -16D\Delta t\sin^{2}(p\pi\Delta x/2)\sin^{2}(q\pi\Delta y/2) \end{bmatrix}}{\begin{bmatrix} 1+8D\Delta t\sin^{4}(p\pi\Delta x/2)+8D\Delta t\sin^{4}(q\pi\Delta y/2) \\ +16D\Delta t\sin^{2}(p\pi\Delta x/2)\sin^{2}(q\pi\Delta y/2) \end{bmatrix}}, \tag{5.19}$$

and from the Fourier analysis we get $|\xi| \le 1$ for all non–negative $D\Delta t$, so the two–dimensional Crank–Nicolson scheme employing the CDS method is unconditionally stable. The stability analysis can be applied in the same way for the explicit forward Euler method and implicit backward Euler method.

5.1.3 Implementation of boundary conditions

The results presented before are without the use of boundary conditions. Here the discussion is elaborated taking the boundary conditions into account. The boundary conditions are very important in computer simulations if these simulations are producing images. The CDS equations employ periodic boundary conditions. The periodic boundary conditions (PCBs) are used to avoid problems with boundary effects caused by finite size, and make the system more like an infinite one, at the cost of possible periodicity effect. The existence of PBC means that any object (atom or molecule) that leaves a simulation box by, say, the right-hand face, and then enters the simulation box by the left-hand face or vice versa. How the periodic boundary conditions are implemented in the matrix of the Laplacian is explained here. Consider the five–point formula for 5×5 grid:

$$\nabla^2_{5p}\psi_{j,k} = \psi_{j+1,k} + \psi_{j-1,k} + \psi_{j,k+1} + \psi_{j,k-1} - 4\psi_{j,k},\ j,k = 1,2,\cdots,5 \tag{5.20}$$

Let *ux, vx, uy and vy* be the array of indexes such that the last entry in *ux* is 1 and the first entry in *vx* is 5 and the same for the *uy* and *vy*. In the following way:

$$ux = uy = (2,3,4,5,1),\ \text{ and } vx = vy = (5,1,2,3,4). \tag{5.21}$$

The equation (5.20) takes the following form:

$$\nabla^2_{5p}\psi_{j,k} = \psi_{ux(j),k} + \psi_{vx(j),k} + \psi_{j,uy(k)} + \psi_{j,vy(k)} - 4\psi_{j,k},\ \ j,k = 1,2,\cdots,5 \tag{5.22}$$

When the matrix of Laplacian is constructed there is a small change in the sub–matrix *M1*. The sub–matrices *M2* and *M3* remain the same and the change in *M1* obviously causes the change of values in *M* matrix and so the C. The matrices *M1* and *M* are given below:

$$M1 = \begin{bmatrix} -1 & 1/4 & 0 & \cdots & 1/4 \\ 1/4 & -1 & 1/4 & \ddots & 0 \\ 0 & 1/4 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1/4 \\ 1/4 & \ldots & 0 & 1/4 & -1 \end{bmatrix},\ M = \begin{bmatrix} M1 & M2 & M3 & \cdots & M1 \\ M2 & M1 & M2 & \ddots & M3 \\ M3 & M2 & \ddots & \ddots & M3 \\ \vdots & \ddots & \ddots & \ddots & M2 \\ M1 & \cdots & M3 & M2 & M1 \end{bmatrix}.$$

The results based on periodic boundary conditions are given in Figure 5.4 for the explicit forward Euler method.

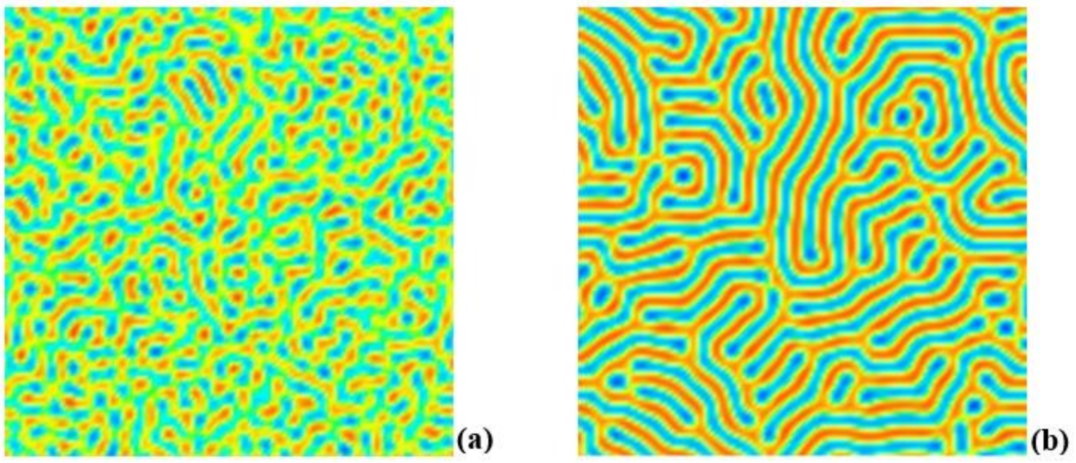


Figure 5.4: Explicit forward Euler method based on 5–point formula using periodic boundary conditions where (a) and (b) images are 100th and 10000th time steps.

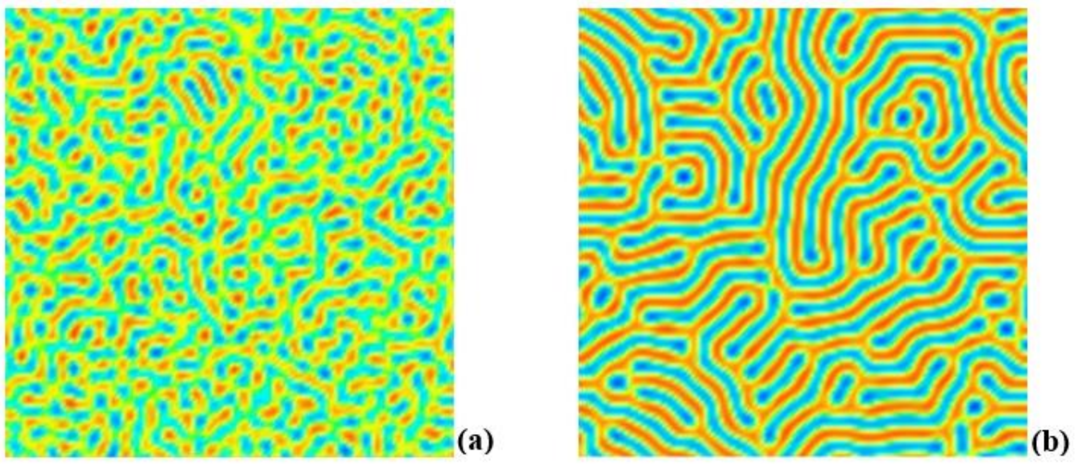


Figure 5.5: Implicit backward Euler method based on 5–point formula using periodic boundary conditions where images (a) and (b) are 100th and 10000th time steps.

The results in Figures 5.4, 5.5 and 5.6 are the images at 100th and 10000th time steps. For the simulation results given in Figures 5.4 and 5.5, the system parameters used are given in Table 5.1 and the other specifications used are given as follows:

- The grid size chosen was 128×128 with grid spacing $\Delta x = \Delta y = 1$;
- The total time of the simulations was up to 10000 time steps;
- The simulations were run with periodic boundary conditions;

- The simulations were started from an initial random disordered state $\psi = \pm 0.3$.

It is emphasized that the value $\Delta t = 0.1$ was used for the explicit forward Euler method. The simulations results given in Figure 5.5 were obtained using the implicit backward Euler method. Here it is clear that the value $\Delta t = 1.0$ does not work in the five–point Laplacian generally in the explicit forward Euler method. The divergence of values takes place immediately on reaching 100th or 1000th time step and so on. When the implicit schemes are employed there is no issue of using $\Delta t = 1.0$ for the five–point Laplacain. The results in Figure 5.6 were obtained successfully using the CN method, which is also unconditionally stable and has a discretization error order $O(\Delta t^2 + h^2)$. In Figures 5.4, 5.5, and 5.6 the simulation results are shown for different stages of evolution of lamellae in a lamellar morphology of *A*-*B* diblock copolymer systems at different time steps. The order parameter evolution takes place successfully, which can be observed from the simulation results given in snapshot (a) of these Figures; snapshot (c) of these Figures shows the microphase separation and the lamellae formations can also be seen.

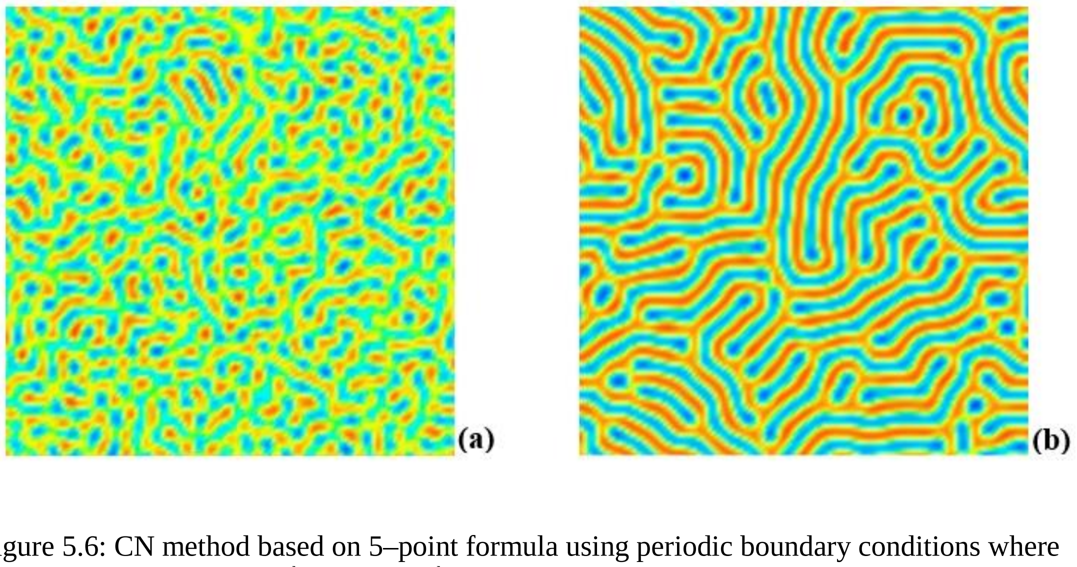


Figure 5.6: CN method based on 5–point formula using periodic boundary conditions where images (a) and (b) are 100th and 10000th time steps respectively.

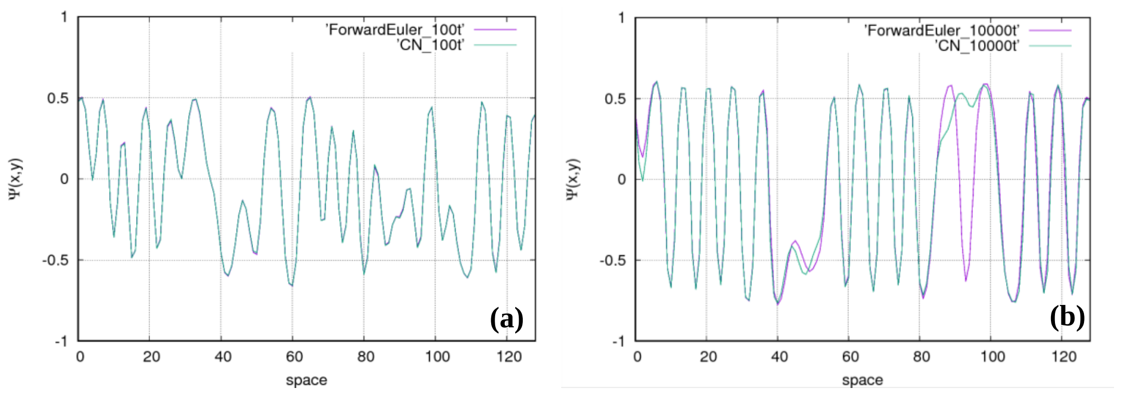


Figure 5.7: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 100th and 10000th time steps in images (a) and (b) respectively. Two lines of different colour show numerical values of $\psi(x)$ for two different methods, the forward Euler and the CN. The numerical values plotted here were obtained from the simulations shown in Figure 5.4 and 5.6.

The Figure 5.7 displays the profile of $\psi(x,y)$ against the space (0 – 128). This space is assumed to be either *x*- or *y*-axis. The range of the space is fixed 0 – 128 because the domain size was chosen 128 x 128. The scale of plot in this Figure is set in such a way that on the vertical axis, $\psi(x,y)$ represents the numerical values in the range ±1 against space (0 – 128) on the horizontal axis. In order to compare the two finite difference methods, the numerical values of order parameter are plotted in Figure 5.7 at different time steps. In Figure 5.7 (a), the values are compared at 100th time step and in Figure 5.7 (b) the values are compared at 10000th time step. The simulation images of these time steps are shown in Figure 5.4 and Figure 5.6. The two lines coloured pink and green show the forward Euler method and the CN method respectively for comparison. In Figure 5.7 (a), both lines are exactly parallel, which illustrates that the tendency of numerical values obtained from the two methods is the same while the methodologies are different. Figure 5.7 (b) shows that the numerical values differ between 80 and 100 on the horizontal axis but the two lines are matching except in this region. The parallel distribution of numerical values on the same scale in Figure 5.7 for the two different methods shows that there is no obvious difference for the evolution of order parameter and therefore the lamellae formations are almost similar in pattern, which can be observed by comparing Figure 5.4 and Figure 5.6.

After the successful implementation of the five–point Laplacian in implicit backward Euler and CN methods, isotropic nine–point Laplacian operators were employed, as used by Oono and Puri [3, 47]. This 9–point isotropic Laplacian operator was employed simultaneously for all three finite difference schemes. The 9–point isotropic Laplacian operator was discussed in Chapter 3 as Laplacian scheme OP(D2Q9), which is given below:

$$\Delta_{OP(D2Q9)}\Psi = \frac{1}{6}\sum_{i=1}^{4}\Psi_i^{(1)} + \frac{1}{12}\sum_{i=1}^{4}\Psi_i^{(2)} - \Psi^{(0)}. \tag{5.23}$$

The Laplacian operator given in equation (5.23) was simulated in matrix *M* considering the periodic boundary conditions. The CDS equations were run using this formula with all previously used parameters and configurations on a 128×128 grid size having $\Delta t = 1.0.$ The simulation results are presented with the same pattern in Figures 5.8, 5.9 and 5.10 for explicit forward Euler, implicit backward Euler and CN methods respectively. These simulation results yield the same information about the modelling of *A*-*B* diblock copolymer systems. The order parameter evolution takes place successfully, which can be observed from the simulation results given in snapshot (a) of the Figures 5.8, 5.9 and 5.10. Snapshot (c) of these Figures shows the microphase separation and the lamellae formations can be seen. The source code for CN method is given in Appendix F for the simulations shown in Figure 5.10. Snapshots of all the simulations in Figures 5.8, 5.9 and 5.10 represent a range of values between $\psi = -0.3$ and $\psi = 0.3$ for an order parameter. Two snapshots are shown for 100th and 10000th time steps.

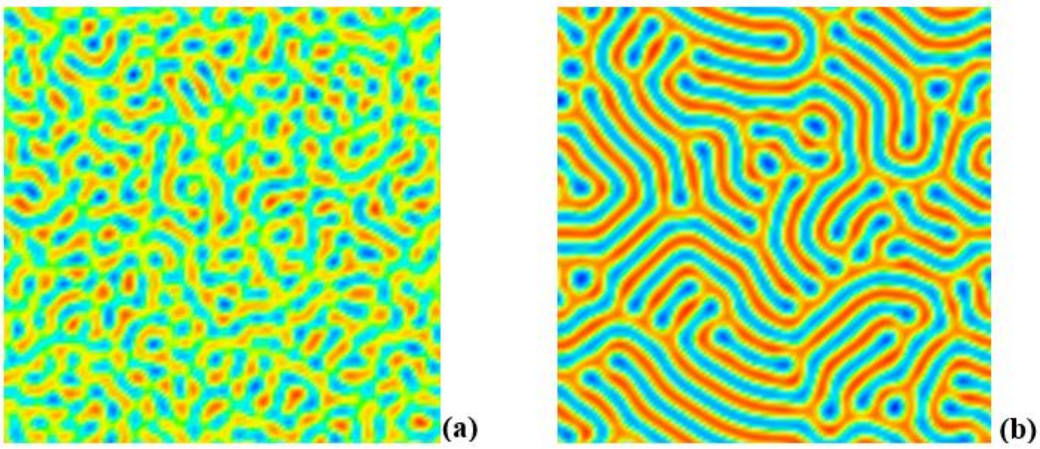


Figure 5.8: Explicit forward Euler method based on 9–point formula using periodic boundary conditions where (a) and (b) images are 100th and 10000th time steps respectively.

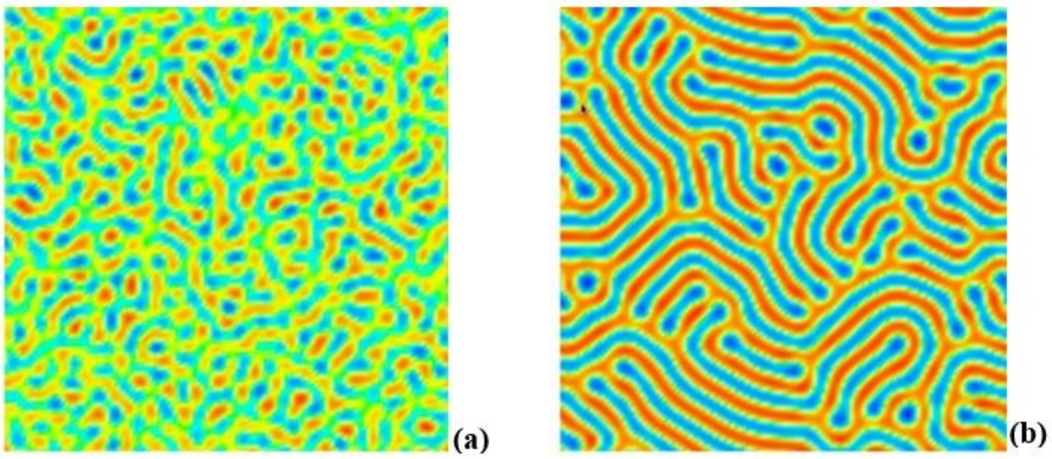


Figure 5.9: Implicit backward Euler method based on 9–point formula using periodic boundary conditions where images (a) and (b) are 100th and 10000th time steps respectively.

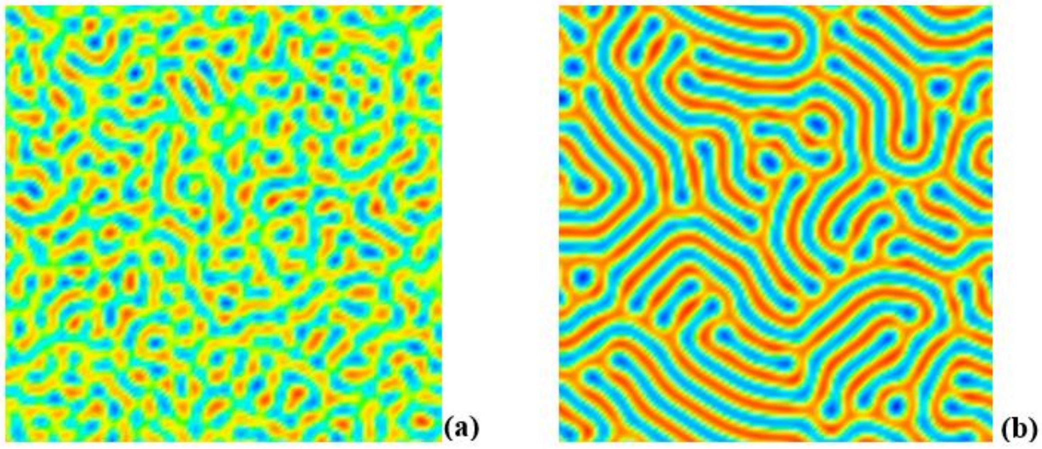


Figure 5.10: CN method based on 9–point formula using periodic boundary conditions where images (a) and (b) are 100th and 10000th time steps respectively.

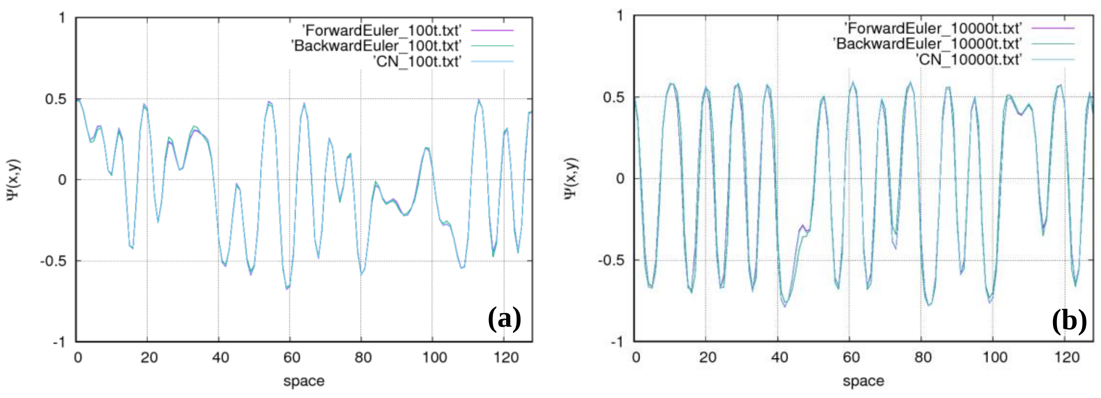


Figure 5.11: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 100th and 10000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler, backward Euler and the CN methods. The numerical values plotted here were obtained from the simulations shown in Figure 5.8, 5.9 and 5.10.

Figure 5.11 is constructed in a similar way to Figure 5.7, where the distribution of numerical values of the order parameter ψ obtained from 2D simulations (Figures 5.8, 5.9 and 5.10) for forward Euler, backward Euler and CN methods are shown for comparison. In Figure 5.11 (a), the values are compared at the 100th time step and in Figure 5.11 (b) the values are compared at the 10000th time step. Three lines of different colours show three different methods and each line represents the numerical values. In the simulations of the three different methods, the 9–point isotropic Laplacian scheme OP(D2Q9) was employed. Three lines are exactly parallel, which shows that the tendency of numerical values obtained from the three different methods is the same, while the methodologies are different. The parallel distribution of the numerical values in Figure 5.11 for the three different methods shows that there is no obvious difference for the evolution of the order parameter and therefore the lamellae formations are almost similar in pattern, which can be observed by comparing Figure 5.8, Figure 5.9 and Figure 5.10.

5.2 Conclusions

The cell dynamics simulations were used to carry out calculations for a diblock copolymer system using different finite difference schemes. The main objective was achieved via the implementation of the Crank–Nicolson scheme, and the results obtained from the CN method were compared with the explicit forward Euler and the implicit backward Euler methods. The conventional approach of algorithms using the forward Euler method was replaced by more stable schemes, especially the CN method. In the first stage, the conventional explicit scheme was transformed into a matrix-based approach, which made it possible to carry out calculations for the implicit backward Euler method based on the five–point Laplacian operator. The issue of boundary conditions was resolved technically using periodic boundary conditions. The explicit forward Euler method has some limitations for the choice of time interval Δt, and implicit schemes overcame these limitations. For implementing the implicit backward Euler scheme, different algorithms were tried to solve the system of $x = M^{-1}b.$ In this regard, LU decomposition worked very slowly so the Conjugate Gradient method was employed for faster calculations. The implicit schemes used matrices and, for larger two-dimensional grids, i.e. 128x128, the huge sparse matrices were produced up to the squares in each dimension length.

Therefore, in this chapter during the simulations for a CN approach applied to cell dynamics equations it was only possible to use a two-dimensional grid of size 128x128. The whole work was limited to this grid size and two dimensions; for three dimensions the scheme encountered limitations of computer memory due to the huge sparse matrices. Both the schemes, implicit backward Euler and CN are stable, but are very slow in comparison to the conventional forward Euler method. The reason is that huge calculations were carried out between different sparse matrices and vectors. The forward Euler method is very fast but not very stable, which is the disadvantage of this method.

The CN scheme was also employed in the two-order parameter system in cell dynamics equations and results were successfully obtained, which are presented in the next chapter. There is an Alternating Direction Implicit (ADI) method which is fast and very stable and has the same properties as the CN scheme. Three-dimensional results for cell dynamics can be obtained using the ADI scheme.

Chapter Six

6 The Cell Dynamics Simulations of Two Order Parameter Systems

In this chapter, two order parameter systems were investigated using the Cell Dynamics Simulation (CDS) method. The two order parameter systems are comprised of a binary blend which contains an *A–B* diblock copolymer and a solvent *C* homopolymer (*A–B/C* systems). In such systems, the phase separation takes place in two different ways – one is the microphase separation in the *A–B* diblock copolymer and the other is macrophase separation between the *A–B* diblock copolymer and the *C* homopolymer. It must be noted that in Chapter three, the CDS model was presented based on one order parameter evolution for the microphase separation in *A–B* diblock copolymer systems.

In this chapter, the main objective of this study is to implement the Crank–Nicolson finite difference scheme for the model equations of the CDS method, based on two order parameter systems. The implementation of the stable finite difference schemes for CDS based on such systems will be helpful and useful to relieve the anisotropy of the system in the late stage of domain growth which usually arise from the discretization of the space. The implementation of implicit methodologies will make these CDS models more reliable in terms of speed, accuracy and time interval stability. Here, the CDS of two order parameter systems has been implemented in explicit and implicit finite difference methods in two dimensions. The two order parameter systems were discussed in section 2.4 of Chapter two.

6.1 Mathematical model of two order parameter systems

Two order parameter systems are systems where a mixture or a blend contains an *A–B* diblock copolymer and a *C* homopolymer [110]. The phase separation triggers in two different ways: macrophase and microphase separations. In the model of the systems, one independent variable represents the microphase separation that takes place in the *A*–*B* diblock copolymer, and the second independent variable represents the macrophase separation between the *A*–*B* diblock copolymer and the solvent *C* homopolymer.

The polymerization indices of the *A*–*B* diblock copolymer and the *C* homopolymer are N_A, N_B and N_C, respectively. The ϕ_A, ϕ_B and ϕ_C are local volume fractions of *A*, *B* and *C* monomers respectively. The block ratio *f* is defined by [59]:

$$f = \frac{N_A}{\left(N_A + N_B\right)} \tag{6.1}$$

Two order parameter with incompressibility conditions are:

$$\begin{aligned} \phi &= \phi_A - \phi_B \\ \psi &= \phi_A + \phi_B \end{aligned} \tag{6.2}$$

where ϕ is for an order parameter in the microphase separation and ψ is the segregation of copolymer/homopolymer. Ohta introduced a new variable $\eta = \psi - \psi_c$, where ψ_c is the volume fraction at the critical point of the macrophase separation [59]. The model of free energy is presented in short–range and long–range parts for copolymer–homopolymer mixtures in terms of ψ and ϕ [59].

$$F\{\psi,\phi\} = F_S\{\psi,\phi\} + F_L\{\psi,\phi\}. \tag{6.3}$$

The short range part is written as follows:

$$F_S\{\psi,\phi\} = \frac{c_1}{2}\int dr\left(\nabla\psi(r)\right)^2 + \frac{c_2}{2}\int dr\left(\nabla\phi(r)\right)^2 + \int dr\{W(\psi,\phi)\} \tag{6.4}$$

where c_1 and c_2 are positive constants and

$$W(\eta,\phi)=g_1(\eta)+g_2(\phi)+b_1\eta\phi-\frac{b_2}{2}\eta\phi^2-\frac{b_3}{2}\eta^2\phi+\frac{b_4}{2}\eta^2\phi^2 \tag{6.5}$$

are local interactions between η and ϕ where b_1 and b_2 are positive constants. The other constant b_3 vanishes for $f=1/2$ and is taken as $b_3=b_0(1/N_A-1/N_B)$ with b_0 a positive constant. The functions $g_1(\eta)$ and $g_2(\phi)$ are even functions where $g_1(\eta)$ exhibits a double–potential below the macrophase–separation temperature and function $g_2(\phi)$ is not double well in the macrophase–separated state [59]. The term $b_1\eta\phi$ in equation (6.5) is responsible for the short–range interaction between the monomers, if the interaction strength between *i* and *j* monomers is put as $u_{ij}(i,j=A,B,C)$ [59]. Thus, the energy obtained from the short–range interaction is written as:

$$\frac{1}{2}\sum_{i,j}u_{ij}\int dr\phi_i\phi_j \tag{6.6}$$

The constant b_1 is given in terms of u_{ij} by:

$$b_1=\frac{1}{4}(u_{AA}-u_{BB})-\frac{1}{2}(u_{AC}-u_{BC}) \tag{6.7}$$

The third term with factor b_1 in equation (6.5) is the interaction strength [111]. The last term in equation (6.5) arises from the configurational entropy of polymer chains. The fourth term $(-b_2/2)\eta\phi^2$ tells how the microphase–separation takes place only in the copolymer rich phase [111]. The last term in equation (6.5) controls phase separations; the first is the macrophase separation between copolymers and homopolymers and the second is the microphase separation in block copolymers [111]. The long–range free energy arises from long–range interaction in copolymer systems and this interaction is resembled to a coulomb type repulsive interaction in copolymer systems [59, 60, 111].

The model equation of the long–range energy used in a two order parameter system is given below:

$$F_L\{\eta,\phi\} = \iint drdr' G(r,r') \left\{ \begin{array}{l} \frac{\alpha}{2}[\phi(r)-\bar{\phi}][\phi(r')-\bar{\phi}] \\ +\beta[\eta(r)-\bar{\eta}][\phi(r')-\bar{\phi}] \\ +\frac{\gamma}{2}[\eta(r)-\bar{\eta}][\eta(r')-\bar{\eta}] \end{array} \right\} \tag{6.8}$$

where $G(r,r')$ is Green's function, and more clearly it is given by the relation $-\nabla^2 G(r,r') = \delta(r-r')$ and $\delta\phi(r) = \phi(r) - \bar{\phi}$. The symbols $\bar{\phi}$ and $\bar{\eta}$ represent the spatial average of ϕ and η respectively [59, 111]. Here the definition of α, β and γ is given as follows [59]:

$$\alpha = a\left[\frac{1}{N_A}+\frac{1}{N_B}\right]^2, \tag{6.9a}$$

$$\beta = a\left[\frac{1}{N_A^2}-\frac{1}{N_B^2}\right]^2, \tag{6.9b}$$

$$\gamma = b\left[\frac{1}{N_A}-\frac{1}{N_B}\right]^2, \tag{6.9c}$$

where *a* is positive constant. Using the above free energy *F*, the dynamical model of phase separation in a copolymer–homopolymer mixture is modelled by the following set of equations in terms of ϕ and ψ [59, 111]:

$$\begin{aligned} \frac{\partial\psi}{\partial t} &= L_1\nabla^2 \frac{\delta F\{\eta,\phi\}}{\delta\psi} \\ \frac{\partial\phi}{\partial t} &= L_2\nabla^2 \frac{\delta F\{\eta,\phi\}}{\delta\phi} \end{aligned} \tag{6.10}$$

where the transport coefficients $L_1 = L_2 = 1$ are positive.

6.2 Numerical method for two order parameter systems

The cell dynamical equations for corresponding partial differential equation (PDE) in equation (6.10) are given as follows [111, 112]:

$$\eta(t+\Delta t;i,j) = \eta(t;i,j) + \langle\langle T_\eta \rangle\rangle - T_\eta(t;i,j), \tag{6.11}$$

$$\phi(t+\Delta t;i,j) = \phi(t;i,j) + \langle\langle T_\phi \rangle\rangle - T_\phi(t;i,j) - \alpha[\phi(t;i,j) - \bar{\phi}], \tag{6.12}$$

where:

$$T_\eta(t;i,j) = -c_1(\langle\langle \eta \rangle\rangle - \eta) - A_1 \tanh\eta + \eta + b_1\phi - \frac{1}{2}b_2\phi^2 - b_3\eta\phi + b_4\eta\phi^2, \tag{6.13}$$

$$T_\phi(t;i,j) = -c_2(\langle\langle \phi \rangle\rangle - \phi) - A_2 \tanh\phi + \phi + b_1\eta - b_2\eta\phi - \frac{1}{2}b_3\eta^2 + b_4\eta^2\phi, \tag{6.14}$$

and the Laplacian for quantity *X* is given as follows:

$$\langle\langle X \rangle\rangle = W_1 \sum_{NN} X + W_2 \sum_{NNN} X. \tag{6.15}$$

where *NN* and *NNN* represent nearest–neighbours and next–nearest neighbours respectively, and *Ws* are weights, i.e. $W_1 = 1/6$ and $W_2 = 1/12$ [3]. The form of the local interactions $dg_i(x)/dx = -A_i \tanh x + x$ (*i*=1 and 2) with the coefficients $A_1 = 1.3$ and $A_2 = 1.1$ are given [59, 111]. The lamellar forming system is chosen for the two order parameter system simulations to understand the essence of phase–separation phenomena of *A–B/C* systems where a lamellar structure of diblock copolymer is expected to appear in the macrophase–separated phase. Therefore, the case of $f = 1/2$ or $(N_A = N_B)$ is considered and this value of *f* makes the term absent identically with coefficient b_3 in equation (6.5). For the absence of this term, the coefficient $b_3 = 0$ is set in equations

(6.13) and (6.14). Also, the higher–order term in equation (6.5) with coefficient b_4 is not considered for first simulations in this chapter because the main objective is to understand the domain growth in *A–B/C* systems and therefore the coefficient $b_4 = 0$ is set. Later in this chapter the higher order term in equation (6.5) with coefficient b_4 is included for the simulations.

6.2.1 Computer simulations

In this section the computer simulations are presented; the steps for the computer program were set up based on discrete equations (6.11)–(6.14), which are given as follows:

1. Assigning random initial values to variables representing two order parameters ψ and ϕ;
2. Setting the periodic boundary conditions, i.e. x, y and z directions;
3. Calculation of first discrete Laplacian for order parameters, i.e. $\langle\langle\psi\rangle\rangle - \psi$;
4. Calculation of the map function (6.13) and (6.14) based on the order parameters with initial random values and other constants with specific values;
5. Calculation of second (outer) discrete Laplacian for the result of step 4;
6. Calculation of order parameters based on new obtained values with respect to time evolution.

The simulation results shown in Figure 6.1 were based on the following specifications: The grid size chosen was 128×128 with grid spacing $\Delta x = \Delta y = 1,$ the total time of the simulations was up to 50000 time steps with time interval $\Delta t = 0.5$. The initial values of ϕ and η at each cell were assigned randomly around their spatial averages and the range of initial random values was taken as $\eta = \phi = \pm 0.5$. The other parameters [59, 111] for simulations given in Figure 6.1 were chosen as $A_1 = 1.3$, $A_2 = 1.1$, $\bar{\phi} = \bar{\eta} = 0$, $\alpha = 0.02$ $c_1 = c_2 = 0.5$, $b_1 = 0.07$, and $b_2 = 0.2$ for the domain growth in the lamellar forming system [59]. The source code for these simulations is given in Appendix G.

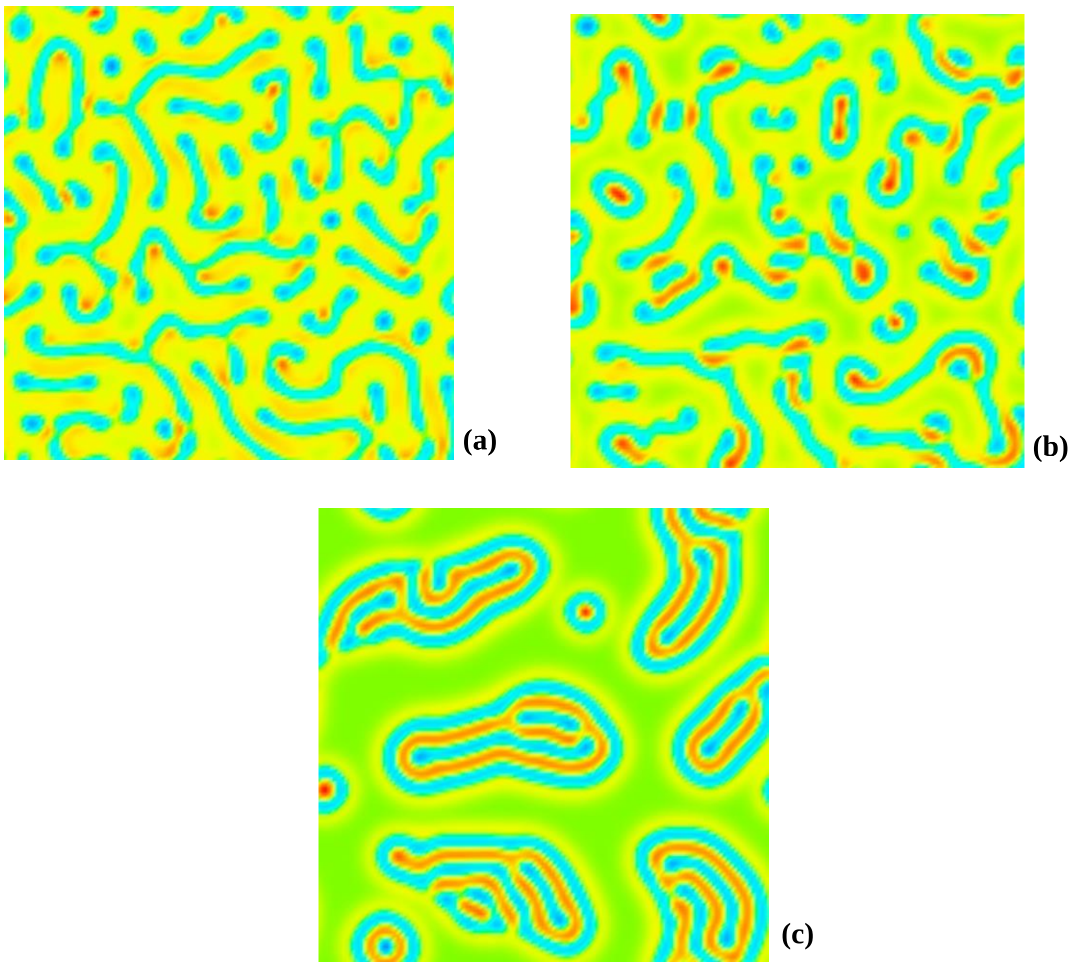


Figure 6.1: The diblock copolymer and homopolymer (A–B/C) systems. Images (a), (b) and (c) are 1000th, 4000th and 50000th time steps respectively.

The phase separation can be noticed, which starts in Figure 6.1 (a) where the macrophase separation between the copolymer–rich (small red particles covered by blue) and homopolymer (yellow) rich phases takes place. The evolution of order parameter in the macrophase separation is carried out by independent variable ψ and the evolution of order parameter in microphase separation is carried out by ϕ. As the evolution of the two order parameter system goes on, the microphase separation actually starts after some time–evolution, which can be noticed in Figure 6.1 (b) at the 4000th time step. In Figure 6.1 (c), the homopolymer is now in the rich domain, as mostly the green domain can be noticed.

The simulation results are also presented for the hexagonal microdomains. This is a different morphology in the *A–B/C* systems. When the value of *f* is chosen different from in 1/2, the lamellar structure does not remain stable and such microdomains come into formation [59]. The objective of including these results is basically to extend the understanding of two order parameter systems and how the model equations can be used to investigate other morphologies. The parameters are different from the lamellar microdomain simulations. The double phase separation corresponds to $f = 0.4$. Other parameters that were used are given as $\bar{\phi} = -0.2$, $\bar{\eta} = 0$, $b_1 = 0.0$, $b_3 = 0.01$, $\alpha = 0.02$, $\beta = 0.02$, and $\gamma = 0.002$ [59]. The simulation results of the hexagonal microdomains' morphology are given in Figure 6.2. The source code used for simulations shown in Figure 6.2 was the same as is given in Appendix G for the simulations in Figure 6.1.

In hexagonal microdomains simulations, the value of *f* is taken differently from that of lamellar morphology, and it can be observed that the term with coefficient b_3 in equation (6.5) comes into play. The idea behind the simulations is to manipulate model equations with different values and discuss the obtained results. So far, the discussion is undertaken for the phase separation in the *A*–*B* diblock copolymer and the *C* homompolymer (*A*–*B*/*C*) systems. Here, the discussion is undertaken in more detail to understand the relation between the kinetics and morphology and the pattern formation within the *A*–*B*/*C* systems [111].

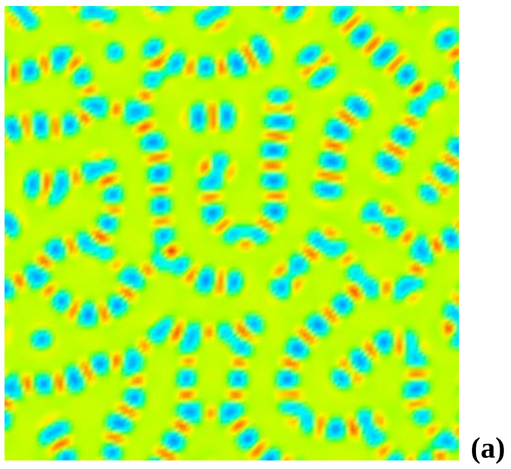
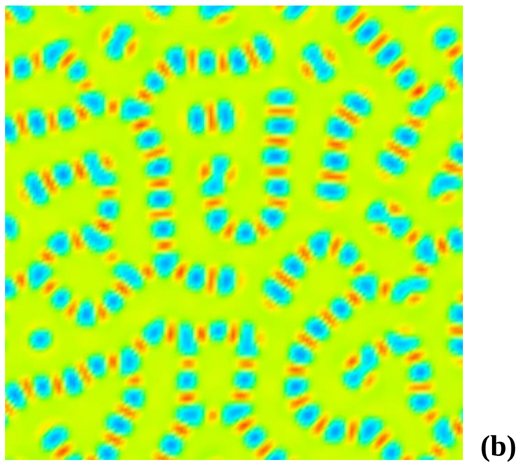



Figure 6.2: Hexagonal microdomains in A– B/C systems. Images (a) and (b) are 10000th and 50000th time steps respectively.

In the next simulations the domain pattern is slightly changed in equation (6.5); the value of b_2 is nonzero and the term of higher order with coefficient b_4 is avoided by setting its value as zero. The term with coefficient b_3 vanishes itself as the parameter *f* =1/2 is chosen. The initial random conditions for the disordered state of η and ϕ were chosen at t = 0.0 where $\bar{\eta} - s < \eta < \bar{\eta} + s$ and $\bar{\phi} - s < \phi < \bar{\phi} + s$ with s = 0.01 [111]. The parameters used in equations (6.11)–(6.14) were chosen, as $\bar{\eta} = -0.2$, $\bar{\phi} = 0.0$, $c_1 = 1.0$, $c_2 = 0.5$, $A_1 = 1.3$, $A_2 = 1.1$, and $\alpha = 0.02$. The simulation results based on these parameters are shown in Figure 6.3.

The phase separation is shown for the *A*–*B* diblock copolymer and *C* homopolymer in Figure 6.3. In Figure 6.3 (a) the phase separation starts and in the first instance the macrophase phase separation takes place between the *A*–*B* diblock copolymer and the *C* homopolymer and then the microphase separation can been seen in the *A*–*B* diblock copolymers. In Figure 6.3 (b) and (d) the macrophase separation can be seen. It can be noted that in Figure 6.3 (d) the diblock copolymers emerge as onion rings. This is not because of any particular use of the Laplacian operator, but due to the random initial conditions described above. The time step $\Delta t = 0.5$ was chosen to keep the model isotropic [111].

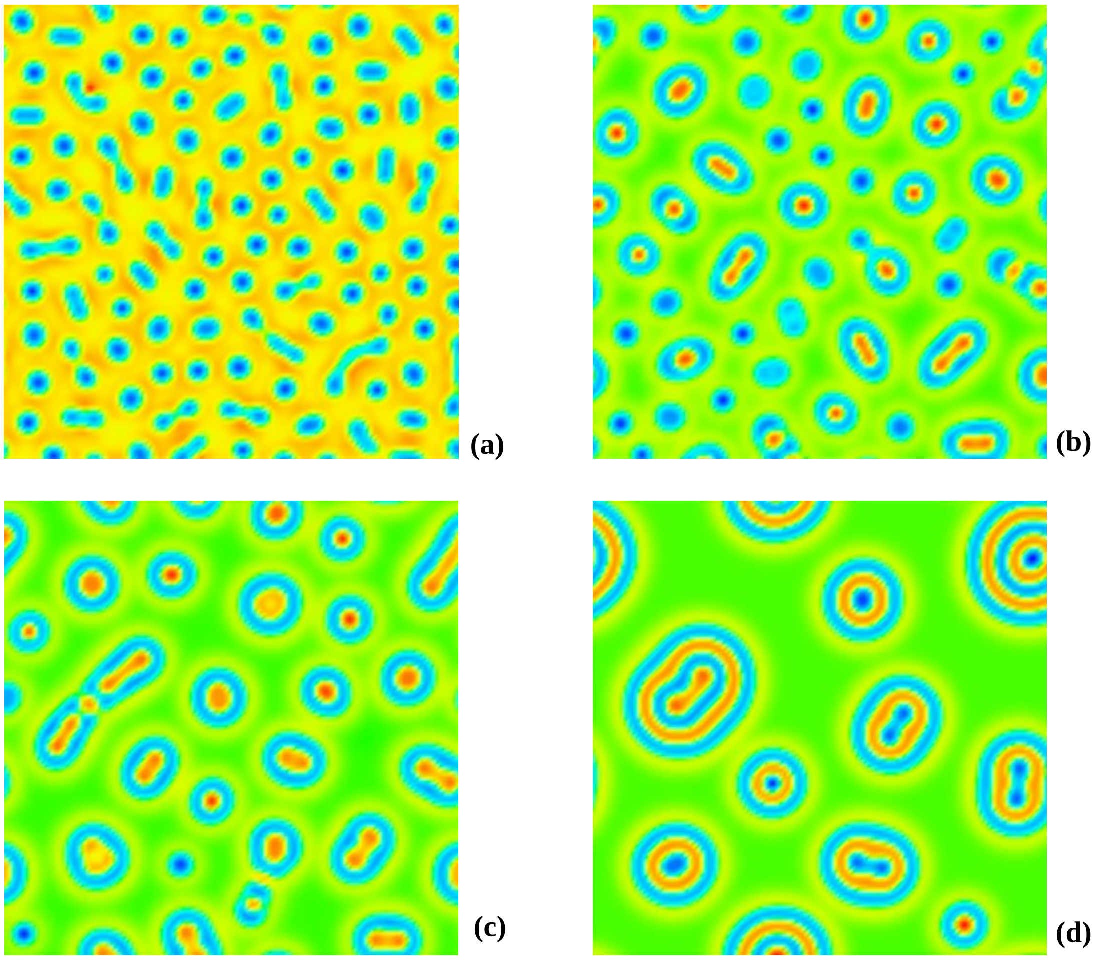


Figure 6.3: The simulation images (a), (b), (c) and (d) are 1000th, 4000th, 10000th and 70000th time steps respectively.

6.3 Three-dimensional simulations of *A–B/C* systems

The 3D simulations have been incorporated for the two order parameter systems to understand the essence of phase–separation phenomena of *A–B/C* systems where a lamellar structure of diblock copolymer is expected to appear in the macrophase–separated phase. The 3D simulations were based on discrete equations (6.11)–(6.14) and computer programs were developed and executed by following the six steps of an algorithm given in section 6.2.1. The simulation results shown in Figure 6.4 and Figure 6.5 were based on the following specifications: The grid size chosen was $75\times75\times50$ with grid spacing $\Delta x=\Delta y=1$, the total time of the simulations was up to 50000 time steps with time interval $\Delta t=0.5$. The initial values of ϕ and η at each cell were assigned randomly around their spatial averages and the range of initial random values was taken as $\eta=\phi=\pm0.3$. The other parameters [59, 111] for simulations given in Figure 6.4 and Figure 6.5 were chosen as $A_1=1.3$, $A_2=1.1$, $\bar{\phi}=\bar{\eta}=0$, $\alpha=0.02$ $c_1=c_2=0.5$, $b_1=0.07$, and $b_2=0.2$ for the domain growth in the lamellar forming system [59]. It must be noted the that the snapshots of 3D simulations are presented without any specific time scale.

The simulations shown in Figure 6.4 were obtained using the 3D 27–point Laplacian operator of Shinozaki and Oono’s choice (Laplacian scheme SO(D3Q27)) in forward Euler method. In Figure 6.4, the images (a), (b) and (c) are shown for different stages of domain growth in a phase separation based on above parameters in *A–B/C* systems at different time steps. The initial stage of evolution can be observed in Figure 6.4 (a) where the macrophase separation takes place between *A–B* diblock copolymers and *C* homopolymer. In Figure 6.4 (b), the simulation image at 10000th time step displays the phase separation in a complete domain growth where *C* homopolymer rich domain for $\eta<0$ is indicated by blue colour and a copolymer rich domain for $\eta>0$ is drawn by the green colour. It should be noted when the microphase separation starts then the copolymer rich domains become sufficiently large. It can be observed from Figure 6.4 (b) and (c) that in most part of the macro–domains (blue colour), the lamellar domains (green colour) are surrounded by a thin layer. These thin layers are the A blocks. The image in Figure 6.4 (c) shows the domain growth of the system at 50000th time step where macro–domains become larger and at this stage the system becomes stable.

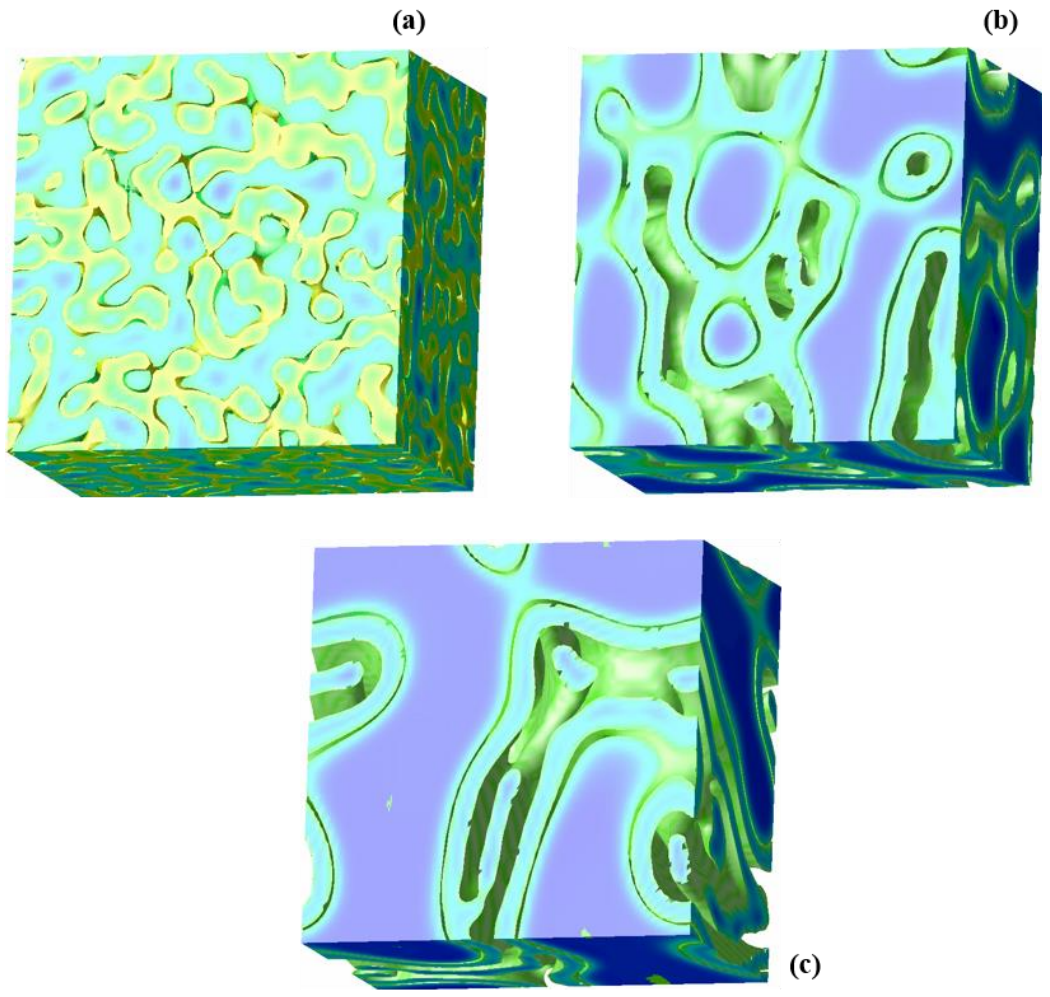


Figure 6.4: The simulation images (a), (b), and (c) are 1000th, 10000th, and 50000th time steps respectively using Laplacian scheme SO(D3Q27).

For 3D simulations given in Figure 6.4, the default CDS averaging operator SO(D3Q27) was employed to obtain isotropic simulation results. The 3D simulations were also executed using 19–point D3Q19 and 27–point BV(D3Q27) Laplacian schemes based on the same parameters used for simulations given in Figure 6.4. The simulation result shown in Figure 6.5 was obtained using Laplacian scheme D3Q19 and the simulation result shown in Figure 6.5 (b) was obtained using BV(D3Q27). In Figure 6.5, images (a) and (b) show the phase separation in *A–B/C* systems in a similar way as shown in Figure 6.4 (c). These 3D Laplacian schemes were discussed in chapter four for isotropic simulation results of spherical morphology and for those simulations these three schemes performed better.

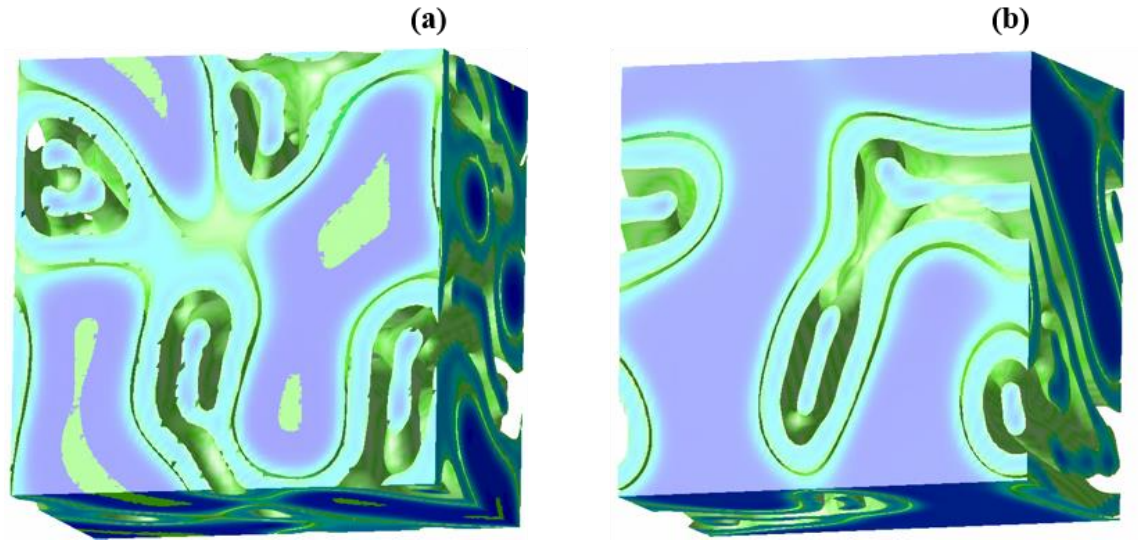


Figure 6.5: The simulation images (a) and (b) were obtained at 50000th time step using Laplacian schemes D3Q19 and BV(D3Q27) respectively.

The simulations results obtained from three different isotropic Laplacian schemes yield almost same results which can be observed by comparing Figure 6.4 (c), Figure 6.5 (a) and Figure 6.5 (b). No any divergence of values was observed during the execution by using these 3D Laplacian schemes. It can be observed that the (b) image of Figure 6.5 is clearer and better in shape compared to Figure 6.4 (c) and Figure 6.5 (a). It must be noted that the execution of simulations based on 19–point Laplacian scheme D3Q19 was observed faster due to the few stencil points. It also allows a larger room for using time interval value. The simulations based on BV(D3Q27) seem more isotropic and this Laplacian scheme performs better.

6.4 Implementation of Crank–Nicolson scheme in *A–B*/*C* systems

The Crank–Nicolson (CN) scheme was explained in Chapter five, where this scheme was implemented for the one order parameter system of the lamellar forming of diblock copolymer systems. In this chapter, the CN scheme was implemented for the two order parameter (*A*–*B*/*C*) systems; that is, for the lamellar forming of the *A*–*B* diblock copolymer and the *C* homopolymer. Before carrying out the implementation of the CN scheme for two–order parameter systems, the implementation of the backward Euler method was undertaken for the two order parameter systems. The backward Euler method is unconditionally stable but first–order accurate [69].

The mathematical model for the two order parameter systems is presented in section 6.1, where this model is discussed in detail in terms of short–range and long–range free energies. The finite differencing for two order parameter systems is implemented, based on the numerical equations given in section 6.2.

Two order parameter with incompressibility conditions are:

$$\begin{aligned} \phi &= \phi_A - \phi_B \\ \psi &= \phi_A + \phi_B \end{aligned} \tag{6.16}$$

where ϕ is for an order parameter in the microphase separation and ψ is the segregation of copolymer/homopolymer. Ohta introduced a new variable $\eta = \psi - \psi_c$, where ψ_c is the volume fraction at the critical point of the macrophase separation [59]. The numerical equations (6.11) and (6.12) are written in the simple form of partial differential equations, which are mainly involved in the two order parameter system and are given as follows:

$$\frac{d\eta}{dt} = -\nabla^2(c_1\nabla^2\eta + A_1 \tanh\eta - \eta - b_1\phi + \frac{1}{2}b_2\phi^2 + b_3\eta\phi - b_4\eta\phi^2). \tag{6.17}$$

$$\frac{d\phi}{dt} = -\nabla^2(c_2\nabla^2\phi + A_2 \tanh\phi - \phi - b_1\eta + b_2\eta\phi + \frac{1}{2}b_3\eta^2 - b_4\eta^2\phi). \tag{6.18}$$

The terms with coefficients b_3 and b_4 are omitted in the above set of equations (6.17) and (6.18) and, for the purpose of finite differencing, these equations in the form of *n* and *n*+1 are given as follows:

$$\eta_{j,k}^{n+1} = \eta_{j,k}^{n} - \nabla^2(c_1\nabla^2\eta_{j,k}^{n} + A_\eta \tanh\eta^n - \eta^n - b_1\phi^n + \frac{1}{2}b_2(\phi^n)^2). \tag{6.19}$$

$$\phi_{j,k}^{n+1} = \phi_{j,k}^{n} - \nabla^2(c_1\nabla^2\phi_{j,k}^{n} + A_\eta \tanh\phi^n - \phi^n - b_1\eta^n + b_2\eta^n\phi^n). \tag{6.20}$$

Equations (6.19) and (6.20) are non–homogenous partial differential equations and will be implemented in the CN methodology. The homogenous parts of the equations will be evaluated first and then the non–homogenous parts will be evaluated later. In both equations (6.19) and (6.20), Laplacian operators are used. The choice of the Laplacian operators is taken as that of the 9–point Laplacian operator given below:

$$\nabla^2 u = \frac{1}{6}\sum_{NN} u + \frac{1}{12}\sum_{NNN} u - u \tag{6.21}$$

The above 9–point Laplacian scheme was discussed in Chapter three for the one order parameter system of lamellar forming of diblock copolymers using the cell dynamics simulation technique. This is the choice used by Oono and Puri [47] and is recognized as the isotropic Laplacian operator. The simulation results given in this chapter were based on this Laplacian operator. The two–order parameter equations (6.19) and (6.20) can be written in the backward Euler method, given as follows:

$$\eta_{j,k}^{n+1} + \nabla^2 (c_1 \nabla^2 \eta_{j,k}^{n+1} + A_1 \tanh \eta^{n+1} - \eta^{n+1} - b_1 \phi^{n+1} + \frac{1}{2} b_2 (\phi^{n+1})^2) = \eta_{j,k}^{n}. \qquad (6.22)$$

$$\phi_{j,k}^{n+1} + \nabla^2 (c_2 \nabla^2 \phi_{j,k}^{n+1} + A_2 \tanh \phi^{n+1} - \phi^{n+1} - b_1 \eta^{n+1} + b_2 (\eta^{n+1} \phi^{n+1})^2) = \phi_{j,k}^{n}. \qquad (6.23)$$

The above equations are derived from equations (6.19) and (6.20) where the terms are shown on the left hand side. To describe equations (6.22) and (6.23) more clearly, these equations are rewritten as follows:

$$\underbrace{\eta_{j,k}^{n+1} + \nabla^2 (c_1 \nabla^2 \eta_{j,k}^{n+1})}_{part1} + \underbrace{\nabla^2 (A_1\, tanh\, \eta_{j,k}^{n+1} - \eta_{j,k}^{n+1} - b_1 \phi_{j,k}^{n+1} + \frac{1}{2} b_2 (\phi_{j,k}^{n+1})^2)}_{part2} = \eta_{j,k}^{n}. \qquad (6.24)$$

$$\underbrace{\phi_{j,k}^{n+1} + \nabla^2 (c_2 \nabla^2 \phi_{j,k}^{n+1})}_{part1} + \underbrace{\nabla^2 (A_2\, tanh\, \phi_{j,k}^{n+1} - \phi_{j,k}^{n+1} - b_1 n_{j,k}^{n+1} + b_2 (n_{j,k}^{n+1} \phi_{j,k}^{n+1})^2)}_{part2} = \phi_{j,k}^{n}. \qquad (6.25)$$

In the above set of equations, part 1 is comprised of homogenous terms and part 2 is comprised of non–homogenous terms. The homogenous parts of equations (6.24) and (6.25) are evaluated in the first step and then the non–homogenous parts are evaluated in the second step. In order to solve the linear system of equations (6.24) and (6.25), the iterative method needed to be used. Equations (6.24) and (6.25) of the backward Euler method are given in two dimensions which contain the Laplacian operator ∇^2 and the biharmonic (bilaplacian) operator ∇^4. The 9–point isotropic Laplacian operator ∇^2 given in equation (6.21) is simulated in the matrix *M* and the biharmonic operator is assumed to be $P = M \times M = \nabla^4$. Both the *M* and *P* matrices are constructed considering the periodic boundary conditions as per the requirements of the two order parameter computer simulations [111]. Basically, the *M* matrix is comprised of submatrices.

In Chapter five, a complete description is given regarding the formation of these matrices for the 5–point Laplacian operator using periodic boundary conditions. The Conjugate Gradient (CG) method was used to solve the system of $x = P^{-1}b$ in equations (6.24) and (6.25) where *x* is unknown and *b* is known. The Conjugate Gradient (CG) method is discussed in detail in Appendix E. This iterative method is efficient and stable compared to the LU decomposition method, which is found to be slower for the higher grids.

The backward Euler method for CDS based on two order parameter systems was implemented for the simulation of lamellar forming of the *A*–*B* diblock copolymer and the *C* homopolymer systems. The grid size chosen was 128×128 with grid spacing $\Delta x = \Delta y = 1,$ and the total time of the simulations was up to 10000 time steps. The initial random conditions for disordered state of η and ϕ at *t* = 0.0 were $\bar{\eta} - s < \eta < \bar{\eta} + s$ and $\bar{\phi} - s < \phi < \bar{\phi} + s$ with *s* = 0.01 [111]. The parameters used in equations (6.24) and (6.24) were chosen as $\bar{\eta} = -0.2,\ \bar{\phi} = 0.0,\ c_1 = 1.0,\ c_2 = 0.5,\ A_1 = 1.3,\ A_2 = 1.1,$ and $\alpha = 0.02$. The simulation results obtained from the backward Euler method using CDS for *A*–*B*/*C* systems are shown in Figure 6.6 and these simulation results represent the phase separation based on two order parameter systems and the dynamics of these systems is the same as that described for the simulation results shown in Figure 6.3.

In order to compare the forward Euler and backward Euler methods, the distribution of two order parameters in *A*–*B*/*C* systems is presented in Figure 6.7 for order parameter ϕ and in Figure 6.8 for order parameter ψ against the space 0–128. Figure 6.7 and Figure 6.8 are constructed in a similar way as in Figure 5.7, where the distribution of numerical values of order parameter ψ obtained from 2D simulations (Figure 6.3 and Figure 6.6) for forward Euler and backward Euler are shown for comparison.

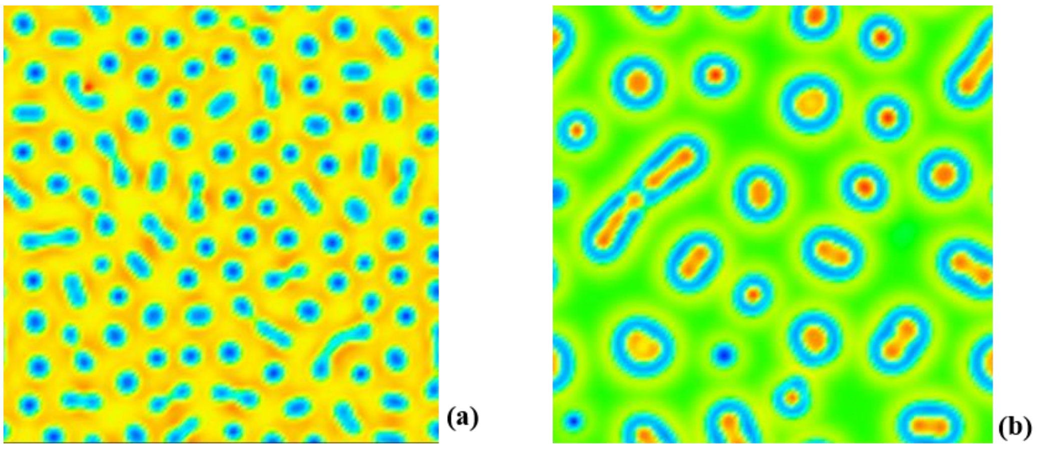


Figure 6.6: The images (a) and (b) are 1000th and 10000th time steps respectively obtained using the backward Euler method for two–order parameter systems.

In Figure 6.7 (a) and Figure 6.8 (a) the values are compared at the 1000th time step and in Figure 6.7 (b) and Figure 6.8 (b) the values are compared at the 10000th time step. The comparison graphs show the simultaneous two order parameters' evolution using the two different methods. In all four images of Figures 6.7 and 6.8, two lines of different colours show two different methods; each line represents the numerical values for an order parameter which was assigned by the initial random values for the simulations. In the simulations of both methods, the 9–point isotropic Laplacian scheme OP(D2Q9) was employed. The lines are exactly parallel, which shows that the tendency of the numerical values obtained from the two different methods is the same, while the methodologies are different. The parallel distribution of numerical values in Figure 6.7 and Figure 6.8 shows that there is no obvious difference for the evolution of order parameters and therefore the phase separations simulated by using the two different methods are almost similar in pattern, which can be observed by comparing the simulation results shown in Figure 6.6 (b) for the backward Euler method and the simulation results shown in the (a) and (c) snapshots of Figure 6.3 for the forward Euler method.

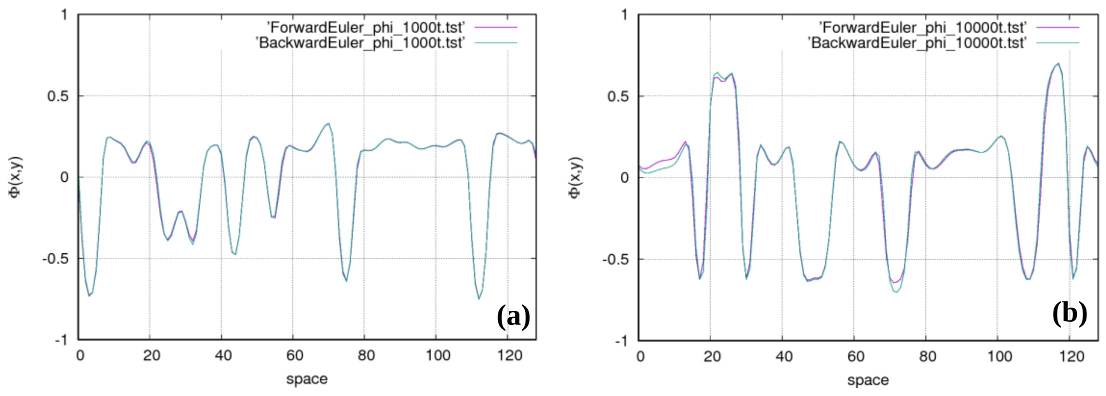


Figure 6.7: The numerical values of order parameter $\phi(x,y)$ are plotted against the space (0 – 128) for 100th and 10000th time steps in images (a) and (b) respectively. Numerical values of $\phi(x,y)$ are shown for the forward Euler and backward Euler methods. The numerical values plotted here were obtained from the simulations shown in Figure 6.3 and 6.6.

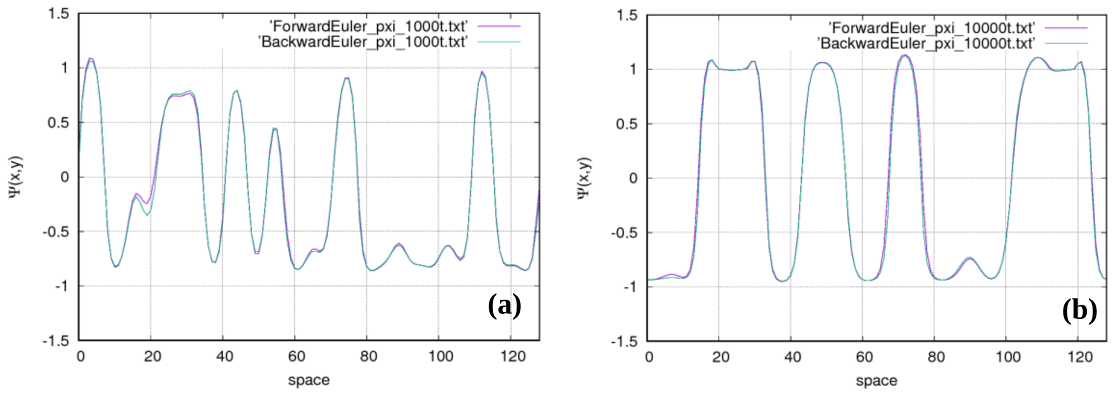


Figure 6.8: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 100th and 10000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler and backward Euler methods. The numerical values plotted here were obtained from the simulations shown in Figure 6.3 and 6.6.

The main objective is to implement the CN scheme for the two order parameter systems and to acquire accurate results, as with those of the forward Euler method. Here, the equations of two order parameter systems are implemented in the CN scheme. The CDS equations of two order parameter systems are given below:

$$\begin{aligned}&\underbrace{\eta^{n+1}_{j,k} + \nabla^2(c_1\nabla^2\eta^{n+1}_{j,k})}_{part1} + \underbrace{\nabla^2(A_1\tanh\eta^{n+1}_{j,k} - \eta^{n+1}_{j,k} - b_1\phi^{n+1}_{j,k} + \frac{1}{2}b_2(\phi^{n+1}_{j,k})^2)}_{part2} \\ &= \underbrace{\eta^{n}_{j,k} + \nabla^2(c_1\nabla^2\eta^{n}_{j,k})}_{part1} + \underbrace{\nabla^2(A_1\tanh\eta^{n}_{j,k} - \eta^{n}_{j,k} - b_1\phi^{n}_{j,k} + \frac{1}{2}b_2(\phi^{n}_{j,k})^2)}_{part2}\end{aligned} \tag{6.26}$$

$$\begin{aligned}&\underbrace{\phi^{n+1}_{j,k} + c_2P\phi^{n+1}_{j,k}}_{part1} + \underbrace{M(A_2\tanh\phi^{n+1}_{j,k} - \phi^{n+1}_{j,k} - b_1\phi^{n+1}_{j,k} + b_2(n^{n+1}_{j,k}\phi^{n+1}_{j,k})^2)}_{part2} \\ &= \underbrace{\phi^{n}_{j,k} + c_2P\phi^{n}_{j,k}}_{part1} + \underbrace{M(A_2\tanh\phi^{n}_{j,k} - \phi^{n}_{j,k} - b_1\phi^{n}_{j,k} + b_2(n^{n}_{j,k}\phi^{n}_{j,k})^2)}_{part2}\end{aligned} \tag{6.27}$$

Equations (6.26) and (6.27) are implemented in the CN scheme, where the terms on the right hand side are evaluated with the forward Euler method in *n* space. When the right hand sides are approximated completely, then the new values are evaluated for *n*+1 at the left hand side from the values approximated at the right hand side by using the backward Euler technique. On the left hand side, for the approximation of the values, the CG method was used. The CN scheme worked successfully and the results using this scheme are presented in the following Figure (6.9), with the graphs for comparison in Figure 6.10 and Figure 6.11.

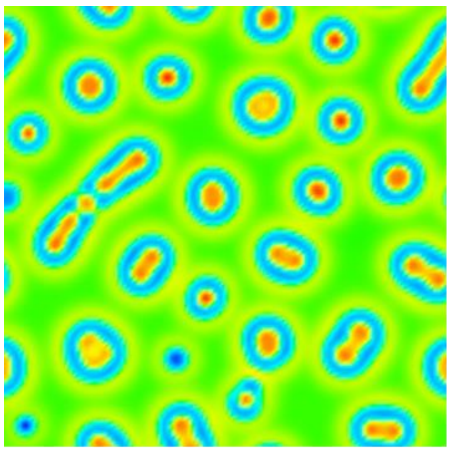
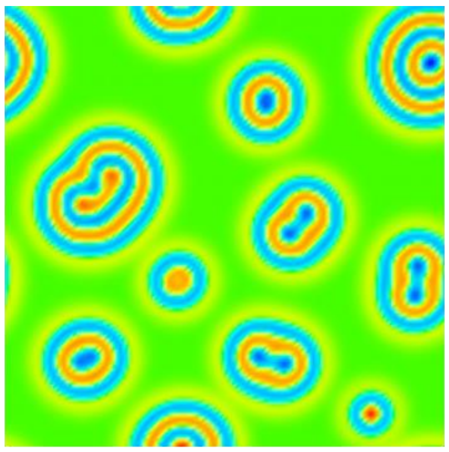

(a)

(b)

Figure 6.9: The images (a) and (b) are 10000th and 70000th time steps respectively obtained using the CN method for two–order parameter systems.

The parameters and the specifications used were the same as those used in the simulations shown in Figure 6.6 for the backward Euler method. The total time of simulation of was up to 70000 time steps.

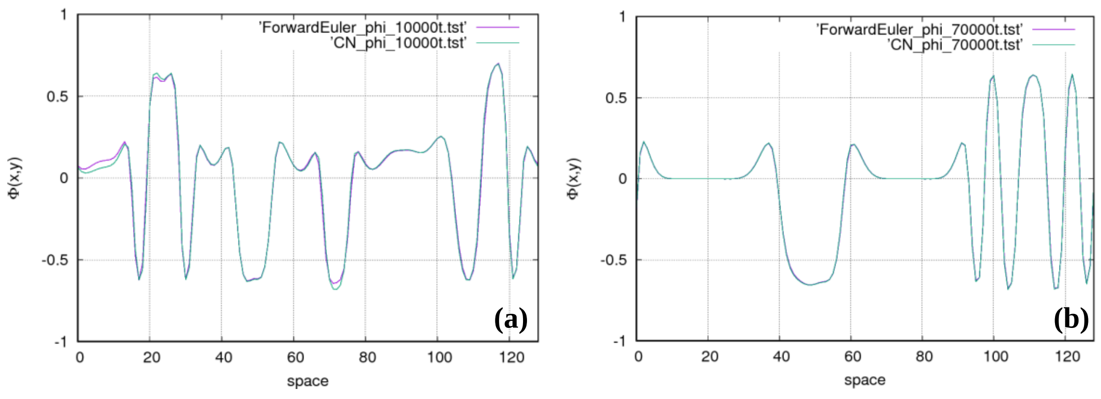


Figure 6.10: The numerical values of order parameter $\phi(x,y)$ are plotted against the space (0 – 128) for 100th and 70000th time steps in images (a) and (b) respectively. Numerical values of $\phi(x,y)$ are shown for the forward Euler and CN methods. The numerical values plotted here were obtained from the simulations shown in Figure 6.3 and 6.9.

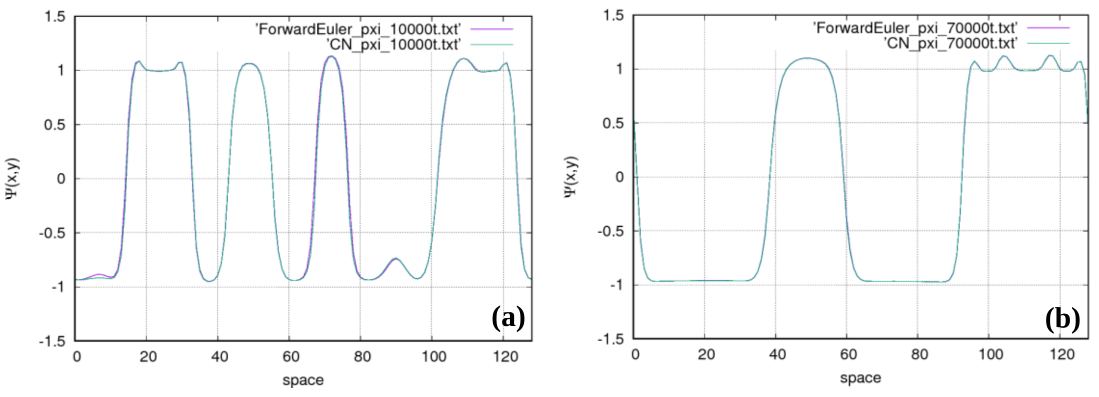


Figure 6.11: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 100th and 70000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler and CN methods. The numerical values plotted here were obtained from the simulations shown in Figure 6.3 and 6.9.

The simulation results presented in Figure 6.9 were obtained using the CN scheme and it can be observed that the order parameters' evolution behaved the same as that of the forward and the backward Euler methods. The macrophase separation between the *A*–*B* diblock copolymer and the *C* homopolymer takes place first, then the microphase separation in the diblock copolymer takes place. The time evolution process for time steps can be observed in Figures 6.9 (a) and 6.9 (b), especially in Figure 6.9 (b) where the diblock copolymers become rich in their domains. The two order parameter systems are comprised of two processes carried out by two independent variables where ϕ represents the microphase separation and ψ represents the macrophase separation.

In order to compare the forward Euler and CN methods, the distribution of two order parameters in *A*–*B*/*C* systems is presented in Figure 6.10 for order parameter $\phi(x,y)$ and in Figure 6.11 for order parameter $\psi(x,y)$ against the space 0–128. Figure 6.10 and Figure 6.11 are constructed in a similar way as in Figure 6.7 and Figure 6.8. In Figure 6.10 (a) and Figure 6.11 (a) the values are compared at the 10000th time step and in Figure 6.10 (b) and Figure 6.11 (b) the values are compared at the 70000th time step. The plots in Figures 6.10 and 6.11 are representing the numerical values obtained for the 2D simulations shown in Figures 6.3 and 6.9. The comparison graphs show the simultaneous two order parameters' evolution using the two different methods. In all four images of Figures 6.10 and 6.11, two lines of different colours show the two different methods and each line represents the numerical values for an order parameter which was assigned by the initial random values for the simulations. The lines are exactly parallel, which shows that the tendency of the numerical values obtained from the two different methods is the same, while the methodologies are different. The parallel distribution of the numerical values in Figure 6.10 and Figure 6.11 shows that there is no obvious difference for the evolution of order parameters and therefore the phase separations simulated by using the two different methods are almost similar in pattern, which can be observed by comparing the simulation results shown in Figure 6.9 for the CN method and the simulation results shown in the (c) and (d) snapshots of Figure 6.3 for the forward Euler method.

The implementation of the Crank–Nicolson method for a two order–parameter system allows the use of any time interval value for simulations; this flexibility of choosing a time interval value helps to relieve the anisotropy of the domain in the late stage of domain growth, which may arise from the discretization of the space. Aya Ito [111] presented the simulations for domain patterns in copolymer and homopolymer mixtures where he chose $\Delta t = 0.5$ to avoid numerical instability and if any other Δt value is used the system becomes unstable. By the implementation of implicit schemes, especially the CN method, it was possible to obtain these simulation results for choosing $\Delta t = 1$.

6.5 Conclusions

In this chapter, the simulation results are presented for a mixture of *A*–*B* diblock copolymers and *C* homopolymers. After changing some parameters in the same set of equations for a two order parameter system, the simulation results were obtained for the hexagonal domains and onion-like lamellar forming, which were investigated to understand the two order parameter system in its various other structure formations. The cell dynamics simulation technique for two order parameter systems was implemented in two other finite difference schemes, the backward Euler and the Crank–Nicolson schemes. As already discussed, the backward Euler scheme is unconditionally stable and so is the Crank–Nicolson. The Matrix based approach was used for the Laplacian operator, where the periodic boundary conditions were set for the 9–point isotropic Laplacian operator. The five–point formula for the Laplacian operator could also be used, as could other isotropic Laplacian operators. In the literature, the only choice taken is that of the 9–point isotropic Laplacian operator of Oono and Puri. The CG method was employed for solving linear systems of equations. Using the implicit schemes, huge and large sparse matrices were produced and for this reason it was only possible to use the grid size of 128×128 for the simulations in these schemes.

It was not possible to carry out the three-dimensional simulations in both the backward Euler and CN schemes. Three-dimensional results for cell dynamics in two order systems can be obtained using the ADI scheme.

Chapter Seven

7 Implementation of the Alternating Direction Implicit method for CDS equations

In this chapter, the Alternating Direction Implicit (ADI) method is implemented for the Cell Dynamics Simulation (CDS) method for the lamellar forming of *A*–*B* diblock copolymers using the one–order parameter system. The implementation of the ADI scheme for CDS is one of the objectives of this study. Two different Finite Difference (FD) methods based on ADI are discussed and implemented for the CDS equations for modelling the lamellar forming system of *A*–*B* diblock copolymers. The computer codes of ADI methods for CDS were developed by following the same algorithm of eight steps which was given at the start of Chapter four. Firstly, the generalized ADI method is implemented, based on the 5–point Laplacian operator. Secondly, Hundsdorfer's ADI method is implemented, based on the 5–point Laplacian. The results obtained from the ADI methods are compared with those of the forward Euler method.

7.1 Implementation of the ADI method for CDS

In this section, the implementation of the ADI method is presented for the Cell Dynamics Simulation (CDS) technique. The model of CDS is comprised of partial differential equations which essentially involve the biharmonic operator. In Chapter five, the biharmonic operator was discussed for the implementation of the CN method for CDS.

The steps needed to solve the Time–Dependent Ginzburg–Landau (TDGL) equation based on the ADI method are shown here. First, equation (3.10) can be re–written in a simple form of PDE as [2]:

$$\frac{\partial \psi}{\partial t} = -\{\nabla^2 (g(\psi) + D\nabla^2 \psi) + B\psi\}\,. \tag{7.1}$$

with ψ the spatial order parameter, *t* the time, ∇^2 the Laplacian on a function of free energy functional, and *D* as a diffusive parameter. Equation (7.1) is non–linear and fourth–order, including the *bi*–Laplacian or biharmonic operator ∇^4. After doing some algebraic manipulation without changing the meaning of the equation (7.1), it can be written as:

$$\frac{\partial \psi}{\partial t} = -\nabla^2 g(\psi) - \nabla^2 (D\nabla^2 \psi) - B\psi\,, \tag{7.2}$$

where $g(\psi)$ is the so–called map function given by [2]:

$$g(\psi) = [1 + \tau - A(1-2f)^2]\psi - v(1-2f)\psi^2 - u\psi. \tag{7.3}$$

Equation (7.2) is a non–homogenous partial differential equation. The first part contains the homogenous terms and the second part contains the non–homogenous term as given below:

$$\psi_{j,k}^{n+1} = \underbrace{\psi_{j,k}^{n} - \Delta t D\nabla^2(\nabla^2)\psi_{j,k}^{n} - \Delta t B\psi_{j,k}^{n}}_{part1} - \underbrace{\Delta t\nabla^2 g(\psi_{j,k}^{n})}_{part2}. \tag{7.4}$$

The Laplacian operator ∇^2 can be seen in the non–homogenous part of the equation. The five–point stencil formula [98] is employed for the Laplacian operator in numerical simulations of the map function. Equations (5.5) to (5.7) can be seen for the five–point formula. In equation (7.4), there is a biharmonic operator $\nabla^2(\nabla^2) = \nabla^4$ in the second term on the right hand side which can be discretized using the thirteen–point stencil [98]. The two-dimensional thirteen–point stencil formula for the biharmonic operator [107, 108] is given in equation (5.8) in section 5.1.1 of Chapter five.

To use splitting operators, the operators $\partial^4/\partial x^4(\psi_{j,k})$ and $\partial^4/\partial y^4(\psi_{j,k})$ in thirteen-point formula are denoted by $\Delta_4^{(x)}$ and $\Delta_4^{(y)}$ respectively. Equation (5.10) in section 5.1.1 can be seen for the full derivative form of $\partial^4/\partial x^4(\psi_{j,k})$ operator. The CDS equation involves the biharmonic operator and many other such types of equations which involve the biharmonic operator are used in various applied mathematical models [113-116]. The biharmonic operator uses the mixed derivatives and therefore the ADI method using the Douglas and Gunn [82] scheme for approximating mixed derivate terms is not suitable to implement for the approximation of biharmonic operator including the boundary value problems [107]. The ADI methods for equations involving mixed derivate terms are present in the literature [79, 117, 118]. Conte and Dames published their work for the implementation of the ADI scheme for problems involving the biharmonic operator [84, 119, 120].

7.1.1 Generalized ADI method

Witelski and Bowen derived a generalized ADI operator–split form (see Ref. [107], equation (2.10)) from D'Yakanov's [121, 122] form and this ADI method for equation (7.4) is given below:

$$\begin{aligned}
&L_x w = \psi^n_{j,k} - \Delta t D \nabla^4 \psi^n_{j,k} - \Delta t B \psi^n_{j,k} - \Delta t \nabla^2 f(\psi^n_{j,k}),\\
&L_y v = w,\\
&\psi^{n+1}_{j,k} = \psi^n_{j,k} + v.
\end{aligned} \tag{7.5}$$

where:

$$L_x = \left(I + \frac{\Delta t I B}{2} + \frac{\Delta t D}{2}\Delta_4^{(x)}\right) \tag{7.6}$$

$$L_y = \left(I + \frac{\Delta t I B}{2} + \frac{\Delta t D}{2}\Delta_4^{(y)}\right) \tag{7.7}$$

The L_x and L_y are matrices where $L_x = L_y = M$. The *M* is a penta–diagonal matrix used to calculate the values for vectors *w* and *v* in equation (7.5). The periodic boundary conditions are implemented in matrix *M* and its general is form is given below:

$$M = \begin{pmatrix} b_1 & c_1 & d_1 & 0 & 0 & f_1 \\ a_1 & b_2 & c_2 & d_2 & 0 & 0 \\ e_1 & a_2 & b_3 & c_3 & d_3 & 0 \\ 0 & e_2 & a_3 & b_4 & c_4 & d_4 \\ 0 & 0 & e_3 & a_4 & b_5 & c_5 \\ f_2 & 0 & 0 & e_4 & a_5 & b_6 \end{pmatrix} \tag{7.8}$$

Equation (7.5) can be seen in three steps and in the first step the explicit part is on the right hand side. In the first step of equation (7.5), the one–dimensional vector of size *N* is calculated explicitly if the grid size is $N \times N$. In the first part of the equation (7.5), the non–homogenous part of equation (7.4) is combined with the full operator. On the left hand side, the values are approximated for vector *w* implicitly in x–direction by using $L_x = M$ matrix. In the second step of equation (7.5), the values for *v* on the left hand side are approximated in y–direction implicitly from vector *w*. The last step calculates the $n+1$ values for order parameter ψ. The LU decomposition method is used to separate *M* in *L* and *U* matrices for the convenience of employing the Thomas algorithm. This technique only uses vectors other than the whole matrix and thus it simplifies the solution. The results shown in Figure 7.1 and Figure 7.2 are the images at 1000th and 100000th time steps.

For simulation results given in Figures 7.1 and 7.2, the system parameters used are given in Table 7.1 and the other specifications used are given as follows:

- The grid size was chosen 128×128 with grid spacing $\Delta x = \Delta y = 1$;
- The total time of the simulations was up to 100000 time steps;
- The simulations were run with periodic boundary conditions;

- The simulations were started from an initial random disordered state $\psi = \pm 0.3$.

Table 7.1: System parameters used in cell dynamical method for lamellae morphology

CDS Parameters	τ	f	u	V	B	D	A
Lamellae Morphology	0.36	0.48	0.38	2.3	0.02	0.7	1.5

The parameters given in Table 7.1 are suggested for lamellar forming diblock copolymer systems [2] and all the simulations in this chapter are based on these. The results in Figure 7.1 are based on the explicit forward Euler method and in Figure 7.2 the results presented are obtained from the general ADI method.

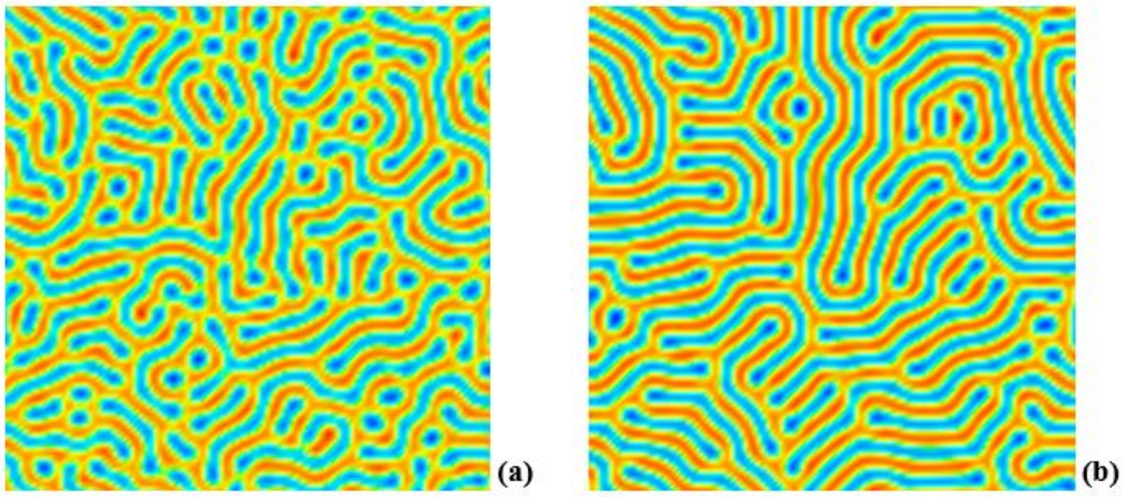


Figure 7.1: Explicit forward Euler method based on 5–point formula (Laplacian scheme A(D2Q5)) using periodic boundary conditions where images (a) and (b) are at 1000th and 100000th time steps respectively.

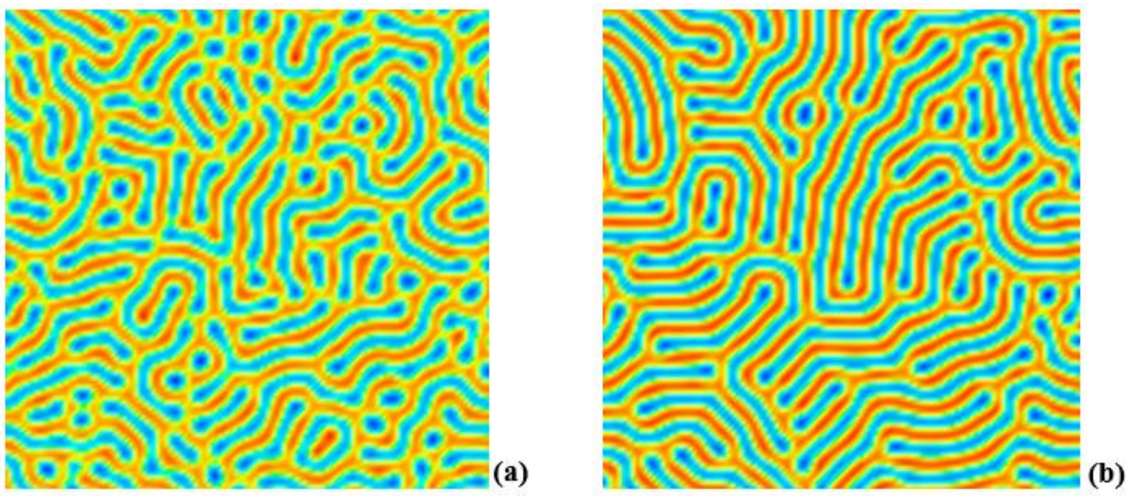


Figure 7.2: Generalized ADI method based on 5–point formula (Laplacian scheme A(D2Q5)) using periodic boundary conditions where images (a) and (b) are at 1000th and 100000th time steps respectively.

In Figures 7.1 and 7.2, the simulation results are shown for different stages of evolution of lamellae in a lamellar morphology of *A*-*B* diblock copolymer systems at different time steps. For the simulation results in Figure 7.2, the order parameter evolution takes place successfully, as can be observed from these results. The microphase separation and the lamellae formations can also be seen.

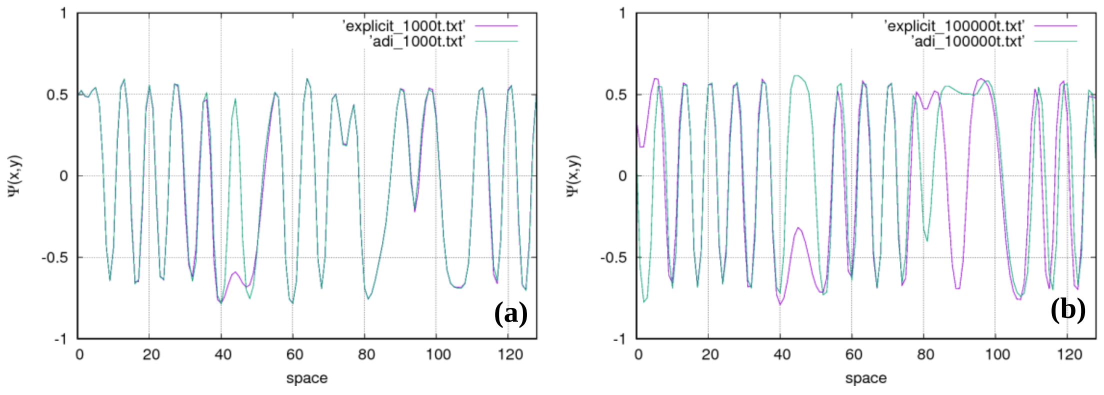


Figure 7.3: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 1000th and 100000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler and generalized ADI methods. The numerical values plotted here were obtained from the simulations shown in Figure 7.1 and 7.2.

Figure 7.3 is constructed in the similar way as Figure 5.7, where the distribution of the numerical values of order parameter ψ obtained from 2D simulations for forward Euler and generalized ADI methods are shown for comparison. In Figure 7.3 (a), the values are compared at 1000th time step and in Figure 7.3 (b) the values are compared at 100000th time step. Two lines of different colours show two different methods and each line represents the numerical values for an order parameter. The order parameter values are shown on the vertical axis against the grid size given on the horizontal axis. The two lines, coloured pink for the forward Euler method and green for the ADI method, show the comparison of the numerical values. In Figure 7.3 (a), the numerical values differ between 40 and 50, and except this range, both lines are exactly parallel, which illustrates that the tendency of numerical values obtained from the two methods is the same while the methodologies are different. Figure 7.3 (b) shows that the numerical values differ between 40 and 60 and between 80 and 100 on the horizontal axis, but the two lines grow parallel except in these regions. The parallel distribution of the numerical values on the same scale in Figure 7.3 for the two different methods show that there is no obvious difference for the evolution of the order parameter and therefore the lamellae formations are almost the same in pattern for both methods, which can be observed by comparing Figures 7.1 and Figure 7.2.

In Figure 7.4 the simulation results were obtained using 256×256 grid size and time interval $\Delta t = 1.0$ in the generalized ADI method for CDS. The system parameters used are given in Table 7.1. The purpose of using 256×256 grid size is to show that the generalized ADI method has no memory issue, while the CN method does not work for such a grid size. On the other hand, the use of time interval $\Delta t = 1.0$ indicates that this method is unconditionally stable for use at any time interval value, unlike the explicit forward Euler method.

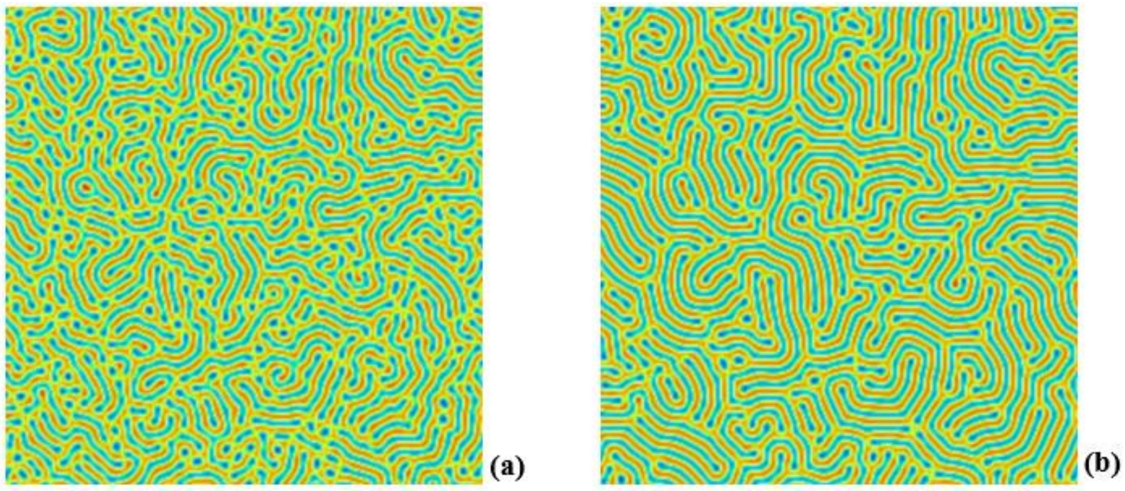


Figure 7.4: Generalized ADI method based on 5–point formula using 256×256 grid size and time interval $\Delta t = 1.0$ where images (a) and (b) are at 1000th and 100000th time steps respectively.

The generalized ADI method is further extended to three–dimensional simulations for CDS. Equation (7.4) can be written in three dimensions as follows:

$$\psi_{i,j,k}^{n+1} = \underbrace{\psi_{i,j,k}^{n} - \Delta t D \nabla^2 (\nabla^2) \psi_{i,j,k}^{n} - \Delta t B \psi_{i,j,k}^{n}}_{part1} - \underbrace{\Delta t \nabla^2 f(\psi_{i,j,k}^{n})}_{part2}. \tag{7.9}$$

The generalized ADI operator split form given in equation (7.5) is given in equation (7.10) as a three–dimensional ADI method for the CDS equation (7.9).

$$\begin{aligned} L_x w &= \psi_{i,j,k}^{n} - \Delta t D \nabla^4 \psi_{i,j,k}^{n} - \Delta t B \psi_{i,j,k}^{n} - \Delta t \nabla^2 f(\psi_{i,j,k}^{n}), \\ L_y v &= w, \\ L_z u &= v \\ \psi_{i,j,k}^{n+1} &= \psi_{i,j,k}^{n} + u. \end{aligned} \tag{7.10}$$

Equation (7.10) can be seen in four steps and in the first step the explicit part is on the right hand side. In the first step of equation (7.10), the one–dimensional vector of size N is calculated explicitly if the grid size is given $N \times N \times N$ and on the left hand side the values are approximated for vector w implicitly by using $L_x = M$ matrix in the x–direction. The other steps are approximated in the same way but in the y and z–directions, except the last step.

In Figures 7.5 and 7.6, the 3D simulation shows the results which were obtained for the two different methods: the generalized ADI and the explicit forward Euler methods respectively. These simulations were obtained using the parameters given in Table 7.1 and the specifications are given as follows:

- The grid size $64 \times 64 \times 64$ was chosen, with grid spacing $\Delta x = \Delta y = 1$;
- The total time of the simulations was up to 10000 time steps;
- The simulations were run with periodic boundary conditions;
- The simulations were started from an initial random disordered state $\psi = \pm 0.3$.

The generalized ADI method overcomes memory usage, unlike the CN method. Because of the limitation of the CN method, the three–dimensional simulations cannot be performed unless using the fastest computers with sufficient memory. The 3D simulation results shown in Figures 7.5 and 7.6 were obtained using 7–point Laplacian scheme M. It must be noted that the 3D simulation results obtained based on the 7–point Laplacian scheme M for spherical morphology were found to be anisotropic and generally the 7–point stencil was also found to be anisotropic. Therefore, the 3D results shown in Figures 7.5 and 7.6 were also anisotropic. These simulation results show the evolution of the order parameter in a microphase separation of *A-B* diblock copolymer and due to the anisotropy of the Lapalcian scheme used, the lamellae formations are not well aligned and healthy. The 3D implementation of the generalized ADI method for CDS remained successful, thus the simulation results shown in Figure 7.6 were obtained by using a more stable method compared to the forward Euler method. It is well known and discussed in previous chapters that the forward Euler method is not more stable in terms of time step value. The graphs of comparison are given in Figure 7.7 for one order parameter evolution using two different methods in three–dimensions.

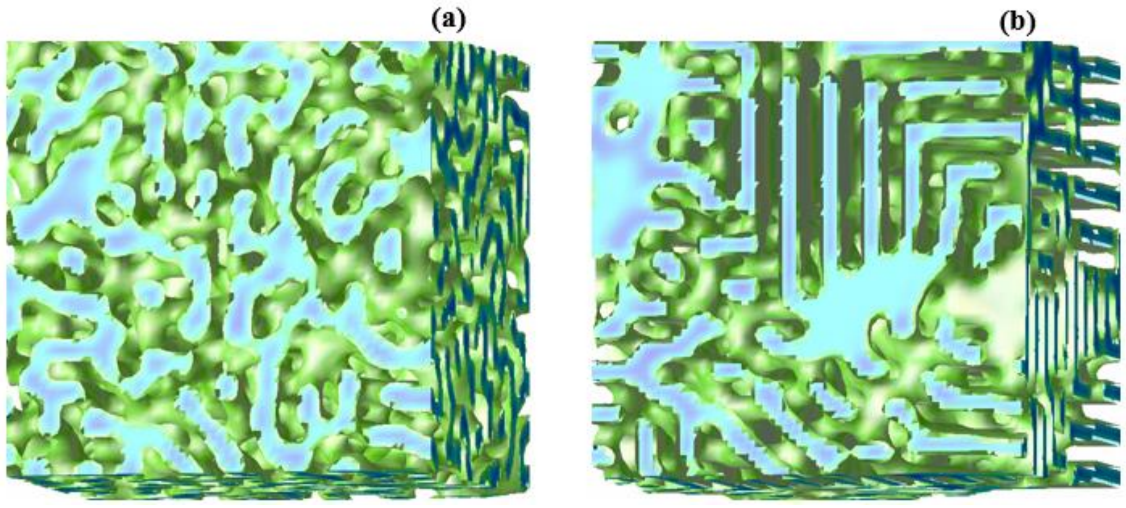


Figure 7.5: Three-dimensional results using explicit method based on 7–point formula using $64 \times 64 \times 64$ grid size and time interval $\Delta t = 0.1$ where images (a) and (b) are at 1000th and 10000th time steps respectively.

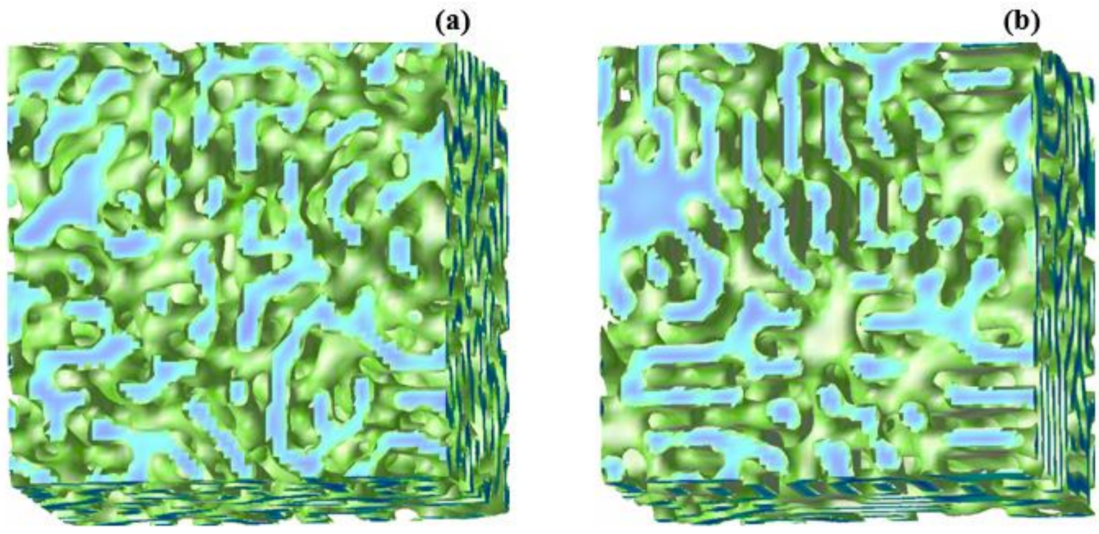


Figure 7.6: Three–dimensional results using generalized ADI method based on 7–point formula using $64 \times 64 \times 64$ grid size and time interval $\Delta t = 1.0$ where images (a) and (b) are at 1000th and 10000th time steps respectively.

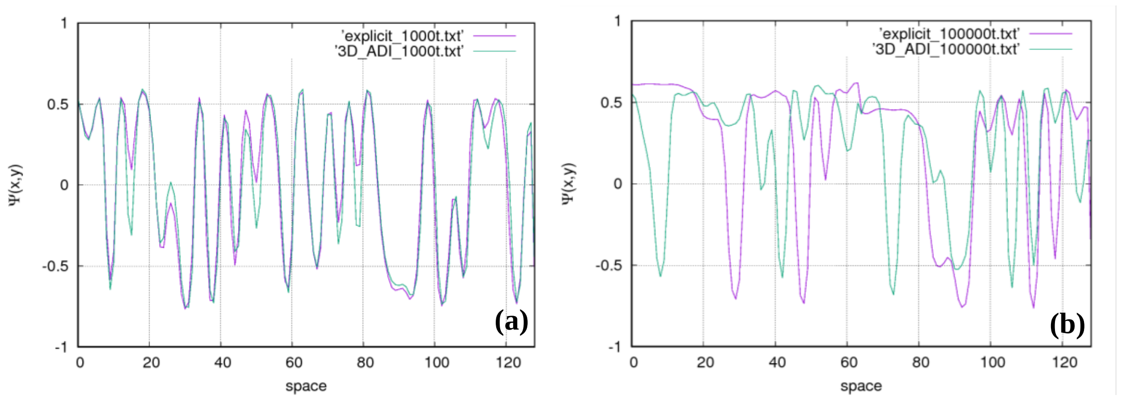


Figure 7.7: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 1000th and 100000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler and generalized ADI methods. The numerical values plotted here were obtained from the 3D simulations shown in Figure 7.5 and 7.6.

Figure 7.7 is similar to Figure 7.3, where the distribution of the numerical values of order parameter ψ obtained from the 3D simulations (Figures 7.5 and 7.6) for the forward Euler and generalized ADI methods are shown for comparison. In Figure 7.7 (a), the values are compared at the 1000th time step and in Figure 7.7 (b) the values are compared at the 100000th time step. Two lines of different colours show the two different methods and each line represents the numerical values for an order parameter. In Figure 7.7 (a), the lines of the numerical values are exactly parallel, which illustrates that the tendency of the numerical values obtained from the two methods is the same, while the methodologies are different. Figure 7.7 (b) shows that the numerical values differ at many places on the horizontal axis. It must be noted that when the simulations were run for both methods, no divergence of values was observed. Sometimes, due to the inappropriate use of time interval value Δt in a method, the divergence of values occurs and no more simulation images can be developed. The change of values shown in Figure 7.7 (b) is due to the use of time interval $\Delta t = 1.0$ and this time step value does not work for the 7–point stencil in forward Euler method. Using both methods, the microphase separation takes place and the generalized ADI method is more stable because of allowing more space for the time step value. This flexibility of using a larger time step value enhances the speed of execution for certain methods, making it more stable and fast.

7.1.2 Hundsdorfer's ADI method

CDS equation (7.2) contains non–homogeneous terms and therefore the generalized ADI method is insufficient to achieve second order accuracy. Witelski and Bowen [107] suggest iterative methods at each time step to achieve second–order accuracy for non–linear problems. These iterative methods can be Newton's method or the pseudo–linear factorization method [107]. Calatroni *et al*. [108] implemented the ADI method for the nonlinear partial differential equations by employing Hundsdorfer's ADI method [123]. Hundsdorfer's ADI method is an extension to Douglas's method [124, 125] by adding another parameter $\sigma > 1$ for stabilising the method. To achieve second–order accuracy in time step and space step, at each time step two calculations were performed. The steps needed to solve CDS equation (7.2) using this method, and are given as follows:

$$Y_0^1 = \psi_{j,k}^n - \Delta t D \nabla^4 \psi_{j,k}^n - \Delta t B \psi_{j,k}^n - \Delta t \nabla^2 f(\psi_{j,k}^n), \tag{7.11a}$$

$$L_x Y_1^1 = Y_0^1 + \Delta_x^4 \psi_{j,k}^n, \tag{7.11b}$$

$$L_y Y_2^1 = Y_1^1 + \Delta_y^4 \psi_{j,k}^n, \tag{7.11c}$$

$$\psi_{j,k}^{n+1/2} = Y_2^1, \tag{7.11d}$$

$$\begin{aligned} \tilde{Y}_0^1 = Y_0^1 - \Delta t D \nabla^4 \psi_{j,k}^{n+1/2} - \Delta t \nabla^2 f(\psi_{j,k}^{n+1/2}) \\ - \sigma(\Delta t D \nabla^4 \psi_{j,k}^n - \Delta t \nabla^2 f(\psi_{j,k}^n) - \Delta t B \psi_{j,k}^n), \end{aligned} \tag{7.11e}$$

$$L_x \tilde{Y}_1^1 = \tilde{Y}_0^1 + \Delta_x^4 \psi_{i,j}^{n+1/2}, \tag{7.11f}$$

$$L_y \tilde{Y}_2^1 = \tilde{Y}_1^1 + \Delta_y^4 \psi_{i,j}^{n+1/2}, \tag{7.11g}$$

$$\psi_{j,k}^{n+1} = \tilde{Y}_2^1. \tag{7.11h}$$

The stabilizing parameter is set at $\sigma = 1$, the intermediate solution is obtained in (7.11d) and the complete approximation is obtained in equation (7.11h). At each time step, two calculations are carried out in each dimension. The above ADI method for equation (7.11) can be seen in equations (3.4) and (4.4) in [108]. The ADI method in equation (7.11) is unconditionally stable and the stability properties of this method are discussed in [126]. The simulation results are given in Figure 7.8 over a 128×128 grid using all the same parameters used in this chapter for other simulations. The comparison graphs between the explicit method and Hundsdorfer's method are given in Figure 7.9. The simulation results in Figure 7.8 show the different stages of evolution of order parameter in a microphase separation of *A*–*B* diblock copolymer systems at different time steps. The CDS equations involve a biharmonic operator, and the CN being stable and second–order accurate in time, uses the sparse matrices to accommodate the biharmonic operator. Due to this, the CDS programs based on the CN method execute for a longer time. Hundsdorfer's ADI method has the same properties as the CN method; it is a second–order accurate and unconditionally stable method. The aided advantage of this method is that this is faster in execution than the CN method. The results given in Figure 7.8 show that the simulations were executed successfully, based on Hundsdorfer's method for a 5–point Laplacian scheme. The implementation of Hundsdorfer's ADI method for CDS makes the CDS numerically more stable, fast and accurate. The simulation results given in Figure 7.8 were executed for the time interval value $\Delta t = 0.1$. The simulations were also executed for $\Delta t = 1.0$ and this method can also be used for the maximum grid size.

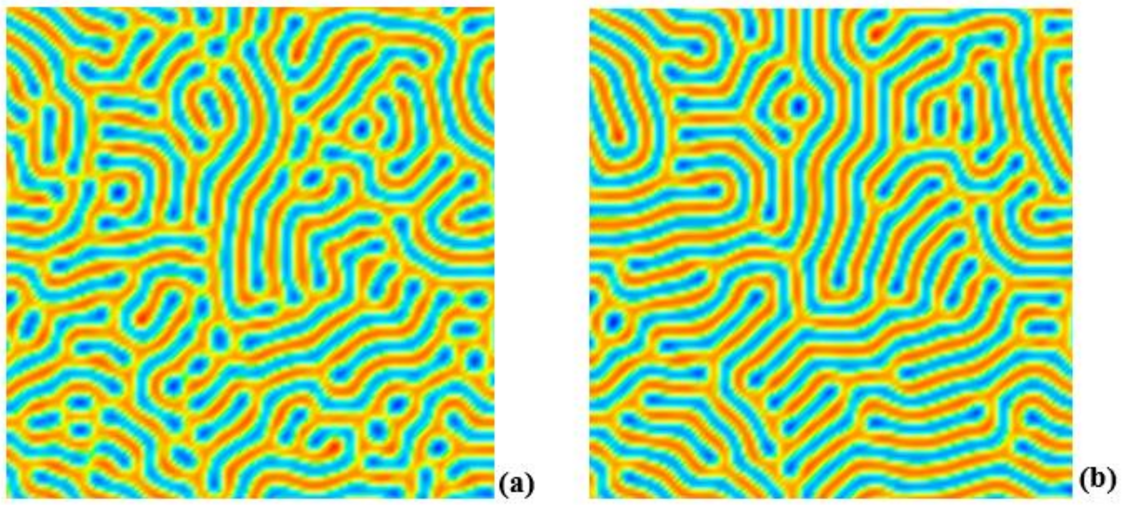


Figure 7.8: Hundsdorfer's ADI method based on 5–point formula (Laplacian scheme A) using periodic boundary conditions where images (a) and (b) at 1000th and 100000th time steps respectively.

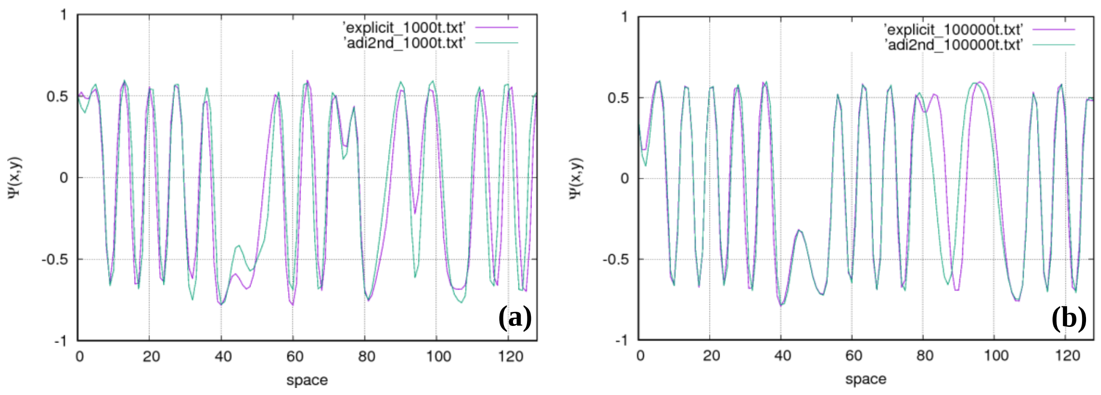


Figure 7.9: The numerical values of order parameter $\psi(x,y)$ are plotted against the space (0 – 128) for 1000th and 100000th time steps in images (a) and (b) respectively. Numerical values of $\psi(x,y)$ are shown for the forward Euler and Hundsdorfer's ADI methods. The numerical values plotted here were obtained from the simulations shown in Figure 7.1 and 7.8.

In Figure 7.9, the distribution of numerical values of order parameter ψ obtained from 2D simulations (Figures 7.1 and 7.8) for the forward Euler and Hundsdorfer's ADI methods are shown for comparison. In Figure 7.9 (a), the values are compared at the 1000th time step and in Figure 7.9 (b) the values are compared at the 100000th time step. In Figure 7.9 (a), the lines of the numerical values are exactly parallel, which illustrates that the tendency of the numerical values obtained from the two methods is the same, while the methodologies are different. A negligible difference in the two lines can be observed in Figure 7.9 (b) between 80 and 100 on the horizontal axis. The parallel distribution of numerical values on the same scale in Figure 7.9 for the two different methods shows that there is no obvious difference for the evolution of the order parameter and therefore the lamellae formations are almost similar in pattern for both methods, which can be observed by comparing Figures 7.1 and 7.8. Also, the simulation results given in Figure 7.2, which were obtained from the generalized ADI method, can be compared with the simulation results given in Figure 7.8. The simulation results in all these figures are similar, but the results shown in Figure 7.8 were obtained by a stronger and more stable method.

7.2 Conclusions

Two different ADI methods have been discussed and implemented for the cell dynamics simulation technique using the same parameters and specifications which were used in the explicit forward Euler method. Two- and three–dimensional simulations were executed using the generalized ADI method. The generalized ADI method was executed using different time interval values which showed that the generalized ADI method is unconditionally stable. The generalized ADI method is first–order accurate and the implementation of the generalized ADI method was not sufficient for CDS to achieve second–order accuracy because the CDS equations involve nonlinear terms. Therefore, the Hundsdorfer's ADI method was implemented, which is second–order accurate. Both these methods are unconditionally stable. In this work, the new CDS models based on the ADI methods were developed using the basic formula of the 5–point Laplacian operator, but these can be further extended by using 9–point isotropic Laplacian schemes in 2D and 27–point isotropic Laplacian schemes in 3D. The CDS technique for two order parameter systems can also be implemented using Hundsdorfer's ADI method in two and three dimensions.

Chapter Eight

8 Conclusions and Future Works

8.1 Conclusions

The achievements of the objectives set out for this study are summarized in this section. The conclusions drawn from the findings of this study are given as follows:

- The 2D 9–point isotropic stencil operators (BV(D2Q9) in three cases) were derived. These are novel isotropic stencil operators presented in this study which are more efficient.
- The stencils PK(D2Q9), BV(D2Q9)$_{case2}$ and BV(D2Q9)$_{case3}$ in 2D 9–point family Laplacians of second–order were found to be isotropic and, among these stencils, the BV(D2Q9)$_{cas2}$ was found to be optimally good in isotropy.
- In 3D, the 19–point stencil (D3Q19) was found to be more isotropic and more stable due to allowing a larger time step value for Δt.
- The stencils OP(D2Q9) and BV(D2Q9)$_{case1}$ in 2D and SO(D3Q27) and BV(D3Q27) in 3D have been found to be slightly anisotropic on the whole range **k,** but because of enabling larger time steps, these can be considered as valid alternatives.
- Various 2D Laplacian schemes were employed in the CDS framework to analyse the isotropic results in lamellar morphology and in macrophase separation of a binary blend. The anisotropic 2D 5–point Laplacian A(D2Q5) did not perform well in simulations of lamellar morphology as compared to isotropic Laplacians.
- The simulations of a binary blend based on Laplacian A(D2Q5) yielded the rectangular shapes of rich domains of *A* blcoks (red coloured subdomains) which were found different compred to simulations of isotropic Laplacians.

- The simulation snapshots obtained by using 2D 9–point Laplacian scheme PK(D2Q9) depicted perfect lamellae formations in microphase separation and circular shapes of A rich domains in macrophase separation, as with that of Oono and Puri's Laplacian scheme OP(D2Q9). The 2D 9–point isotropic stencil operators derived from the method of B.A.C. van Vlimmeren produced stable simulation results in both phenomena.
- The results showed that two–dimensional Laplacian schemes D2Q5 (equation (3.57)), D2Q9 (equation (3.60) and BK(D2Q9) (equation (3.61)) are unstable for simulations.
- The simulations results obtained by using 2D 9–point star Laplacain scheme $(D2Q9)_{star}$ and 17–point D2Q17 were found badly anistropic for the macrophase sepration of a binary blend. The simulations based on these stencils took longer time for executions compared to isotropic 9–point family Laplcains.
- In 3D simulations, the Laplacian schemes D3Q19, D3Q27 and PK(D3Q27) were found to be stable for simulation results for the evolution of order parameter in a spherical morphology. The well formed and well aligned spherical particles in circular shapes were observed based on these Laplacains. Laplacian scheme BV(D3Q27) also produced stable results compared to the original CDS choice for Laplacian scheme SO(D3Q27) of Shinozaki and Oono.
- In 3D simulations, the Laplacian schemes D3Q7 and D3Q15 were found to be anistropic. Mostly the mixed particles and rectangular shapes were observed in a spherical morphology.
- Novel models of CDS have been developed in this study by implementing implicit finite difference schemes which include backward Euler and Crank–Nicolson. The results obtained from the CN method were compared with the explicit

forward Euler and the implicit backward Euler methods, and the order parameter evolution were the same in both methods.

- In CN methodology for CDS based on one-order parameter evolution, a 9–point isotropic Laplacian operator OP(D2Q9) was successfully employed and the simulation results obtained were the same as those obtained for the explicit forward Euler method.
- In both the schemes, however, the implicit backward Euler and the CN schemes were stable but were found to be very slow in comparison to the forward Euler method.
- Two–dimensional simulations were possible for the implicit backward Euler and CN methods based on CDS. The whole work was limited to a grid of size 128 x 128; for a larger grid or three–dimensional simulations, these implicit schemes have limitations of computer memory due to the huge sparse matrices.
- The CDS for a two-order parameter system was implemented in the Crank–Nicolson scheme.
- The 9–point isotropic Laplacian operator OP(D2Q9) of Oono and Puri's choice was employed in the implementation of the CN scheme for CDS, based on two-order parameter systems; the simulation results obtained were the same as those obtained for the explicit forward Euler method.
- In the CDS technique conducted by Aya Ito for domain patterns in copolymer and homopolymer mixtures, the time step value $\Delta t = 0.5$ was chosen to avoid numerical instability. If any other Δt value was used, the system became unstable. By the implementation of implicit schemes, especially the CN method, it was possible to obtain these simulations for choosing $\Delta t = 1$.

- Another novel model of CDS was developed in this study by the implementation of the Alternating Direction Implicit (ADI) method for the modelling of one-order parameter in the lamellar forming of *A*–*B* diblock copolymer systems. The results obtained were compared with the forward Euler method.
- As the generalized ADI method is first–order accurate, to achieve second–order accuracy, the Hundsdorfer's ADI method was employed, which is also unconditionally stable in terms of time step. The simulation results obtained by using both methods were same as those obtained from forward Euler method.
- Generally, the ADI method is faster than the CN method and the implementation of the ADI method for CDS makes the CDS more stable, faster and more robust.

8.2 Future works

The application of the research work conducted in this study and future work suggestions are presented as follows:

- The mathematical methods implemented for the CDS in this work can be applied to the modelling of soft matters by modifying map function. These can be extended to the modelling of nano–structures in the field of soft materials and nano–technology.
- The CDS based on implicit methods, e.g. CN and ADI, may also be applied to widespread applications such as the modelling of reaction–diffusion systems for studying chemical reactions and population dynamics, the investigation of spinodal decomposition, the simulation of microemulations and binary blends containing surfactants or hard particles, cross-linked polymer blends, and so on.

- The CDS methods developed in this study can be applied to investigate three-order parameter systems and their properties based on a mixture of two more solvents.
- The CDS based on implicit methods developed in this study will have wider applications (in terms of computer simulations) in the field of bio–mimicking.
- The ADI method as an independent and fast algorithm is preferred for parallel computing and therefore the ADI implementation for CDS enables it in the incorporation of parallel computing where large and very complex problems can be simulated via CDS and molecular information can be gained.
- The CDS based on the ADI method can be extended to current multi-core implementation. Such algorithms for CDS method on distributed memory architecture can solve the scalability issue and can also execute a larger domain size of the CDS without giving any consideration to memory limitations.
- Work can be carried out for CDS in the CN scheme including the shear flow and noise terms in the modelling of diblock copolymers with different morphologies.
- The ADI method for CDS can be developed for 2D simulations based on a 9–point Laplacian operator and for 3D simulations, including the shear flow and noise terms.
- The CDS technique for two-order parameter systems can also be implemented using the ADI method in 2D and 3D and the executions of such system can be faster and more stable.
- Using implicit CDS methods, various other morphologies can be investigated which include spheres, hexagonally-packed cylinders, and more complex structures such as gyroids.

- A new mathematical model for the CDS, based on the finite volume method, can be designed so that the limitations of structured/regular grids can be addressed and the accuracy of the results can be improved.

References

[1] I. W. Hamley, *Introduction to Soft Matter: Synthetic and Biological Self-Assembling Materials.* John Wiley & Sons, 2013.

[2] P. Marco, *Mesoscale Modelling of Block Copolymer Systems.* Germany: VDM Verlag Dr. Muller Aktiengesellschaft, 2010.

[3] Y. Oono and S. Puri, "Computationally efficient modeling of ordering of quenched phases," *Phys. Rev. Lett.,* vol. 58, pp. 836, 1987.

[4] M. Pinna and A. Zvelindovsky, "Large scale simulation of block copolymers with cell dynamics," *The European Physical Journal B,* vol. 85, pp. 1-18, 2012.

[5] M. Cheng and A. D. Rutenberg, "Maximally fast coarsening algorithms," *Physical Review E,* vol. 72, pp. 055701, 2005.

[6] S. P. Thampi, S. Ansumali, R. Adhikari and S. Succi, "Isotropic discrete Laplacian operators from lattice hydrodynamics," *Journal of Computational Physics,* vol. 234, pp. 1-7, 2013.

[7] M. Patra and M. Karttunen, "Stencils with isotropic discretization error for differential operators," *Numerical Methods for Partial Differential Equations,* vol. 22, pp. 936-953, 2006.

[8] D. J. Duffy, *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach.* John Wiley & Sons, 2013.

[9] S. P. Thampi, I. Pagonabarraga and R. Adhikari, "Lattice-Boltzmann-Langevin simulations of binary mixtures," *Physical Review E,* vol. 84, pp. 046709, 2011.

[10] D. D. Chung, *Applied Materials Science: Applications of Engineering Materials in Structural, Electronics, Thermal, and Other Industries.* CRC Press, 2001.

[11] V. Zackay, E. Parker, J. Morris and G. Thomas, "The application of materials science to the design of engineering alloys. A Review," *Materials Science and Engineering,* vol. 16, pp. 201-221, 1974.

[12] National Research Council staff, *Mathematical Research in Material Science.* National Academies Press, 1993.

[13] S. C. Glotzer and W. Paul, "Molecular and mesoscale simulation methods for polymer materials," *Annual Review of Materials Research,* vol. 32, pp. 401-436, 2002.

[14] X. He, M. Song, H. Liang and C. Pan, "Self-assembly of the symmetric diblock copolymer in a confined state: Monte Carlo simulation," *J. Chem. Phys.,* vol. 114, pp. 10510-10513, 2001.

[15] B. Hayes, "The Science of Computing: The Wheel of Fortune," *Am. Sci.,* vol. 81, pp. 114-118, 1993.

[16] U. Landman, R. Barnett and W. Luedtke, "Simulations of materials: from electrons to friction," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences,* vol. 341, pp. 337-350, 1992.

[17] F. H. Stillinger and T. A. Weber, "Molecular dynamics study of chemical reactivity in liquid sulfur," *J. Phys. Chem.,* vol. 91, pp. 4899-4907, 1987.

[18] R. Evans, "The nature of the liquid-vapour interface and other topics in the statistical mechanics of non-uniform, classical fluids," *Adv. Phys.,* vol. 28, pp. 143-200, 1979.

[19] M. Wertheim, "Integral equation for the Smith–Nezbeda model of associated fluids," *J. Chem. Phys.,* vol. 88, pp. 1145-1155, 1988.

[20] J. Gunton, M. San Miguel, P. S. Sahni, C. Domb and J. Lebowitz, "Phase transitions and critical phenomena," 1983.

[21] E. Helfand, S. M. Bhattacharjee and G. H. Fredrickson, "Molecular weight dependence of polymer interfacial tension and concentration profile," *J. Chem. Phys.,* vol. 91, pp. 7200-7208, 1989.

[22] K. M. Hong and J. Noolandi, "Theory of phase equilibriums in systems containing block copolymers," *Macromolecules,* vol. 16, pp. 1083-1093, 1983.

[23] A. A. Wheeler, W. J. Boettinger and G. B. McFadden, "Phase-field model for isothermal phase transitions in binary alloys," *Physical Review A,* vol. 45, pp. 7424, 1992.

[24] A. Bray, "Theory of phase ordering kinetics," *Physica A: Statistical Mechanics and its Applications,* vol. 194, pp. 41-52, 1993.

[25] C. Harrison, D. H. Adamson, Z. Cheng, J. M. Sebastian, S. Sethuraman, D. A. Huse, R. A. Register and P. M. Chaikin, "Mechanisms of ordering in striped patterns," *Science,* vol. 290, pp. 1558-1560, Nov 24, 2000.

[26] S. Puri and N. Parekh, "Non-algebraic domain growth in binary alloys with quenched disorder," *Journal of Physics A: Mathematical and General,* vol. 25, pp. 4127, 1992.

[27] L. Tsarkova, G. A. Sevink and G. Krausch, "Nanopattern evolution in block copolymer films: Experiment, simulations and challenges," in *Complex Macromolecular Systems I*Anonymous Springer, 2010, pp. 33-73.

[28] H. Emmerich, "Advances of and by phase-field modelling in condensed-matter physics," *Adv. Phys.,* vol. 57, pp. 1-87, 2008.

[29] A. N. Singh, R. D. Thakre, J. C. More, P. K. Sharma and Y. Agrawal, "Block Copolymer Nanostructures and Their Applications: A Review," *Polym. Plast. Technol. Eng.,* vol. 54, pp. 1077-1095, 2015.

[30] T. P. Lodge, "Block copolymers: past successes and future challenges," *Macromolecular Chemistry and Physics,* vol. 204, pp. 265-273, 2003.

[31] S. Tallegas, T. Baron, G. Gay, C. Aggrafeil, B. Salhi, T. Chevolleau, G. Cunge, A. Bsiesy, R. Tiron and X. Chevalier, "Block copolymer technology applied to nanoelectronics," *Physica Status Solidi (C),* vol. 10, pp. 1195-1206, 2013.

[32] M. A. Hillmyer, "Nanoporous materials from block copolymer precursors," in *Block Copolymers II*Anonymous Springer, 2005, pp. 137-181.

[33] A. Urbas, R. Sharp, Y. Fink, E. L. Thomas, M. Xenidou and L. J. Fetters, "Tunable block copolymer/homopolymer photonic crystals," *Adv Mater,* vol. 12, pp. 812-814, 2000.

[34] D. J. Arriola, E. M. Carnahan, P. D. Hustad, R. L. Kuhlman and T. T. Wenzel, "Catalytic production of olefin block copolymers via chain shuttling polymerization," *Science,* vol. 312, pp. 714-719, May 5, 2006.

[35] C. Park, J. Yoon and E. L. Thomas, "Enabling nanotechnology with self assembled block copolymer patterns," *Polymer,* vol. 44, pp. 6725-6760, 2003.

[36] R. A. Segalman, "Patterning with block copolymer thin films," *Materials Science and Engineering: R: Reports,* vol. 48, pp. 191-226, 2005.

[37] J. Dawkins, "Block copolymers: synthetic strategies, physical properties and applications. N Hadjichristidis, S Pispas and GA Floudas. John Wiley & Sons, Ltd, Chichester, UK, 2002. pp 440, ISBN 0-471-39436-x," *Polym. Int.,* vol. 53, pp. 232-232, 2004.

[38] F. S. Bates, "Polymer-polymer phase behavior," *Science,* vol. 251, pp. 898-905, Feb 22, 1991.

[39] X. Li, J. Guo, Y. Liu and H. Liang, "Microphase separation of diblock copolymer poly (styrene-b-isoprene): A dissipative particle dynamics simulation study," *J. Chem. Phys.,* vol. 130, pp. 74908, 2009.

[40] S. Lin, N. Numasawa, T. Nose and J. Lin, "Brownian molecular dynamics simulation on self-assembly behavior of rod-coil diblock copolymers," *Macromolecules,* vol. 40, pp. 1684-1692, 2007.

[41] S. Ren and I. Hamley, "Cell dynamics simulations of microphase separation in block copolymers," *Macromolecules,* vol. 34, pp. 116-126, 2001.

[42] T. L. Chantawansri, A. W. Bosse, A. Hexemer, H. D. Ceniceros, C. J. García-Cervera, E. J. Kramer and G. H. Fredrickson, "Self-consistent field theory simulations of block copolymer assembly on a sphere," *Physical Review E,* vol. 75, pp. 031802, 2007.

[43] G. Gonnella, E. Orlandini and J. Yeomans, "Spinodal decomposition to a lamellar phase: effects of hydrodynamic flow," *Phys. Rev. Lett.,* vol. 78, pp. 1695, 1997.

[44] G. Gonnella, E. Orlandini and J. Yeomans, "Lattice Boltzmann simulations of lamellar and droplet phases," *Physical Review E,* vol. 58, pp. 480, 1998.

[45] S. Puri and H. Frisch, "Segregation dynamics of binary mixtures with simple chemical reactions," *Journal of Physics A: Mathematical and General,* vol. 27, pp. 6027, 1994.

[46] Y. Oono and S. Puri, "Study of phase-separation dynamics by use of cell dynamical systems. I. Modeling," *Physical Review A,* vol. 38, pp. 434, 1988.

[47] A. Shinozaki and Y. Oono, "Spinodal decomposition in 3-space," *Physical Review E,* vol. 48, pp. 2622, 1993.

[48] S. Puri and Y. Oono, "Study of phase-separation dynamics by use of cell dynamical systems. II. Two-dimensional demonstrations," *Physical Review A,* vol. 38, pp. 1542, 1988.

[49] I. W. Hamley, "Cell dynamics simulations of block copolymers," *Macromolecular Theory and Simulations,* vol. 9, pp. 363-380, 2000.

[50] N. Parekh and S. Puri, "A new numerical scheme for the Fisher equation," *Journal of Physics A: Mathematical and General,* vol. 23, pp. L1085, 1990.

[51] H. Furukawa, "A dynamic scaling assumption for phase separation," *Adv. Phys.,* vol. 34, pp. 703-750, 1985.

[52] M. San Miguel, M. Grant and J. D. Gunton, "Phase separation in two-dimensional binary fluids," *Physical Review A,* vol. 31, pp. 1001, 1985.

[53] M. Bahiana and Y. Oono, "Cell dynamical system approach to block copolymers," *Physical Review A,* vol. 41, pp. 6763, 1990.

[54] H. Kodama and M. Doi, "Shear-induced instability of the lamellar phase of a block copolymer," *Macromolecules,* vol. 29, pp. 2652-2658, 1996.

[55] S. Ren, I. Hamley, G. Sevink, A. Zvelindovsky and J. Fraaije, "Mesoscopic simulations of lamellar orientation in block copolymers," *Macromolecular Theory and Simulations,* vol. 11, pp. 123-127, 2002.

[56] T. Uneyama and M. Doi, "Density functional theory for block copolymer melts and blends," *Macromolecules,* vol. 38, pp. 196-205, 2005.

[57] T. Uneyama, "Density functional simulation of spontaneous formation of vesicle in block copolymer solutions," *J. Chem. Phys.,* vol. 126, pp. 114902, 2007.

[58] T. Ohta and K. Kawasaki, "Equilibrium morphology of block copolymer melts," *Macromolecules,* vol. 19, pp. 2621-2632, 1986.

[59] T. Ohta and A. Ito, "Dynamics of phase separation in copolymer-homopolymer mixtures," *Physical Review E,* vol. 52, pp. 5250, 1995.

[60] T. Ohta and K. Kawasaki, "Comment on the free energy functional of block copolymer melts in the strong segregation limit," *Macromolecules,* vol. 23, pp. 2413-2414, 1990.

[61] N. Maurits, J. Fraaije, P. Altevogt and O. Evers, "Simple numerical quadrature rules for Gaussian chain polymer density functional calculations in 3D and implementation on parallel platforms," *Computational and Theoretical Polymer Science,* vol. 6, pp. 1-8, 1996.

[62] A. A. Joshi, D. W. Shattuck, P. M. Thompson and R. M. Leahy, "A parameterization-based numerical method for isotropic and anisotropic diffusion smoothing on non-flat surfaces," *IEEE Trans. Image Process.,* vol. 18, pp. 1358-1365, 2009.

[63] B. Kamgar-Parsi, B. Kamgar-Parsi and A. Rosenfeld, "Optimally isotropic Laplacian operator," *IEEE Trans. Image Process.,* vol. 8, pp. 1467-1472, 1999.

[64] A. H. Panaretos, J. T. Aberle and R. E. Díaz, "The effect of the 2-D Laplacian operator approximation on the performance of finite-difference time-domain schemes for Maxwell's equations," *Journal of Computational Physics,* vol. 227, pp. 513-536, 2007.

[65] F. Xiao, X. Tang, L. Wang and H. Ma, "2-D isotropic finite difference time domain method," in *Microwave Conference Proceedings, 2005. APMC 2005. Asia-Pacific Conference Proceedings*, 2005, pp. 4 pp.

[66] A. Kumar, "Isotropic finite-differences," *Journal of Computational Physics,* vol. 201, pp. 109-118, 2004.

[67] C. Chow, "Discretization errors and rotational symmetry: the Laplacian operator on non-hypercubical lattices," *Nuclear Physics B,* vol. 547, pp. 281-302, 1999.

[68] A. Spitzbart and N. Macon, "Numerical differentiation formulas," *The American Mathematical Monthly,* vol. 64, pp. 721-723, 1957.

[69] R. L. Burden and J. D. Faires, "Numerical analysis," 1993.

[70] G. D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods.* Oxford university press, 1985.

[71] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems.* Siam, 2007.

[72] J. Crank and P. Nicolson, "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type," in *Mathematical Proceedings of the Cambridge Philosophical Society*, 1947, pp. 50-67.

[73] P. Wilmott, S. Howison and J. Dewynne, *The Mathematics of Financial Derivatives: A Student Introduction.* Cambridge University Press, 1995.

[74] P. Moin, *Fundamentals of Engineering Numerical Analysis.* Cambridge University Press, 2010.

[75] D. W. Peaceman and J. Rachford Henry H, "The numerical solution of parabolic and elliptic differential equations," *Journal of the Society for Industrial and Applied Mathematics,* vol. 3, pp. 28-41, 1955.

[76] S. Sirca and M. Horvat, "Computational Methods for Physicists," 2012.

[77] (). *The Heat equation in 2 and 3 spatial dimensions*. Available: http://www.cems.uvm.edu/~tlakoba/math337/notes_15.pdf.

[78] W. H. Press, *Numerical Recipes with Source Code CD-ROM 3rd Edition: The Art of Scientific Computing.* Cambridge University Press, 2007.

[79] I. J. Craig and A. D. Sneyd, "An alternating-direction implicit scheme for parabolic equations with mixed derivatives," *Comput. Math. Appl.,* vol. 16, pp. 341-350, 1988.

[80] W. Lee, "Tridiagonal matrices: Thomas algorithm," *MS6021, Scientific Computation, University of Limerick,* .

[81] J. Douglas, "Alternating direction methods for three space variables," *Numerische Mathematik,* vol. 4, pp. 41-63, 1962.

[82] J. Douglas and J. E. Gunn, "A general formulation of alternating direction methods," *Numerische Mathematik,* vol. 6, pp. 428-453, 1964.

[83] J. Douglas Jim, "On the Numerical Integration of ∂^2u∂x^2 ∂^2u∂y^2=∂u∂t by Implicit Methods," *Journal of the Society for Industrial and Applied Mathematics,* vol. 3, pp. 42-65, 1955.

[84] S. Conte, "Numerical solution of vibration problems in two space variables," *Pacific J.Math,* vol. 7, pp. 1535-1544, 1957.

[85] J. Douglas and H. H. Rachford, "On the numerical solution of heat conduction problems in two and three space variables," *Transactions of the American Mathematical Society,* vol. 82, pp. 421-439, 1956.

[86] M. Eres, L. Schwartz and R. Roy, "Fingering phenomena for driven coating films," *Physics of Fluids (1994-Present),* vol. 12, pp. 1278-1295, 2000.

[87] L. W. Schwartz, "Hysteretic effects in droplet motions on heterogeneous substrates: direct numerical simulation," *Langmuir,* vol. 14, pp. 3440-3453, 1998.

[88] L. W. Schwartz, R. V. Roy, R. R. Eley and S. Petrash, "Dewetting patterns in a drying liquid film," *J. Colloid Interface Sci.,* vol. 234, pp. 363-374, 2001.

[89] M. Eres, D. Weidner and L. Schwartz, "Three-dimensional direct numerical simulation of surface-tension-gradient effects on the leveling of an evaporating multicomponent fluid," *Langmuir,* vol. 15, pp. 1859-1871, 1999.

[90] Y. Oono, S. Puri, C. Yeung and M. Bahiana, "Cell dynamical system study of phase separation dynamics," *Journal of Applied Crystallography,* vol. 21, pp. 883-885, 1988.

[91] A. Shinozaki and Y. Oono, "Spinodal decomposition in a Hele-Shaw cell," *Physical Review A,* vol. 45, pp. R2161, 1992.

[92] T. Ohta, Y. Enomoto, J. L. Harden and M. Doi, "Anomalous rheological behavior of ordered phases of block copolymers. 1," *Macromolecules,* vol. 26, pp. 4928-4934, 1993.

[93] J. Feng and E. Ruckenstein, "Long-range ordered structures in diblock copolymer melts induced by combined external fields," *J. Chem. Phys.,* vol. 121, pp. 1609-1625, 2004.

[94] D. Hale, "Compact finite-difference approximations for anisotropic image smoothing and painting," *Matrix,* vol. 500, pp. D12, .

[95] S. Fomel and J. F. Claerbout, "Exploring three-dimensional implicit wavefield extrapolation with the helix transform," *Stanford Exploration Project,* vol. 95, pp. 43-60, 1997.

[96] (February 22, 2012). *Difference Between Isotropic and Anisotropic*. Available: http://www.differencebetween.net/science/chemistry-science/difference-between-isotropic-and-anisotropic/.

[97] T. Petrie and J. Randall, "Spherical isotropy representations," *Publications Mathématiques De L'IHÉS,* vol. 62, pp. 5-40, 1985.

[98] M. Abramowitz and I. A. Stegun, "Handbook of mathematical functions," *Applied Mathematics Series,* vol. 55, pp. 62, 1966.

[99] A. Kumar, "Isotropic averaging for cell-dynamical-system simulation of spinodal decomposition," *Pramana,* vol. 61, pp. 1-5, 2003.

[100] H. Tomita, "Preservation of isotropy at the mesoscopic stage of phase separation processes," *Progress of Theoretical Physics,* vol. 85, pp. 47-56, 1991.

[101] P. Teixeira and B. Mulder, "Comment on``Study of phase-separation dynamics by use of cell dynamical systems. I. Modeling''," *Physical Review E,* vol. 55, pp. 3789, 1997.

[102] B. Van Vlimmeren and J. Fraaije, "Calculation of noise distribution in mesoscopic dynamics models for phase separation of multicomponent complex fluids," *Comput. Phys. Commun.,* vol. 99, pp. 21-28, 1996.

[103] J. Fraaije, B. Van Vlimmeren, N. Maurits, M. Postma, O. Evers, C. Hoffmann, P. Altevogt and G. Goldbeck-Wood, "The dynamic mean-field density functional method and its application to the mesoscopic dynamics of quenched block copolymer melts," *J. Chem. Phys.,* vol. 106, pp. 4260-4269, 1997.

[104] T. Rogers, K. Elder and R. C. Desai, "Numerical study of the late stages of spinodal decomposition," *Physical Review B,* vol. 37, pp. 9638, 1988.

[105] A. Zvelindovsky and G. Sevink, "Sphere morphology of block copolymer systems under shear," *EPL (Europhysics Letters),* vol. 62, pp. 370, 2003.

[106] I. Rychkov, "Block copolymers under shear flow," *Macromolecular Theory and Simulations,* vol. 14, pp. 207-242, 2005.

[107] T. P. Witelski and M. Bowen, "ADI schemes for higher-order nonlinear diffusion equations," *Applied Numerical Mathematics,* vol. 45, pp. 331-351, 2003.

[108] L. Calatroni, B. Düring and C. Schönlieb, "ADI splitting schemes for a fourth-order nonlinear partial differential equation from image processing," *arXiv Preprint arXiv:1305.5362,* 2013.

[109] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* Siam, 1994.

[110] Z. Ling-Cui, S. Min-Na, P. Jun-Xing, W. Bao-Feng, Z. Jin-Jun and W. Hai-Shun, "Copolymer—homopolymer mixtures in a nanopore," *Chinese Physics B,* vol. 22, pp. 096401, 2013.

[111] A. Ito, "Domain patterns in copolymer-homopolymer mixtures," *Physical Review E,* vol. 58, pp. 6158, 1998.

[112] J. Zhang, G. Jin and Y. Ma, "Wetting-driven structure ordering of a copolymer/homopolymer/nanoparticle mixture in the presence of a modulated potential," *The European Physical Journal E,* vol. 18, pp. 359-365, 2005.

[113] A. L. Bertozzi, "The mathematics of moving contact lines in thin liquid films," *Notices of the AMS,* vol. 45, pp. 689-697, 1998.

[114] C. M. Elliott and D. A. French, "Numerical studies of the Cahn-Hilliard equation for phase separation," *IMA Journal of Applied Mathematics,* vol. 38, pp. 97-128, 1987.

[115] J. M. Hyman, B. Nicolaenko and S. Zaleski, "Order and complexity in the Kuramoto-Sivashinsky model of weakly turbulent interfaces," *Physica D,* vol. 23, pp. 265-292, 1986.

[116] A. Novick-Cohen and L. A. Segel, "Nonlinear aspects of the Cahn-Hilliard equation," *Physica D,* vol. 10, pp. 277-298, 1984.

[117] S. McKee and A. Mitchell, "Alternating direction methods for parabolic equations in two space dimensions with a mixed derivative," *The Computer Journal,* vol. 13, pp. 81-86, 1970.

[118] R. Mohanty and M. Jain, "High accuracy difference schemes for the system of two space nonlinear parabolic differential equations with mixed derivatives and variable coefficients," *J. Comput. Appl. Math.,* vol. 70, pp. 15-32, 1996.

[119] S. D. Conte and R. T. Dames, "On an alternating direction method for solving the plate problem with mixed boundary conditions," *Journal of the ACM (JACM),* vol. 7, pp. 264-273, 1960.

[120] S. D. Conte and R. T. Dames, "An alternating direction method for solving the biharmonic equation," *Mathematical Tables and Other Aids to Computation,* vol. 12, pp. 198-205, 1958.

[121] A. R. Mitchell and D. F. Griffiths, *The Finite Difference Method in Partial Differential Equations.* John Wiley, 1980.

[122] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations.* Siam, 2004.

[123] W. Hundsdorfer, "Accuracy and stability of splitting with stabilizing corrections," *Applied Numerical Mathematics,* vol. 42, pp. 213-233, 2002.

[124] van der Houwen, Piet J and J. G. Verwer, "One-step splitting methods for semi-discrete parabolic equations," *Computing,* vol. 22, pp. 291-309, 1979.

[125] W. Hundsdorfer and J. G. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations.* Springer Science & Business Media, 2013.

[126] K. In't Hout and B. Welfert, "Unconditional stability of second-order ADI schemes applied to multi-dimensional diffusion equations with mixed derivative terms," *Applied Numerical Mathematics,* vol. 59, pp. 677-692, 2009.

Appendix A

2D CDS Code (Default)

The Input File for the CDS Fortran Program. It is general for all methodologies for 2D.

```
'New simulation (0) or continue a previous one (1)'
0
'Input filename containing starting atomic configuration (max 80 c)
for 1'
restartfile.dat
'Insert D'
0.7d0
'Insert A'
1.5d0
'Insert B'
0.02d0
'Insert f'
0.48d0
'Insert Tau'
0.36d0
'Insert v'
2.3d0
'Insert u'
0.38d0
'Insert Grid size'
128,128,1
'Insert the deltat'
1.d0
'Total TimeSteps'
1000000
'Save order parameter configuration for restarting every ... steps'
1000
'Write pos-neg order parameter in the following file'
'final.bak'
'Saving pos-neg order parameter in the following steps(max 10;5 for
line):'
200,700,1000,3000,5000
10000,20000,50000,70000,100000
'Insert file record positive'
'final2.bak'
'Input the first name (you must input 8 characters)'
'cdsnew01'
'Input the second name (you must input 2 characters tau only 28)'
20
```

The CDS Fortran Program

```
c
c ****************************************************************
c           CDS 2D simulation source code
c ****************************************************************

        program order_parameter
        implicit none
        double precision pxi(0:300,0:300)
        double precision pxi0(0:300,0:300)
        double precision zxi(0:300,0:300)
        double precision apxi1(0:300,0:300)
        double precision aapxi1(0:300,0:300)
        double precision bapxi1(0:300,0:300)
        double precision capxi1(0:300,0:300)
        double precision apxi2(0:300,0:300)
        double precision aapxi2(0:300,0:300)
        double precision bapxi2(0:300,0:300)
        double precision capxi2(0:300,0:300)
        double precision f(0:300,0:300)
        double precision map1(0:300,0:300)
        double precision mxi1(0:300,0:300)
        double precision z(0:300)
        double precision h1,h2,h3
        double precision hx(0:100)
        double precision hy(0:100)
        double precision hz(0:100)
        double precision tau,v,u,omega,omega0
        double precision a,b,d,r,e,sh,e0,sh0,c1,c2,c3

c ****************************************************************
c           Boudary conditions declarations
c ****************************************************************

        integer upx(0:300),upy(0:300)
        integer downx(0:300),downy(0:300)
        double precision m(0:300)
        integer m1(0:300)

c ****************************************************************
c        Parameters for the CD simualtion
c***************************************************************
        integer i,j,k,s,nx,ny,nz,ex,ey,ez,seed,t,time,conf,ktime
        integer s1,s2,s3,s4,s5,s6,s7,s8,s9,s10
        integer everyconf,ftime,bcx,bcy,bcz,ht,R2
        character*80 label
        character*80 finalposition,fileconf,writeconf

c****************************************************************
c     Parameter to record data(order parameter)
c****************************************************************
        real delapse,dtime,t1(2)
        character*8 name1
        character*2 name2
        character*7 name3
        character*29 name4
        integer ma,esse2

c****************************************************************
c                 File open
```

```
c*****************************************************************

      open(unit=98,file = 'cds.in', status='old',form='formatted')

c*******************************************************
c     Read input data from file CDS.IN
c*******************************************************

        read(98,*) label
        read(98,*) conf
        read(98,*) label
        read(98,*) fileconf
        read(98,*) label
        read(98,*) d
        read(98,*) label
        read(98,*) a
        read(98,*) label
        read(98,*) b
        read(98,*) label
        read(98,*) r
        read(98,*) label
        read(98,*) tau
        read(98,*) label
        read(98,*) v
        read(98,*) label
        read(98,*) u
        read(98,*) label
        read(98,*) nx,ny,nz
        read(98,*) label
        read(98,*) eta
        read(98,*) label
        read(98,*) e
        read(98,*) label
        read(98,*) sh
        read(98,*) label
        read(98,*) omega
        read(98,*) label
        read(98,*) bcx,bcy,bcz
        read(98,*) label
        read(98,*) h1,h2,h3
        read(98,*) label
        read(98,*) deltat
        read(98,*) label
        read(98,*) time
        read(98,*) label
        read(98,*) everyconf
        read(98,*) label
        read(98,*) writeconf
        read(98,*) label
        read(98,*) s1,s2,s3,s4,s5
        read(98,*) s6,s7,s8,s9,s10
        read(98,*) label
        read(98,*) finalposition
        read(98,*) label
        read(98,*) name1
        read(98,*) label
        read(98,*) r2
```

```
c*****************************************************************
c           Random value initialization for Pxi
c*****************************************************************

       ktime=0
        do i = 1, nx
           do j =1, ny
                 call random_number(temp)
                 if (temp.ge.0 .and. temp.lt.0.5) then
                   Pxi(i,j) = 0.3d0
                 else
                   Pxi(i,j) = -0.3d0
                 endif
           enddo
        enddo

c*****************************************************************
c           Costant numbers
c*****************************************************************

        pi2=2.d0*dacos(-1.d0)
        ma=0
        name2=char(r2)
        write(name2,'(i2.2)') r2

c*****************************************************************
c           Laplacian weights
c*****************************************************************

        c1=1.0d0/6.0d0
              c2=1.0d0/12.0d0
c**************************************************************
c These following steps are to take boundary conditions into account
c                 For x
c***************************************************************

        if (nx.eq.1) then
           do s=1 , nx
              upx(s) = s+1
              downx(s) = s-1
           enddo
              dO j=1,ny
                 do i=1,nx
                    pxi0(downx(i),j)=0.0d0
                    pxi0(upx(i),j)=0.0d0
                    pxi(downx(i),j)=0.0d0
                    pxi(upx(i),j)=0.0d0
                    map1(downx(i),j)=0.0d0
                    map1(upx(i),j)=0.0d0
                 enddo
              enddo

        c1=1.0d0/6.0d0
        c2=1.0d0/12.0d0

        else
           do s=1 , nx
              upx (s) = s+1
              downx (s) = s-1
           end do
           if(bcx.eq.0) then
```

```
              upx (nx) = 1
              downx (1) = nx
           else
              upx (nx) =nx
              downx (1) = 1
           end if
        end if
c**************************************************************
c                 FOR y
c**************************************************************
        if (ny.eq.1) then
           do s=1 , ny
              upy(s) = s+1
              downy(s) =s-1
           enddo
               do j=1,ny
                 do i=1,nx
                    pxi0(i,downy(j))=0.0d0
                    pxi0(i,upy(j))=0.0d0
                    pxi(i,downy(j))=0.0d0
                    pxi(i,upy(j))=0.0d0
                    map1(i,downy(j))=0.0d0
                    map1(i,upy(j))=0.0d0
                 enddo
              enddo

           c1=1.0d0/6.0d0
           c2=1.0d0/12.0d0

        else
           do s=1 , ny
              upy (s) = s+1
              downy (s) = s-1
           enddo
           if(bcy.eq.0) then
              upy (ny) = 1
              downy (1) = ny
           else
              upy (ny) = ny
              downy (1) = 1
           endif
        endif

c *******************************************************************
c       Now it starts to run time (t) evolution
c *******************************************************************

       do t = ktime, time-1,1


c*****************************************************************
c       This following step is to calculate First Laplacian
c        APxi1 = [<< Pxi >> - Pxi]
c*****************************************************************

              do j=1,ny
                 do i=1,nx
                    pxi0(i,j)=pxi(i,j)
                    aapxi1(i,j)=c1*(pxi(upx(i),j)
     1              + pxi(downx(i),j)
     1              + pxi(i,upy(j))+pxi(i,downy(j)))
```

```
                  bapxi1(i,j)=c2*(pxi(downx(i),upy(j))
     1            +pxi(downx(i),downy(j))
   1       +pxi(upx(i),upy(j))+pxi(upx(i),downy(j)))

                  apxi1(i,j) = aapxi1(i,j) +bapxi1(i,j)

               enddo
            enddo

c*****************************************************************
c       This following step is to calculate map function
c*****************************************************************

            do j=1,ny
               do i=1,nx
                  f(i,j) = (tau-a*((1-2*r)**2))*pxi(i,j)
     1            -v*(1-2*r)*(pxi(i,j)**2)-u*(pxi(i,j)**3)
               enddo
            enddo
c*****************************************************************
c         This following step is to calculate
c         Map = {f(Pxi) + D[<<Pxi>> - Pxi] - Pxi}
c*****************************************************************

             do j=1,ny
               do i=1,nx
                   map1(i,j) = f(i,j) + d*(apxi1(i,j)
     1             - pxi(i,j))
               enddo
            enddo
c*****************************************************************
c The following is to calculate outer Laplacian on map function
c*****************************************************************

        do j=1,ny
            do i=1,nx
               aapxi2(i,j)=c1*(map1(upx(i),j)
     1         +map1(downx(i),j)
     1         +map1(i,upy(j))+map1(i,downy(j)))

               bapxi2(i,j)=c2*(map1(downx(i),upy(j))
     1         +map1(downx(i),downy(j))
     1         +map1(upx(i),upy(j))+map1(upx(i),downy(j)))

               apxi2(i,j)=aapxi2(i,j)
     1         +bapxi2(i,j)+capxi2(i,j)
           enddo
       enddo
c******************************************************************
c This following step is to calculate whole equation for Pxi(t+1,n)
c******************************************************************

            do j=1,ny
               do i=1,nx
                  pxi(i,j) = pxi0(i,j)+deltat*
     1            ( - b * pxi(i,j)  +map1(i,j) - APxi2(i,j))
               enddo
            enddo
```

```
c**********************************************************
c    writing Pxi values in files for time steps
c**********************************************************
           esse =(t+1)*deltat
           esse2=esse
           name3=char(esse2)
           write(name3,'(I7.7)') esse2
           name4=name1//'_pxi.'//name2//'_t'//name3//'.txt'
           if( esse .lt. 100.D0 ) then
              if( mod(t+1, int(10.D0/deltat)) .eq. 0 ) then
                  ma=ma+1
                  open(ma,file=name4)
                  write(ma,*) "#Grid", nx,ny
                  do i = 1, Nx
                     do j = 1, Ny
                        write (ma,*) pxi(i,j)
                        enddo
                     enddo
              close(ma)
              endif
           endif
           if( esse .lt. 1000.D0 ) then
              if( mod(t+1, int(100.D0/deltat)) .eq. 0 ) then
                   ma=ma+1
                   open(ma,file=name4)
                   write(ma,*) "#Grid", nx,ny
                   do i = 1, nx
                      do j = 1, ny
                         write (ma,*) pxi(i,j)
                      enddo
                   enddo
                   close(ma)
              endif
            elseif( esse .lt. 10000.D0 ) then
              if( mod(t+1, int(1000.D0/deltat)) .eq. 0 )then
                 ma=ma+1
                 open(ma,file=name4)
                 write(ma,*) "#Grid", nx,ny
                 do i = 1, nx
                    do j = 1, ny
                       write (ma,*) pxi(i,j)
                    enddo
                 enddo
                 close(ma)
              endif
           elseif( esse .le. 1000000.D0) then
              if( mod(t+1, int(10000.D0/deltat)) .eq. 0) then
                   ma=ma+1
                   open(ma,file=name4)
                   write(MA,*) "#Grid", Nx,Ny
                   do i = 1, Nx
                      do j = 1, Ny
                         write (Ma,*) pxi(i,j)
                       enddo
                   enddo
              close(ma)
              endif
           else
              if(mod(t+1,int(100000.D0/deltat)) .eq. 0)  then
                   ma=ma+1
                   open(ma,file=name4)
                  write(MA,*) "#Grid", Nx,Ny
                  do i = 1, nx
                     do j = 1, ny
                        write (ma,*) pxi(i,j)
                     enddo
                  enddo
               close(ma)
               endif
            endif
            If ( esse .le. 10000000.D0) then
               if( MOD(t+1,int(1000000.D0/deltat)) .eq. 0)  then
                  ma=ma+1
                  open(ma,file=name4)
                  if ((nz.eq.1)) then
                     write(ma,*) "#Grid", nx,ny
                     do i = 1, nx
                        do j = 1, ny
                           write (ma,*) pxi(i,j)
                        enddo
                     enddo
                   close(ma)
                  endif
               endif
               If(ma.gt.80) then
                  ma=0
               endif
            endif
         enddo

      end
c*******************End of source Code***********
```

Appendix B

CDS Code for 9–point Star Laplacain Scheme

```
c
c *****************************************************************
c    Here is presented the segment of the code from Appendix A. Only
c    the Laplacian scheme simulation part is changed according 9 –
c    point star scaling.
c *****************************************************************


c******************************************************************
c       This following step is to calculate First Laplacian
c        APxi1 = [<< Pxi >> - Pxi]
c******************************************************************

              do j=1,ny
                 do i=1,nx
                    pxi0(i,j)=pxi(i,j)
                    aapxi1(i,j)=c1*(pxi(upx(i),j)
     1              + pxi(downx(i),j) + pxi(i,upy(j))+pxi(i,downy(j))

                    aaapxi1(i,j)=c2*(pxi(upx(i+1),j)
     1              + pxi(downx(i-1),j)
     1              + pxi(i,upy(j+1))+pxi(i,downy(j-1))

                    apxi1(i,j)=aapxi1(i,j)+aaapxi1(i,j)

c    ***************************************************************

                 enddo
              enddo

c******************************************************************
c The following is to calculate outer Laplacian on map function
c******************************************************************

              do j=1,ny
                 do i=1,nx

                    aapxi2(i,j)=c1*(map1(upx(i),j)
     1              +map1(downx(i),j)
     1              +map1(i,upy(j))+map1(i,downy(j))


                    aaapxi2(i,j)=c2*(map1(upx(i+1),j)
     1              +map1(downx(i+1),j)
     1              +map1(i,upy(j+1))+map1(i,downy(j+1))

                    apxi2(i,j)=aapxi2(i,j)+aaapxi2(i,j)

                 enddo
              enddo
```

Appendix C

CDS Code for 17–point Laplacian Scheme

```
c
c *****************************************************************
c    Here is presented the segment of the code from Appendix A. Only
c    the Laplacian scheme simulation part is changed according 17 –
c    point scaling.
c *****************************************************************


c******************************************************************
c       This following step is to calculate First Laplacian
c        APxi1 = [<< Pxi >> - Pxi]
c******************************************************************

               do j=1,ny
                 do i=1,nx
                    pxi0(i,j)=pxi(i,j)
                    aapxi1(i,j)=c1*(pxi(upx(i),j
    1               + pxi(downx(i),j)
    1              + pxi(i,upy(j))+pxi(i,downy(j))

c    ***********************************************************

                    bapxi1(i,j)=c2*(pxi(downx(i),upy(j))
     1              +pxi(downx(i),downy(j))
     1              +pxi(upx(i),upy(j))+pxi(upx(i),downy(j))


c    **************************************************************

                    aaapxi1(i,j)=c3*(pxi(upx(i+1),j)
     1              + pxi(downx(i-1),j)
     1              + pxi(i,upy(j+1))+pxi(i,downy(j-1))


c    *********************************************************

                    bbapxi1(i,j)=c4*(pxi(downx(i-1),upy(j+1))
     1              +pxi(downx(i-1),downy(j-1))
     1              +pxi(upx(i+1),upy(j+1))+pxi(upx(i+1),downy(j-1))


c    ***************************************************************
                    apxi1(i,j)=aapxi1(i,j)+bapxi1(i,j)
     1              +aaapxi1(i,j)+bbapxi1(i,j)
c    ***************************************************************

                 enddo
              enddo



c******************************************************************
c The following is to calculate outer Laplacian on map function
c******************************************************************
```

```
          do j=1,ny
               do i=1,nx

                  aapxi2(i,j)=c1*(map1(upx(i),j)
   1              +map1(downx(i),j)
   1              +map1(i,upy(j))+map1(i,downy(j))

                  bapxi2(i,j)=c2*(map1(downx(i),upy(j))
   1              +map1(downx(i),downy(j))
   1              +map1(upx(i),upy(j))+map1(upx(i),downy(j))

                  aaapxi2(i,j)=c3*(map1(upx(i+1),j)
   1              +map1(downx(i-1),j)
   1              +map1(i,upy(j+1))+map1(i,downy(j-1))

                  bbapxi2(i,j)=c4*(map1(downx(i-1),upy(j+1))
   1              +map1(downx(i-1),downy(j-1))
   1              +map1(upx(i+1),upy(j+1))
  1       +map1(upx(i+1),downy(j-1))

                  apxi2(i,j)=aapxi2(i,j)++bapxi2(i,j)
   1              +aaapxi2(i,j)+bbapxi2(i,j)

               enddo
            enddo
```

Appendix D

3D CDS Code (Default)

```
‘The Input File for the CDS Fortran Program. It is general all
‘methodologies for 2D.

'New simulation (0) or continue a previous one (1)'
0
'Input filename containing starting atomic configuration (max 80 c)
for 1'
restartfile.dat
'Insert D'
0.5d0
'Insert A'
1.5d0
'Insert B'
0.01d0
'Insert f'
0.40d0
'Insert Tau'
0.20d0
'Insert v'
2.3d0
'Insert u'
0.38d0
'Insert Grid size'
75,75,50
'Insert the deltat'
1.d0
'Total TimeSteps'
1000000
'Save order parameter configuration for restarting every ... steps'
1000
'Write pos-neg order parameter in the following file'
'final.bak'
'Saving pos-neg order parameter in the following steps(max 10;5 for
line):'
200,700,1000,3000,5000
10000,20000,50000,70000,100000
'Insert file record positive'
'final2.bak'
'Input the first name (you must input 8 characters)'
'cdsnew01'
'Input the second name (you must input 2 characters tau only 28)'
20
```

```
c ****************************************************************
c     CDS simulation for Laplacian scheme
c     1/80+3/80+6/80
c ****************************************************************

        program order parameter
        implicit none
        double precision pxi(0:400,0:400,0:2)
        double precision pxi0(0:400,0:400,0:2)
        double precision zxi(0:400,0:400,0:2)
        double precision apxi1(0:400,0:400,0:2)
        double precision aapxi1(0:400,0:400,0:2)
        double precision bapxi1(0:400,0:400,0:2)
        double precision capxi1(0:400,0:400,0:2)
        double precision apxi2(0:400,0:400,0:2)
        double precision aapxi2(0:400,0:400,0:2)
        double precision bapxi2(0:400,0:400,0:2)
        double precision capxi2(0:400,0:400,0:2)
        double precision f(0:400,0:400,0:2)
        double precision map1(0:400,0:400,0:2)
        double precision mxi1(0:400,0:400,0:2)
        double precision z(0:400)
        double precision h1,h2,h3
        double precision hx(0:100)
        double precision hy(0:100)
        double precision hz(0:100)
        double precision tau,v,u,omega,omega0
        double precision a,b,d,r,e,sh,e0,sh0,c1,c2,c3
c ****************************************************************
c -------------Boudary conditions -------------------
c ****************************************************************
         integer upx(0:400),upy(0:400),upz(0:2)
        integer downx(0:400),downy(0:400),downz(0:2)
        double precision m(0:400)
        integer m1(0:400)
c ****************************************************************
c ---------Parameters for the CD simualtion------------------------
c ****************************************************************
        integer i,j,k,s,nx,ny,nz,ex,ey,ez,seed,t,time,conf,ktime
        integer s1,s2,s3,s4,s5,s6,s7,s8,s9,s10
        double precision mx,my
        double precision nxx,nyy,nzz,r1,rr
        integer everyconf,ftime,bcx,bcy,bcz,ht,R2
        character*80 label
        character*80 finalposition,fileconf,writeconf
c*****************************************************************
c ----------- Parameter to record data(order parameter)------------
c*****************************************************************
        real delapse,dtime,t1(2)
        character*8 name1
        character*2 name2
        character*7 name3
        character*29 name4
        integer ma,esse2
c ****************************************************************
c       ******************************************************
c       open(unit=9,file"cds.in",status='old',form='formatted')
        open(unit=98,file = 'cds.in', status='old',form='formatted')
c       ******************************************************
```

```
c
c       Read input data from file CDS.IN
        read(98,*) label
        read(98,*) conf
        read(98,*) label
        read(98,*) fileconf
        read(98,*) label
        read(98,*) d
        read(98,*) label
        read(98,*) a
        read(98,*) label
        read(98,*) b
        read(98,*) label
        read(98,*) r
        read(98,*) label
        read(98,*) tau
        read(98,*) label
        read(98,*) v
        read(98,*) label
        read(98,*) u
        read(98,*) label
        read(98,*) nx,ny,nz
        read(98,*) label
        read(98,*) deltat
        read(98,*) label
        read(98,*) time
        read(98,*) label
        read(98,*) everyconf
        read(98,*) label
        read(98,*) writeconf
        read(98,*) label
        read(98,*) s1,s2,s3,s4,s5
        read(98,*) s6,s7,s8,s9,s10
        read(98,*) label
        read(98,*) finalposition
        read(98,*) label
        read(98,*) name1
        read(98,*) label
        read(98,*) r2

        open(90, file = finalposition)

c *****************************************************************
c     These following steps are to create randomly initial values
c     order parameter Pxi, these values are: +0.3 or -0.3.
c *****************************************************************


        flag=0
        if (conf.eq.1) then
        open (77,file=fileconf,status='old')
        do i=1,nx
           do j=1,ny
              do k=1,nz
                 read(77,*) pxi(i,j,k)
              enddo
           enddo
        enddo
c        ktime=0
        else
        ktime=0
        do i = 1, nx
           do j =1, ny
              do k= 1, nz
                 call random_number(temp)
                 if (temp.ge.0 .and. temp.lt.0.3) then
                   Pxi(i,j,k) = 0.3d0
                 else
                   Pxi(i,j,k) = -0.3d0
                 endif
              enddo
           enddo
        enddo
        endif

c**********************************************************************
c Constant numbers and Laplacian weights
c**********************************************************************

        c1=1.0d0/6.0d0
        c2=1.0d0/12.0d0
        c3=0.0d0
        ma=0
        name2=char(r2)
        write(name2,'(i2.2)') r2

c ********************************************************************
c These following steps are to take boundary conditions into account
c  *******************************************************************
c                       For x
c ********************************************************************
        if (nx.eq.1) then
           do s=1 , nx
              upx(s) = s+1
              downx(s) = s-1
           enddo
           dO k=1,nz
              dO j=1,ny
                 dO i=1,ny
                    pxi0(downx(i),j,k)=0.0d0
                    pxi0(upx(i),j,k)=0.0d0
                    pxi(downx(i),j,k)=0.0d0
                    pxi(upx(i),j,k)=0.0d0
                    map1(downx(i),j,k)=0.0d0
                    map1(upx(i),j,k)=0.0d0
                 enddo
              enddo
           enddo
           c1=1.0d0/6.0d0
           c2=1.0d0/12.0d0
           c3=0.0d0
        else
           do s=1 , nx
              upx (s) = s+1
              downx (s) = s-1
              hx(s)=0.0d0
           end do
           if(bcx.eq.0) then
              upx (nx) = 1
              downx (1) = nx
           else
              upx (nx) =nx
```

```
            downx (1) = 1
            hx(nx)= h1
            hx(1)=h1
         end if
      end if
c  ****************************************************
c  *********** FOR y *********************************
c  ****************************************************
      if (ny.eq.1) then
         do s=1 , ny
            upy(s) = s+1
            downy(s) =s-1
         enddo
         do k=1,nz
            do j=1,ny
               do i=1,ny
                  pxi0(i,downy(j),k)=0.0d0
                  pxi0(i,upy(j),k)=0.0d0
                  pxi(i,downy(j),k)=0.0d0
                  pxi(i,upy(j),k)=0.0d0
                  map1(i,downy(j),k)=0.0d0
                  map1(i,upy(j),k)=0.0d0
               enddo
            enddo
         enddo
         c1=1.0d0/6.0d0
         c2=1.0d0/12.0d0
         c3=0.0d0
      else
         do s=1 , ny
            upy (s) = s+1
            downy (s) = s-1
            hy(s)=0.0d0
         enddo
         if(bcy.eq.0) then
            upy (ny) = 1
            downy (1) = ny
         else
            upy (ny) = ny
            downy (1) = 1
            hy(ny)= h2
            hy(1)=h2
         endif
      endif
c  ****************************************************
c  *********** FOR z *********************************
c  ****************************************************
      if (nz.eq.1) then
         do s=1 , nz
            upz(s) = s+1
            downz(s) =s-1
         enddo
         do k=1,nz
            do j=1,ny
               do i=1,ny
                  pxi0(i,j,upz(k))=0.0d0
                  pxi0(i,j,downz(k))=0.0d0
                  pxi(i,j,upz(k))=0.0d0
                  pxi(i,j,downz(k))=0.0d0
                  map1(i,j,upz(k))=0.0d0
                  map1(i,j,downz(k))=0.0d0
```

```
                 enddo
              enddo
           enddo
           c1=1.0d0/6.0d0
           c2=1.0d0/12.0d0
           c3=0.0d0
        else
           do s=1 , nz
              upz (s) = s+1
              downz (s) = s-1
              hz(s)=0.0d0
           end do
           if(bcz.eq.0) then
              upz (nz) = 1
              downz (1) = nz
           else
              upz (nz) = nz
              downz (1) = 1
              hz(nz)= h3
              hz(1)=h3
           end if
        endif
c
c ******************************************************************
        delapse=dtime(t1)
c ******************************************************************
c
        if (sh.ne.0.0d0) then
           do i=1,nx
              z(i)=0.0d0
           enddo
        endif
c ******************************************************************
c       Now it starts to run time (t) evolution
c ******************************************************************
c
        do t = ktime, time-1,1

c ******************************************************************
c       APxi1 = [<< Pxi >> - Pxi]
c ******************************************************************

           do k=1,nz
              do j=1,ny
                 do i=1,nx
                    pxi0(i,j,k)=pxi(i,j,k)
                    aapxi1(i,j,k)=c1*(pxi(upx(i),j,k)
     1              + pxi(downx(i),j,k)
     1              + pxi(i,upy(j),k)+pxi(i,downy(j),k)
     1              + pxi(i,j,upz(k))+pxi(i,j,downz(k)))
c    *********************************************************
                    bapxi1(i,j,k)=c2*(pxi(downx(i),upy(j),k)
     1              +pxi(downx(i),downy(j),k)
     1              +pxi(upx(i),upy(j),k)+pxi(upx(i),downy(j),k)
     1              +pxi(i,downy(j),upz(k))+pxi(i,downy(j),downz(k))
     1              +pxi(i,upy(j),upz(k))+pxi(i,upy(j),downz(k))
     1              +pxi(downx(i),j,upz(k))+pxi(downx(i),j,downz(k))
     1              +pxi(upx(i),j,upz(k))+pxi(upx(i),j,downz(k)))
c    ******************************************************************
                    capxi1(i,j,k)=c3*(pxi(downx(i),downy(j),upz(k))
     1              +pxi(downx(i),upy(j),upz(k))
     1               +pxi(downx(i),downy(j),downz(k))
     1               +pxi(downx(i),upy(j),downz(k))
     1               +pxi(upx(i),downy(j),upz(k))
     1               +pxi(upx(i),upy(j),upz(k))
     1               +pxi(upx(i),downy(j),downz(k))
     1               +pxi(upx(i),upy(j),downz(k)))
c    ****************************************************************
                     apxi1(i,j,k)=aapxi1(i,j,k)
     1               +bapxi1(i,j,k)+capxi1(i,j,k)
c    ****************************************************************
                  enddo
               enddo
            enddo
c        This following step is to calculate Map function:
            do k=1,nz
               do j=1,ny
                  do i=1,nx
                     f(i,j,k) = (tau-a*((1-2*r)**2))*pxi(i,j,k)
     1               -v*(1-2*r)*(pxi(i,j,k)**2)-u*(pxi(i,j,k)**3)
                  enddo
               enddo
            enddo
c        This following step is to calculate:
c        Map = {Atanh(Pxi) + D[<<Pxi>> - Pxi] - Pxi}
            do k=1,nz
               do j=1,ny
                  do i=1,nx
                      map1(i,j,k) = f(i,j,k) + d*(apxi1(i,j,k)
     1                - pxi(i,j,k))
                  enddo
               enddo
            enddo
c      This following step is to take into account the boundary
conditions
c        for Map(i,j) which will be needed for next step.
c
            do k=1,nz
               do j=1,ny
                  do i=1,nx

                     aapxi2(i,j,k)=c1*(map1(upx(i),j,k)
     1               +map1(downx(i),j,k)
     1               +map1(i,upy(j),k)+map1(i,downy(j),k)
     1               +map1(i,j,upz(k))+map1(i,j,downz(k)))

                     bapxi2(i,j,k)=c2*(map1(downx(i),upy(j),k)
     1               +map1(downx(i),downy(j),k)
     1               +map1(upx(i),upy(j),k)+map1(upx(i),downy(j),k)
     1               +map1(i,downy(j),upz(k))+map1(i,downy(j),downz(k))
     1               +map1(i,upy(j),upz(k))+map1(i,upy(j),downz(k))
     1               +map1(downx(i),j,upz(k))+map1(downx(i),j,downz(k))
     1               +map1(upx(i),j,upz(k))+map1(upx(i),j,downz(k)))

                     capxi2(i,j,k)=c3*(map1(downx(i),downy(j),upz(k))
     1               +map1(downx(i),upy(j),upz(k))
     1               +map1(downx(i),downy(j),downz(k))
     1               +map1(downx(i),upy(j),downz(k))
     1               +map1(upx(i),downy(j),upz(k))
     1               +map1(upx(i),upy(j),upz(k))
     1               +map1(upx(i),downy(j),downz(k))
     1               +map1(upx(i),upy(j),downz(k)))
```

```
                  apxi2(i,j,k)=aapxi2(i,j,k)
     1            +bapxi2(i,j,k)+capxi2(i,j,k)
               enddo
            enddo
         enddo
c *******************************************************************
c This following step is to calculate whole equation for Pxi(t+1,n)
c ********************************************************************
         do k=1,nz
            do j=1,ny
               do i=1,nx
                  pxi(i,j,k) = pxi(i,j,k)+deltat*
      1            (- b * pxi(i,j,k)
      1             - apxi2(i,j,k) +map1(i,j,k))
               enddo
            enddo
         enddo


c  *******************************************************
c  *****Save the configuration every tot step*************
c  *******************************************************
         ftime=((t+1)*deltat)+ktime
         ht=(t+1)*deltat
         esse =(t+1)*deltat
         if((ht).eq.esse) then
            if  ((ht).eq.(everyconf* (ht/everyconf)))then
               open (unit=96,file=writeconf)
               write (96,*) ftime , nx,ny,nz,deltat
               write (96,*) d,a,b,r,tau,v,u,e,sh,omega
               do i=1,nx
                  do j=1,ny
                     do k=1,nz
                        write (96,*) pxi(i,j,k)
                     enddo
                  enddo
               enddo
            close (96)
            endif
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         esse =(t+1)*deltat
         esse2=esse
         name3=char(esse2)
         write(name3,'(I7.7)') esse2
         name4=name1//'_pxi.'//name2//'_t'//name3//'.txt'
         if( esse .lt. 100.D0 ) then
            if( mod(t+1, int(10.D0/deltat)) .eq. 0 ) then
                ma=ma+1
                open(ma,file=name4)
                write(ma,*) "#Grid", nx,ny,nz
                do i = 1, Nx
                   do j = 1, Ny
                      do k = 1 , Nz
                         write (ma,*) pxi(i,j,k)
                      enddo
                   enddo
                enddo
            close(ma)
            endif
```

```
endif
if( esse .lt. 1000.D0 ) then
   if( mod(t+1, int(100.D0/deltat)) .eq. 0 ) then
        ma=ma+1
        open(ma,file=name4)
        write(ma,*) "#Grid", nx,ny,nz
        do i = 1, nx
           do j = 1, ny
              do k = 1 , nz
                 write (ma,*) pxi(i,j,k)
              enddo
           enddo
        enddo
        close(ma)
   endif
elseif( esse .lt. 10000.D0 ) then
   if( mod(t+1, int(1000.D0/deltat)) .eq. 0 )then
      ma=ma+1
      open(ma,file=name4)
      write(ma,*) "#Grid", nx,ny,nz
      do i = 1, nx
         do j = 1, ny
            do k = 1 , nz
               write (ma,*) pxi(i,j,k)
            enddo
         enddo
      enddo
      close(ma)
   endif
elseif( esse .le. 1000000.D0) then
   if( mod(t+1, int(10000.D0/deltat)) .eq. 0) then
        ma=ma+1
        open(ma,file=name4)
        write(MA,*) "#Grid", Nx,Ny,Nz
        do i = 1, Nx
           do j = 1, Ny
              do k = 1 , Nz
                 write (Ma,*) pxi(i,j,k)
              enddo
           enddo
        enddo
   close(ma)
   endif
else
   if(mod(t+1,int(100000.D0/deltat)) .eq. 0)  then
        ma=ma+1
        open(ma,file=name4)
        write(MA,*) "#Grid", Nx,Ny,Nz
        do i = 1, nx
           do j = 1, ny
              do k = 1 , nz
                 write (ma,*) pxi(i,j,k)
              enddo
           enddo
        enddo
   close(ma)
   endif
endif
If ( esse .le. 10000000.D0) then
   if( MOD(t+1,int(1000000.D0/deltat)) .eq. 0)  then
      ma=ma+1
```

```
                 open(ma,file=name4)
                 if ((nz.eq.1)) then
                    write(ma,*) "#Grid", nx,ny,nz
                    do i = 1, nx
                       do j = 1, ny
                          do k = 1 , nz
                             write (ma,*) pxi(i,j,k)
                          enddo
                       enddo
                    enddo
                   close(ma)
                 endif
              endif
              If(ma.gt.80) then
                 ma=0
              endif
           endif
        enddo
c       ******************************************
c       ******************************************
        delapse=dtime(t1)
        write(95,*) delapse,t1(1),t1(2)
c       ******************************************
c       ******************************************
        close(90)
        close(97)
        close(95)
        stop
        end
c *************end of program ********************
```

Appendix E

Conjugate Gradient Method

Basic algorithm for a non–preconditioned CG technique works as follows:

Given a linear system $Ax = b$, the first step with guess $x^0 = (0,0,0)^T$ and $P = P^{-1} = I$, where *I* is the identity or unit matrix and *P* the preconditioning matrix. Following is given CG algorithm [109]:

Step 0 – Preliminaries

$$\begin{aligned} &A \to a_{j,k}, 1 \le j,k \le N \\ &b \to b_k, 1 \le k \le N \\ &P \to \gamma_{j,k}, 1 \le j,k \le N \\ &x \to x_j, 1 \le j \le N \end{aligned} \tag{1}$$

Step 1 – Setup Step

$$\begin{aligned} &r^0 = b - Ax^0 = b \\ &w = P^{-1} r^0 = b \\ &v^1 = P^{-t} w = b \\ &\alpha = \langle w, w \rangle = \sum_{j=1}^{N} w_j^2 \end{aligned} \tag{2}$$

Step 2 – Start iteration with *k* = 1 until $k \le N$

Step 3 – Check if *v* < *Tolerance* ?: Solution x_n, residual r_n found. Stop.

$$\begin{aligned}
&u = Av^1 \\
&t_1 = \frac{\alpha}{\langle v^1, u\rangle} = (t_k = \sum_{k=1}^{N} \frac{\alpha}{v_k u_k}) \\
&x^1 = x^0 + t_1 v^1 \\
&r^1 = r^0 - t_1 u \\
&w = P^{-1} r^1 (= r^1) \\
&\beta = \langle w, w\rangle = (\beta = \sum_{k=1}^{N} w_j^2)
\end{aligned} \tag{3}$$

Step 4 – Check if $|\beta| < Tolerance$?: If $\|r\| < Tolerance$?: Solution x_n, residual r_n found. Stop.

$$\begin{aligned}
&s_1 = \frac{\beta}{\alpha} \\
&v^2 = P^{-t} w + s_1 v^1 \\
&\alpha = \beta
\end{aligned} \tag{4}$$

Step 5 – Check if ($k < N$) ?: Maximal numbers of iterations exceeded. Stop.

Step 6 – Restart iteration with $k = 2$, etc.

$$\begin{aligned}
&u = Av^2 \\
&\cdots = \cdots \\
&v^3 = \cdots \\
&\alpha = \beta
\end{aligned} \tag{5}$$

The preconditioning matrix $P \approx L$, where L comes from the Cholesky factorization LL^T of A. it follows that $P^{-1}P^T \approx A^{-1}$ is good approximation [76].

Appendix F

CDS CN Code

The Input File for the CDS Fortran Program. It is general all methodologies for 2D.

```
'New simulation (0) or continue a previous one (1)'
0
'Input filename containing starting atomic configuration (max 80 c)
for 1'
restartfile.dat
'Insert D'
0.7d0
'Insert A'
1.5d0
'Insert B'
0.02d0
'Insert f'
0.48d0
'Insert Tau'
0.36d0
'Insert v'
2.3d0
'Insert u'
0.38d0
'Insert Grid size'
128,128,1
'Insert the deltat'
1.d0
'Total TimeSteps'
1000000
'Save order parameter configuration for restarting every ... steps'
1000
'Write pos-neg order parameter in the following file'
'final.bak'
'Saving pos-neg order parameter in the following steps(max 10;5 for
line):'
200,700,1000,3000,5000
10000,20000,50000,70000,100000
'Insert file record positive'
'final2.bak'
'Input the first name (you must input 8 characters)'
'cdsnew01'
'Input the second name (you must input 2 characters tau only 28)'
20
```

The CDS CN Fortran Program

```
c
c ****************************************************************
c CDS 2D simulation source code (Fortran code for CN Method)
c ****************************************************************

c ***********************************************
c     Module section for diagonal Matrices
c ***********************************************
         module routines
             contains
                subroutine diag(NN,MM,d,deltat,nx)
                integer nx,snx,tnx,n,hnx,i,j,nstp,h,u
                double precision  d, deltat,c1,c2
                real MM1(0:200,0:200)
                real MM2(0:200,0:200)
                real NN1(0:200,0:200)
                real NN2(0:200,0:200)
                real MA(0:200,0:200)
                real MB(0:200,0:200)
                real MC(0:200,0:200)
                real MD(0:200,0:200)
                real A(0:200,0:200)
                real B(0:200,0:200)
                real C(0:200,0:200)
                real, dimension(:,:), allocatable,intent(inout)::MM
                real, dimension (:,:), allocatable ,intent(inout)::NN
                nstp = nx*nx
                allocate(MM(0:nstp+1,0:nstp+1))
                allocate(NN(0:nstp+1,0:nstp+1))
                write(*,*) "First"

c***********************************************
c     Laplacian weights
c***********************************************
                c1 = 1.d0/6.d0
                c2 = 1.d0/12.d0
c***********************************************
                do i=1,nx
                    MM1(i,i) = -1.d0
                    if (i<nx) then
                           MM1(i,i+1) = c1
                           MM1(i+1,i) = c1
                    end if
                    if (i .eq. nx) then
                       MM1(1,i) = c1
                       MM1(i,1) = c1
                    end if
                enddo

                do i=1,nx
                   MM2(i,i) = c1
                   if (i<nx) then
                        MM2(i,i+1) = c2
                        MM2(i+1,i) = c2
                    end if
                    if (i .eq. nx) then
                       MM2(1,i) = c2
                       MM2(i,1) = c2
                    end if
```

```
  enddo

   snx = nx
   tnx = nx*nx

  do i=1,snx
      do j=1,snx
           NN(i,j) = MM1(i,j)
           NN(i,j+snx) = MM2(i,j)
           NN((tnx-snx)+i,(tnx-snx)+j) = MM1(i,j)
           NN((tnx-snx)+i,(tnx-snx*2)+j) = MM2(i,j)
      end do
  end do
  do n = 1,nx-2
     snx = n*nx
    do i = snx+1, nx+snx
       hnx=1
       do j = (snx - nx) + 1,snx
              NN(i,j) = MM2(i-snx,hnx)
              NN(i,j+nx) = MM1(i-snx,hnx)
              NN(i,j+2*nx) = MM2(i-snx,hnx)
              hnx = hnx+1
        enddo
 enddo
enddo
snx = nx
tnx = nx*nx
do i=1,snx
   do j=1,snx
      NN(i,(tnx-snx)+j) = MM2(i,j)
      NN((tnx-snx)+i,j) = MM2(i,j)
   end do
 end do

 MA(1:nx,1:nx)=MATMUL(MM1(1:nx,1:nx),d*MM1(1:nx,1:nx))
 MB(1:nx,1:nx)=MATMUL(MM2(1:nx,1:nx),d*MM2(1:nx,1:nx))
 MC(1:nx,1:nx)=MATMUL(MM1(1:nx,1:nx),d*MM2(1:nx,1:nx))
 MD(1:nx,1:nx)=MATMUL(MM2(1:nx,1:nx),d*MM1(1:nx,1:nx))
 A(1:nx,1:nx)= MA(1:nx,1:nx)+MB(1:nx,1:nx)+MB(1:nx,1:nx)
 B(1:nx,1:nx) = MC(1:nx,1:nx)+MD(1:nx,1:nx)
 C(1:nx,1:nx) = MB(1:nx,1:nx)

 do n=1,nx
     snx = n*nx
     u=1
     do i=snx-nx+1,snx
        hnx=1
        do j=(snx-nx)+1,snx
           MM(i,j) = A(u,hnx)
           hnx=hnx+1
        end do
        u = u+1
      end do
  end do

 do n=2,nx
     snx = n*nx
     u=1
     do i=snx-nx+1,snx
        hnx=1
        do j=(snx-2*nx)+1,snx-nx
```

```
                      MM(i,j) = B(u,hnx)
                      MM(i-nx,j+nx) = B(u,hnx)
                      hnx=hnx+1
                   end do
                   u=u+1
                 end do
             end do
            do n=3,nx
               snx = n*nx
               u=1
               do i=snx-nx+1,snx
                  hnx = 1
                  do j=(snx-3*nx)+1,snx-2*nx
                     MM(i,j) = C(u,hnx)
                     MM(i-2*nx,j+2*nx) = C(u,hnx)
                     hnx = hnx+1
                  end do
                  u=u+1
               end do
            end do
          snx = nx
          tnx = nx*nx
          do i=1,snx
             do j=1,snx
                MM(i,(tnx-snx)+j) = B(i,j)
                MM((tnx-snx)+i,j) = B(i,j)
                MM(i,(tnx-2*snx)+j) = C(i,j)
                MM((tnx-2*snx)+i,j) = C(i,j)
                MM(i+nx,(tnx-snx)+j) = C(i,j)
                MM((tnx-snx)+i,j+nx) = C(i,j)
             end do
          end do

        end subroutine diag
c *******************************************
c     Conjugate Gradient Method
c **************** *************************
       subroutine conjgradx(ML,uu,x,tol,n,nx)
       integer n,i,j,k,e,snx,p,nx
       real alpha,beta,h,t,s,tol,sum1
       real A(0:4100,0:4100)
       real, dimension(:,:), allocatable :: ML
       real, dimension(:), allocatable :: uu
       real, dimension(:), allocatable :: x
       real, dimension(:,:), allocatable:: C
       real, dimension(:), allocatable:: r
       real, dimension(:), allocatable :: w
       real, dimension(:), allocatable :: v
       real, dimension(:), allocatable ::u
       real, dimension(:), allocatable :: l

       allocate(C(0:n+1,0:n+1))
       allocate(r(0:n+1))
       allocate(w(0:n+1))
       allocate(v(0:n+1))
       allocate(u(0:n+1))
       allocate(l(0:n+1))
       C = 0.d0
       do i=1,n
        C(i,i) = 1.d0
       enddo
```

```
          do i=1,n
             do j=1,n
                r(i) = uu(i)- ML(i,j)*x(j)
             end do
          end do

          if (NORM2(r) < tol) then
              return
          end if

         w(1:n) = MATMUL(C(1:n,1:n),r(1:n))
         v(1:n) = MATMUL(C(1:n,1:n),w(1:n))

         alpha = DOT_PRODUCT(w(1:n),w(1:n))

         do k=1,n
            u(1:n) = MATMUL(ML(1:n,1:n),v(1:n))

            t = alpha/DOT_PRODUCT(v(1:n),u(1:n))

            x(1:n) = x(1:n) + t*v(1:n)
            r(1:n) = r(1:n) - t*u(1:n)

             if (NORM2(r) < tol) then
                 return
              end if
            w(1:n) = MATMUL(C(1:n,1:n),r(1:n))
            beta =0.d0
            beta = DOT_PRODUCT(w(1:n),w(1:n))
            s= beta/alpha
            v(1:n) = MATMUL(C(1:n,1:n),w(1:n))
     1                    +s*v(1:n)

          alpha = beta
       enddo
       deallocate(C,w,v,u,l,r)
       end subroutine conjgradx
c*********************************************************
      end module routines
c*********************************************************



c*********************************************************
c     Main Program
c*********************************************************

        program order_parameter
        use routines
        implicit none
        real, dimension (:,:), allocatable ::pxi
        real, dimension (:,:), allocatable ::apxi1
        real, dimension (:,:), allocatable ::f
        real, dimension (:,:), allocatable :: pxin
        real, dimension (:,:), allocatable ::map1
        double precision tau,v,u,tmp,tt
        double precision a,b,d,r,e,sh,e0,sh0,c1,c2,c3
        real, dimension (:,:), allocatable,save ::MM
        real, dimension (:,:), allocatable ::Id
        real, dimension (:,:), allocatable,save ::NN
        real, dimension (:,:), allocatable ::ML
        real, dimension (:,:), allocatable ::MR
        real, dimension (:), allocatable ::uu
        real, dimension (:), allocatable ::uu1
        real, dimension (:), allocatable ::x
        double precision L,alpha,beta,cond
        integer nstep,p,n,hnx,it,mx,tnx,snx,nt

        double precision eta,pi2,temp,deltat,deltat0,esse,cost,rand
c ****************************************************************
c ---------Parameters for the CD simualtion-----------------------
c ****************************************************************
        integer i,j,k,s,nx,ny,nz,ex,ey,ez,seed,t,time,conf,ktime
        integer s1,s2,s3,s4,s5,s6,s7,s8,s9,s10
        integer everyconf,ftime,bcx,bcy,bcz,ht,R2
        real tol
        character*80 label
        character*80 finalposition,fileconf,writeconf
c*****************************************************************
c ----------- Parameter to record data(order parameter)------------
c*****************************************************************
        real delapse,dtime,t1(2)
        character*8 name1
        character*2 name2
        character*7 name3
        character*29 name4
        integer ma,esse2
c ****************************************************************
c*****************************************************************
c                File open
c*****************************************************************

      open(unit=98,file = 'cds.in', status='old',form='formatted')

c***************************************************
c     Read input data from file CDS.IN
c***************************************************

        read(98,*) label
        read(98,*) conf
        read(98,*) label
        read(98,*) fileconf
        read(98,*) label
        read(98,*) d
        read(98,*) label
        read(98,*) a
        read(98,*) label
        read(98,*) b
        read(98,*) label
        read(98,*) r
        read(98,*) label
        read(98,*) tau
        read(98,*) label
        read(98,*) v
        read(98,*) label
        read(98,*) u
        read(98,*) label
        read(98,*) nx,ny,nz
        read(98,*) label
        read(98,*) bcx,bcy,bcz
        read(98,*) label
        read(98,*) deltat
        read(98,*) label
        read(98,*) time
        read(98,*) label
        read(98,*) everyconf
        read(98,*) label
        read(98,*) writeconf
        read(98,*) label
        read(98,*) s1,s2,s3,s4,s5
        read(98,*) s6,s7,s8,s9,s10
        read(98,*) label
        read(98,*) finalposition
        read(98,*) label
        read(98,*) name1
        read(98,*) label
        read(98,*) r2

        data tol /1.0e-10/
        nstep = nx*ny
        allocate (pxi(0:nx+1,0:ny+1))
        allocate (apxi1(0:nx+1,0:ny+1))
        allocate (pxin(0:nx+1,0:ny+1))
        allocate (map1(0:nx+1,0:ny+1))
        allocate (f(0:nx+1,0:ny+1))
        allocate (Id(0:nstep+1,0:nstep+1))
        allocate (ML(0:nstep+1,0:nstep+1))
        allocate (MR(0:nstep+1,0:nstep+1))
        allocate (uu(0:nstep+1))
        allocate (uu1(0:nstep+1))
        allocate (x(0:nstep+1))
        ktime=0
        do i = 1, nx
           do j =1, ny
                 call random_number(temp)
                 if (temp.ge.0 .and. temp.lt.0.3) then
                   pxi(i,j) = 0.3d0
                 else
                   pxi(i,j) = -0.3d0
                 endif
           enddo
        enddo



c*******************************************************************
c            Costant numbers
c*******************************************************************

        pi2=2.d0*dacos(-1.d0)
        c1=1.0d0/4.0d0
        nstep = nx*nx
        ma=0
        name2=char(r2)
        write(name2,'(i2.2)') r2
c*******************************************************************

        call diag(NN,MM,d,deltat,nx)

        Id = 0.d0
        do i=1,nstep
```

```
              Id(i,i) = 1.d0
         end do
         MR(1:nstep,1:nstep) = Id(1:nstep,1:nstep)
     1    -((b*deltat)/2.d0)*Id(1:nstep,1:nstep)
     1    - (deltat/2.0)* MM(1:nstep,1:nstep)

         ML(1:nstep,1:nstep) = Id(1:nstep,1:nstep)
     1    +((b*deltat)/2.d0)*Id(1:nstep,1:nstep)
     1    + (deltat/2.0)* MM(1:nstep,1:nstep)


c******************************************************************
c       Now we start to run time (t) evolution
c *****************************************************************
c
        do t = ktime,time-1,1

c******************************************************************
c       This following step is to calculate
C       First Laplacian (Forward Euler Method)
c        APxi1 = [<< Pxi >> - Pxi]
c******************************************************************
           uu=0.d0
            snx = 1
              do i=1,nx
                 do j=1,nx
                    uu(snx) = pxi(i,j)
                    snx = snx+1
                 enddo
              enddo
          uu1 = 0.d0
          uu1(1:nstep) = MATMUL(MR(1:nstep,1:nstep),uu(1:nstep))

          snx = 1
             do i=1,nx
                 do j=1,nx
                    apxi1(i,j) =uu1(snx)
                    snx = snx+1
                 enddo
              enddo
c******************************************************************
c  This following step is to calculate map function
c  (Forward Euler method)
c******************************************************************

          do j=1,ny
              do i=1,nx
                   f(i,j) = (tau-a*((1-2*r)**2))*pxi(i,j)
     1             -v*(1-2*r)*(pxi(i,j)**2)-u*(pxi(i,j)**3)
                 enddo
              enddo
           uu=0.d0
           uu1 = 0.d0
           snx = 1
           do i=1,nx
               do j=1,nx
                  uu(snx) = f(i,j)
                  snx = snx+1
               enddo
           enddo
           uu1(1:nstep) = MATMUL(NN(1:nstep,1:nstep),uu(1:nstep))
           snx = 1
           do i=1,nx
             do j=1,nx
                f(i,j) = (deltat/2.d0)*uu1(snx)
                snx = snx+1
             end do
          end do

             do i=1,nx
                do j=1,ny
                    pxi(i,j) = apxi1(i,j)-f(i,j)
                enddo
             enddo

c******************************************************************
c       This following step is to calculate First Laplacian
c       (Backward Euler Method)
c        Pxin = [<< Pxi >> - Pxi]
c******************************************************************

          x(1:nstep) = 0.d0
          uu=0.d0
            snx = 1
              do i=1,nx
                 do j=1,nx
                    uu(snx) = pxi(i,j)
                    snx = snx+1
                 enddo
              enddo
         call conjgradx(ML,uu,x,tol,nstep,nx)

          snx = 1
             do i=1,nx
                 do j=1,nx
                    pxin(i,j) =x(snx)
                    snx = snx+1
                 enddo
              enddo
c******************************************************************
c       This following step is to calculate map function
c          (Backward Euler method)
c******************************************************************


          do j=1,ny
               do i=1,nx
                    f(i,j) = (tau-a*((1-2*r)**2))*pxin(i,j)
     1             -v*(1-2*r)*(pxin(i,j)**2)-u*(pxin(i,j)**3)
                 enddo
              enddo
           uu=0.d0
           snx = 1
           do i=1,nx
               do j=1,nx
                  uu(snx) = f(i,j)
                  snx = snx+1
               enddo
           enddo
           uu1(1:nstep) = MATMUL(NN(1:nstep,1:nstep),uu(1:nstep))

           snx = 1
```

```
           do i=1,nx
             do j=1,nx
                f(i,j) = (deltat/2.d0)*uu1(snx)
                snx = snx+1
             end do
          end do

c******************************************************************
c    This following step is to calculate whole equation for Pxi(t+1,n)
c******************************************************************

             snx = 1
             do i=1,nx
                do j=1,ny
                    pxi(i,j) = pxin(i,j)-f(i,j)
                    snx = snx+1
                enddo
             enddo

c*****************************************************
c    writing Pxi values in files for time steps in files
c*****************************************************
           esse =(t+1)*deltat
           esse2=esse
           name3=char(esse2)
           write(name3,'(I7.7)') esse2
           name4=name1//'_pxi.'//name2//'_t'//name3//'.txt'
           if( esse .lt. 100.D0 ) then
              if( mod(t+1, int(10.D0/deltat)) .eq. 0 ) then
                  ma=ma+1
                  open(ma,file=name4)
                  write(ma,*) "#Grid", nx,ny
                  do i = 1, Nx
                     do j = 1, Ny
                        write (ma,*) pxi(i,j)
                        enddo
                     enddo
              close(ma)
              endif
           endif
           if( esse .lt. 1000.D0 ) then
              if( mod(t+1, int(100.D0/deltat)) .eq. 0 ) then
                   ma=ma+1
                   open(ma,file=name4)
                   write(ma,*) "#Grid", nx,ny
                   do i = 1, nx
                      do j = 1, ny
                         write (ma,*) pxi(i,j)
                      enddo
                   enddo
                   close(ma)
              endif
            elseif( esse .lt. 10000.D0 ) then
              if( mod(t+1, int(1000.D0/deltat)) .eq. 0 )then
                 ma=ma+1
                 open(ma,file=name4)
                 write(ma,*) "#Grid", nx,ny
                 do i = 1, nx
                    do j = 1, ny
                       write (ma,*) pxi(i,j)
                    enddo
                 enddo
                 close(ma)
              endif
           elseif( esse .le. 1000000.D0) then
              if( mod(t+1, int(10000.D0/deltat)) .eq. 0) then
                   ma=ma+1
                   open(ma,file=name4)
                   write(MA,*) "#Grid", Nx,Ny
                   do i = 1, Nx
                      do j = 1, Ny
                         write (Ma,*) pxi(i,j)
                       enddo
                   enddo
              close(ma)
              endif
           else
              if(mod(t+1,int(100000.D0/deltat)) .eq. 0)  then
                   ma=ma+1
                   open(ma,file=name4)
                   write(MA,*) "#Grid", Nx,Ny
                   do i = 1, nx
                      do j = 1, ny
                          write (ma,*) pxi(i,j)
                      enddo
                   enddo
              close(ma)
              endif
           endif
           If ( esse .le. 10000000.D0) then
              if( MOD(t+1,int(1000000.D0/deltat)) .eq. 0)  then
                 ma=ma+1
                 open(ma,file=name4)
                 if ((nz.eq.1)) then
                    write(ma,*) "#Grid", nx,ny
                    do i = 1, nx
                       do j = 1, ny
                          write (ma,*) pxi(i,j)
                       enddo
                    enddo
                  close(ma)
                 endif
              endif
              If(ma.gt.80) then
                 ma=0
              endif
           endif
         enddo

        end
c************************End of program*********************
```

Appendix G

Two Order Parameter Systems CDS Code

```
'New simulation (0) or continue a previous one (1)'
0
'Input filename containing starting atomic configuration (max 80 c)
for 1'
'restartfile.dat'
'Inserire D'
0.5d0
'Inserire A'
1.5d0
'Inserire B'
0.02d0
'Inserire f'
0.48d0
'Inserire Tau'
0.30d0
'Inserire v'
1.5d0
'Inserire u'
0.5d0
'Box dimension: 0 for 2D and 1 for 3D'
0
'Inseririre N mesh'
128,128,1
'Insert the size of the noise'
0.0d0
'Insert the electric field'
0.003d0
'Insert the amplitude of the shear'
0.0d0
'Insert the omega of the shear'
0.0d0
'Insert the neutral wall (0,0,0)=(x,y,z) ex:1,0,0'
0,0,0
'Insert the deltat'
0.5d0
'Total TimeSteps'
2050000
'Save order parameter every step...'
10,100,1000,10000,100000,1000000
'Save order parameter configuration for restarting every ... steps'
100
'Write pos-neg order parameter in the following file'
'final.bak'
'Saving pos-neg order parameter in the following steps(max 10;5 for
line):'
200,700,1000,3000,5000
10000,20000,50000,70000,100000
'Inserire file record positive'
'final2.bak'
'Input the first name (you must input 8 characters)'
'newcds01'
'cc1, Ohta (2.2)'
1.0d0
'cc2, Ohta (2.2)'
0.5d0
'b1, Ohta (2.3), (2.4a), (2.4b)'
```

```
0.07d0
'b2, Ohta (2.3)'
0.2d0
'b3, Ohta (2.3)'
0.0d0
'b4, Ohta (2.3)'
0.0d0
'alpha, Ohta (2.6a)'
0.02d0
'beta, Ohta (2.6b)'
0.0d0
'A1 for tanh, Ohta (p 52 III)'
1.3d0
'A2 for tanh, Ohta (p 52 III)'
1.1d0
'pxi_c, eta, critical point, input positive value instead of negative'
0.0d0
'Average phi'
0.0d0
'Average pxi - not needed, set pxi_c instead'
-0.2d0
```

```
c ***********************************************************
c Here the Two order parameter source code is presented
c which was used for both the 2D and 3D simulations. Only the
c changes for either dimension can be made in the IN file given
c above for this system.
c ***********************************************************
       program order_parameter
        implicit none

        real, dimension (:,:,:), allocatable :: pxi
        real, dimension (:,:,:), allocatable :: pxi0
        real, dimension (:,:,:), allocatable :: zxi, zxi2
        real, dimension (:,:,:), allocatable :: apxi1
        real, dimension (:,:,:), allocatable :: aapxi1
        real, dimension (:,:,:), allocatable :: bapxi1
        real, dimension (:,:,:), allocatable :: capxi1
        real, dimension (:,:,:), allocatable :: apxi2
        real, dimension (:,:,:), allocatable :: aapxi2
        real, dimension (:,:,:), allocatable :: bapxi2
        real, dimension (:,:,:), allocatable :: capxi2
        real, dimension (:,:,:), allocatable :: f
        real, dimension (:,:,:), allocatable :: ff
        real, dimension (:,:,:), allocatable :: map1
        real, dimension (:,:,:), allocatable :: mxi1, mxi2
        real, dimension (:), allocatable :: z
        double precision tau,v,u,omega,omega0
        double precision a,b,d,r,e,sh,e0,sh0,c1,c2,c3
c ******************************************************************
c --------------Random generator for the noise---------------------
c ******************************************************************
        double precision caso1,caso2,caso3,caso4,caso5,caso6,caso7
        double precision caso8,caso9
        real, dimension (:,:,:), allocatable :: csi1,csi2,csi3
        double precision sumvx,sumvy,sumvz,vmax1,vmax2,vmax3
c ******************************************************************
c -------------Boudary condiction and shear flow-------------------
c ******************************************************************
        double precision eta,pi2,temp,deltat,deltat0,esse,cost,rand
        integer, dimension (:), allocatable :: upx, upy, upz
        integer, dimension (:), allocatable :: downx, downy, downz
        integer, dimension (:), allocatable :: m, m1
c ******************************************************************
c ----------INTEGER for random generator for the noise-------------
c ******************************************************************
        real, dimension (:,:,:), allocatable :: a1, a2, a3
        integer flag
c ******************************************************************
c ---------Parameters for the CD simualtion------------------------
c ******************************************************************
        integer i,j,k,s,nx,ny,nz,ex,ey,ez,seed,t,time,conf,ktime
        integer s1,s2,s3,s4,s5,s6,s7,s8,s9,s10
        integer f1,f2,f3,f4,f5,f6
        integer everyconf,ftime,bcx,bcy,bcz,ht,r2
        character*80 label
        character*80 finalposition,fileconf,writeconf
c*******************************************************************
c ----------- Parameter to record data(order parameter)------------
c*******************************************************************
```

```
                integer restart
        character*80 restartfile1
        character*80 restartfile2

        real delapse,dtime,t1(2)
        character*8 name1
        character*7 name3
        character*29 name4
        character*29 name5

        integer ma, ma2, esse2, simdim

       real, dimension (:,:,:), allocatable :: phi
       real, dimension (:,:,:), allocatable :: aphi1
       real, dimension (:,:,:), allocatable :: aaphi1
       real, dimension (:,:,:), allocatable :: baphi1
       real, dimension (:,:,:), allocatable :: caphi1
       real, dimension (:,:,:), allocatable :: map2
       real, dimension (:,:,:), allocatable :: aphi2
       real, dimension (:,:,:), allocatable :: aaphi2
       real, dimension (:,:,:), allocatable :: baphi2
       real, dimension (:,:,:), allocatable :: caphi2


c       cc1 will be cc1 / 2 (save computing time)
c       cc2 will be cc2 / 2 (save computing time)

        double precision cc1, cc2, bb1, bb2, bb3, bb4, aa1, aa2
        double precision alpha, beta, gamma, psic, phim, pxim


c ****************************************************************
        open(unit=98,file = 'cds.in',status='old',form='formatted')
c ***************************************************************
c ***************************************************************
c       Read input data from file CDS.IN
c ***************************************************************


        read(98,*) label
        read(98,*) conf

        read(98,*) label
        read(98,*) fileconf

        read(98,*) label
        read(98,*) d

        read(98,*) label
        read(98,*) a

        read(98,*) label
        read(98,*) b

        read(98,*) label
        read(98,*) r
        read(98,*) label
        read(98,*) tau

        read(98,*) label
        read(98,*) v

        read(98,*) label
        read(98,*) u

        read(98,*) label
        read(98,*) simdim

        read(98,*) label
        read(98,*) nx,ny,nz

        read(98,*) label
        read(98,*) eta

        read(98,*) label
        read(98,*) e

        read(98,*) label
        read(98,*) sh

        read(98,*) label
        read(98,*) omega

        read(98,*) label
        read(98,*) bcx,bcy,bcz

        read(98,*) label
        read(98,*) deltat

        read(98,*) label
        read(98,*) time

        read(98,*) label
        read(98,*) f1,f2,f3,f4,f5,f6

        read(98,*) label
        read(98,*) everyconf

        read(98,*) label
        read(98,*) writeconf

        read(98,*) label
        read(98,*) s1,s2,s3,s4,s5
        read(98,*) s6,s7,s8,s9,s10

        read(98,*) label
        read(98,*) finalposition

        read(98,*) label
        read(98,*) name1

        read(98,*) label
        read(98,*) cc1
```

```
        read(98,*) label
        read(98,*) cc2

        read(98,*) label
        read(98,*) bb1

        read(98,*) label
        read(98,*) bb2

        read(98,*) label
        read(98,*) bb3

        read(98,*) label
        read(98,*) bb4

        read(98,*) label
        read(98,*) alpha

        read(98,*) label
        read(98,*) beta

        read(98,*) label
        read(98,*) aa1

        read(98,*) label
        read(98,*) aa2

        read(98,*) label
        read(98,*) psic

        read(98,*) label
        read(98,*) phim

        read(98,*) label
        read(98,*) pxim

        open(90, file = finalposition)

c ****************************************************************
c Allocate Arrays
c ****************************************************************

        allocate (pxi(0:nx+1,0:ny+1,0:nz+1))
        allocate (zxi(0:nx+1,0:ny+1,0:nz+1))
        allocate (zxi2(0:nx+1,0:ny+1,0:nz+1))
        allocate (apxi1(0:nx+1,0:ny+1,0:nz+1))
        allocate (aapxi1(0:nx+1,0:ny+1,0:nz+1))
        allocate (bapxi1(0:nx+1,0:ny+1,0:nz+1))
        allocate (capxi1(0:nx+1,0:ny+1,0:nz+1))
        allocate (apxi2(0:nx+1,0:ny+1,0:nz+1))
        allocate (aapxi2(0:nx+1,0:ny+1,0:nz+1))
        allocate (bapxi2(0:nx+1,0:ny+1,0:nz+1))
        allocate (capxi2(0:nx+1,0:ny+1,0:nz+1))

        allocate (phi(0:nx+1,0:ny+1,0:nz+1))
        allocate (aphi1(0:nx+1,0:ny+1,0:nz+1))
        allocate (aaphi1(0:nx+1,0:ny+1,0:nz+1))
        allocate (baphi1(0:nx+1,0:ny+1,0:nz+1))
        allocate (caphi1(0:nx+1,0:ny+1,0:nz+1))
        allocate (aphi2(0:nx+1,0:ny+1,0:nz+1))
        allocate (aaphi2(0:nx+1,0:ny+1,0:nz+1))
        allocate (baphi2(0:nx+1,0:ny+1,0:nz+1))
        allocate (caphi2(0:nx+1,0:ny+1,0:nz+1))

        allocate (f(0:nx+1,0:ny+1,0:nz+1))
        allocate (ff(0:nx+1,0:ny+1,0:nz+1))
        allocate (map1(0:nx+1,0:ny+1,0:nz+1))
        allocate (map2(0:nx+1,0:ny+1,0:nz+1))
        allocate (mxi1(0:nx+1,0:ny+1,0:nz+1))
        allocate (mxi2(0:nx+1,0:ny+1,0:nz+1))
        allocate (csi1(0:nx+1,0:ny+1,0:nz+1))
        allocate (csi2(0:nx+1,0:ny+1,0:nz+1))
        allocate (csi3(0:nx+1,0:ny+1,0:nz+1))
        allocate (a1(0:nx+1,0:ny+1,0:nz+1))
        allocate (a2(0:nx+1,0:ny+1,0:nz+1))
        allocate (a3(0:nx+1,0:ny+1,0:nz+1))

        allocate (upx(nx))
        allocate (upy(ny))
        allocate (upz(nz))
        allocate (downx(nx))
        allocate (downy(ny))
        allocate (downz(nz))

        allocate (m(nx))
        allocate (z(nx))
        allocate (m1(nx))
c ****************************************************************
c   These following steps are to create randomly initial values
c ****************************************************************

        flag=0
        if (conf.eq.1) then
           open (77,file=fileconf,status='old')
           do i=1,nx
              do j=1,ny
                 do k=1,nz
                    read(77,*) pxi(i,j,k)
                 enddo
              enddo
           enddo
        else
           ktime=0
           do i = 1, nx
              do j =1, ny
                 do k= 1,  nz
                    call random_number(temp)
                    if (temp.GE.0.0d0 .and. temp.LT.0.5d0) then
                       pxi(i,j,k) = -0.19d0
                    else
                       pxi(i,j,k) = -0.21d0
                    endif
                    call random_number(temp)
                    if (temp.ge.0.0d0 .and. temp .lt. 0.5d0) then
                       phi(i,j,k) = 0.01d0
                    else
```

```
                     phi(i,j,k) = -0.01d0
                  endif

               enddo
            enddo
         enddo
      endif

c
c *****************************************************************
c -----------------Costant numbers---------------------------------
c *****************************************************************
      if (simdim.eq.1) then

c For 3D
          c1 = 6.0d0/80.d0
          c2 = 3.0d0/80.0d0
          c3 = 1.0d0/80.0d0
      else
         if (simdim.eq.0) then
c For 2D
         c1 = 1.0d0/6.0d0
         c2 = 1.0d0/12.0d0
         c3 = 0.0d0
         endif
      endif

c For control unit file I/O
       ma= 0
       ma2 = 40
c ****************************************************************
c These following steps are to take boundary conditions into
c    account
c ****************************************************************
c *************For x **********************************************
c *****************************************************************
        If (nx.eq.1) then
           do s=1 , nx
              upx(s) = s+1
              downx(s) = s-1
           enddo
           do k=1,nz
              do j=1,ny
                 do i=1,nx
                    pxi(downx(i),j,k)=0.0d0
                    pxi(upx(i),j,k)=0.0d0
                    map1(downx(i),j,k)=0.0d0
                    map1(upx(i),j,k)=0.0d0

                    phi(downx(i),j,k)=0.0d0
                    phi(upx(i),j,k)=0.0d0
                    map2(downx(i),j,k)=0.0d0
                    map2(upx(i),j,k) = 0.0d0

                  enddo
              enddo
           enddo
           if (simdim.eq.1) then
```

```
c For 3D
               c1 = 6.0d0/80.d0
               c2 = 3.0d0/80.0d0
               c3 = 1.0d0/80.0d0
           else
               if (simdim.eq.0) then
c For 2D
                  c1 = 1.0d0/6.0d0
                  c2 = 1.0d0/12.0d0
                  c3 = 0.0d0
               endif
            endif
        else
           do s=1,nx
              upx(s) = s+1
              downx(s) = s-1
           enddo
           if(bcx.eq.0) then
             upx(nx) = 1
             downx(1) = nx
           else
             upx(nx) =nx
             downx(1) = 1
           end if
        end if

c  ****************************************************
c  ************ FOR y *********************************
c  ****************************************************
        if (ny.eq.1) then
           do s=1 , ny
              upy(s) = s+1
              downy(s) =s-1
           enddo
           do k=1,nz
              do j=1,ny
                 do i=1,nx
                    pxi(i,downy(j),k)=0.0d0
                    pxi(i,upy(j),k)=0.0d0
                    map1(i,downy(j),k)=0.0d0
                    map1(i,upy(j),k)=0.0d0

                   phi(i,downy(j),k)=0.0d0
                   phi(i,upy(j),k)=0.0d0
                   map2(i,downy(j),k)=0.0d0
                   map2(i,upy(j),k) = 0.0d0

                 enddo
              enddo
           enddo
           if (simdim.eq.1) then

c For 3D
               c1 = 6.0d0/80.d0
               c2 = 3.0d0/80.0d0
               c3 = 1.0d0/80.0d0
           else
```

```
               if (simdim.eq.0) then
c For 2D
                  c1 = 1.0d0/6.0d0
                  c2 = 1.0d0/12.0d0
                  c3 = 0.0d0
               endif
            endif
        else
           do s=1 , ny
               upy(s) = s+1
               downy(s) = s-1
           enddo
           if(bcy.eq.0) then
                upy(ny) = 1
                downy(1) = ny
           else
                upy(ny) = ny
                downy(1) = 1
           endif
        endif
c  ****************************************************
c  ************ FOR z *********************************
c  ****************************************************
        if (nz.eq.1) then
           do s=1 , nz
              upz(s) = s+1
              downz(s) =s-1
           enddo
           do k=1,nz
              do j=1,ny
                 do i=1,nx
                    pxi(i,j,upz(k))=0.0d0
                    pxi(i,j,downz(k))=0.0d0
                    map1(i,j,upz(k))=0.0d0
                    map1(i,j,downz(k))=0.0d0

                    phi(i,j,upz(k))=0.0d0
                    phi(i,j,downz(k))=0.0d0
                    map2(i,j,downz(k))=0.0d0
                    map2(i,j,upz(k)) = 0.0d0

                 enddo
              enddo
           enddo
           if (simdim.eq.1) then

c For 3D
               c1 = 6.0d0/80.d0
               c2 = 3.0d0/80.0d0
               c3 = 1.0d0/80.0d0
           else
               if (simdim.eq.0) then
c For 2D
                  c1 = 1.0d0/6.0d0
                  c2 = 1.0d0/12.0d0
                  c3 = 0.0d0
               endif
            endif
        else
           do s=1 , nz
              upz(s) = s+1
              downz(s) = s-1
           end do
           if(bcz.Eq.0) then
              upz(nz) = 1
              downz(1) = nz
           else
             upz(nz) = nz
             downz(1) = 1
           end if
        end if

c ****************************************************************
        delapse=dtime(t1)
c ****************************************************************

        if (sh.ne.0.0d0) then
             do i=1,nx
                z(i)=0.0d0
             enddo
        endif
c
******************************************************************
c       Now we start to run time (t) evolution
c
******************************************************************
        do t =ktime, time-1,1

       write(*,*) 'T',t

c
******************************************************************
c       APxi1 = [<< Pxi >> - Pxi]
c       ****************************************************
c
        do k=1,nz
           do j=1,ny
              do i=1,nx
                 aapxi1(i,j,k)=c1*(pxi(upx(i),j,k)
     1           + pxi(downx(i),j,k)
     1           + pxi(i,upy(j),k)+pxi(i,downy(j),k)
     1           + pxi(i,j,upz(k))+pxi(i,j,downz(k)))
c    ****************************************************
                 bapxi1(i,j,k)=c2*(pxi(downx(i),upy(j),k)
     1           +pxi(downx(i),downy(j),k)
     1           +pxi(upx(i),upy(j),k)+pxi(upx(i),downy(j),k)
     1           +pxi(i,downy(j),upz(k))+pxi(i,downy(j),downz(k))
     1           +pxi(i,upy(j),upz(k))+pxi(i,upy(j),downz(k))
     1           +pxi(downx(i),j,upz(k))+pxi(downx(i),j,downz(k))
     1           +pxi(upx(i),j,upz(k))+pxi(upx(i),j,downz(k)))
c    ******************************************************
                 capxi1(i,j,k)=c3*(pxi(downx(i),downy(j),upz(k))
     1           +pxi(downx(i),upy(j),upz(k))
     1           +pxi(downx(i),downy(j),downz(k))
     1           +pxi(downx(i),upy(j),downz(k))
     1           +pxi(upx(i),downy(j),upz(k))
     1           +pxi(upx(i),upy(j),upz(k))
     1           +pxi(upx(i),downy(j),downz(k))
     1           +pxi(upx(i),upy(j),downz(k)))
c ****************************************************************

apxi1(i,j,k)=aapxi1(i,j,k)+bapxi1(i,j,k)+capxi1(i,j,k)

c ****************************************************************
              enddo
           enddo
        enddo

        do k=1, nz
           do j=1, ny
              do i = 1,nx
                 aaphi1(i,j,k) = c1*(phi(upx(i),j,k)
     1           +phi(downx(i),j,k)
     1           +phi(i,upy(j),k) + phi(i,downy(j),k)
     1           +phi(i,j,upz(k)) + phi(i,j,downz(k)))

c *****************************************************************

                 baphi1(i,j,k) = c2*(phi(downx(i),upy(j),k)
     1           +phi(downx(i), downy(j),k)
     1           +phi(upx(i),upy(j),k)+phi(upx(i),downy(j),k)
     1           +phi(i,downy(j),upz(k))+phi(i,downy(j),downz(k))
     1           +phi(i,upy(j),upz(k))+phi(i,upy(j),downz(k))
     1           +phi(downx(i),j,upz(k))+phi(downx(i),j,downz(k))
     1           +phi(upx(i),j,upz(k))+phi(upx(i),j,downz(k)))
c ******************************************************************
                 capxi1(i,j,k)=c3*(phi(downx(i),downy(j),upz(k))
     1           +phi(downx(i),upy(j),upz(k))
     1           +phi(downx(i),downy(j),downz(k))
     1           +phi(downx(i),upy(j),downz(k))
     1           +phi(upx(i),downy(j),upz(k))
     1           +phi(upx(i),upy(j),upz(k))
     1           +phi(upx(i),downy(j),downz(k))
     1           +phi(upx(i),upy(j),downz(k)))
C ****************************************************************

aphi1(i,j,k)=aaphi1(i,j,k)+baphi1(i,j,k)+caphi1(i,j,k)

              enddo
           enddo
        enddo

c       This following step is to calculate

        do i=1,nx
           do j =1,ny
              do k =1,nz
                 map1(i,j,k) = -cc1*(apxi1(i,j,k)-pxi(i,j,k))
     1           -aa1*tanh(pxi(i,j,k))+pxi(i,j,k)
     1           +bb1*phi(i,j,k)
     1           -0.5d0*bb2*phi(i,j,k)**2
     1           +bb3*pxi(i,j,k)*(phi(i,j,k)**2)
```

```
                 enddo
              enddo
          enddo

          do i=1,nx
             do j=1,ny
                do k=1,nz
                   map2(i,j,k) =-cc2*(aphi1(i,j,k)-phi(i,j,k))
       1           -aa2*tanh(phi(i,j,k)) + phi(i,j,k)
       1           +bb1*pxi(i,j,k)
       1           -bb2*pxi(i,j,k)*phi(i,j,k)
       1           +bb3*phi(i,j,k)*(pxi(i,j,k)**2)
                enddo
              enddo
          enddo

c ***********************************************************
c  This following step is to take into account the boundary
c  conditions
c ***********************************************************

c*******for Map(i,j) which will be needed for next step.*******

          do k=1,nz
             do j=1,ny
                do i=1,nx
                   aapxi2(i,j,k)=c1*(map1(upx(i),j,k)
       1           +map1(downx(i),j,k)
       1           +map1(i,upy(j),k)+map1(i,downy(j),k)
       1           +map1(i,j,upz(k))+map1(i,j,downz(k)))

                   bapxi2(i,j,k)=c2*(map1(downx(i),upy(j),k)
       1           +map1(downx(i),downy(j),k)
       1           +map1(upx(i),upy(j),k)+map1(upx(i),downy(j),k)
       1           +map1(i,downy(j),upz(k))+map1(i,downy(j),downz(k))
       1           +map1(i,upy(j),upz(k))+map1(i,upy(j),downz(k))
       1           +map1(downx(i),j,upz(k))+map1(downx(i),j,downz(k))
       1           +map1(upx(i),j,upz(k))+map1(upx(i),j,downz(k)))

                   capxi2(i,j,k)=c3*(map1(downx(i),downy(j),upz(k))
       1           +map1(downx(i),upy(j),upz(k))
       1           +map1(downx(i),downy(j),downz(k))
       1           +map1(downx(i),upy(j),downz(k))
       1           +map1(upx(i),downy(j),upz(k))
       1           +map1(upx(i),upy(j),upz(k))
       1           +map1(upx(i),downy(j),downz(k))
       1           +map1(upx(i),upy(j),downz(k)))


apxi2(i,j,k)=aapxi2(i,j,k)+bapxi2(i,j,k)+capxi2(i,j,k)
                enddo
             enddo
          enddo

          do k=1,nz
             do j=1,ny
                do i=1,nx
                aaphi2(i,j,k) = c1*(map2(upx(i),j,k)
     1          +map2(downx(i),j,k)
     1          +map2(i,upy(j),k) + map2(i,downy(j),k)
     1          +map2(i,j,upz(k)) + map2(i,j,downz(k)))

                baphi2(i,j,k)=c2*(map2(downx(i),upy(j),k)
     1          +map2(downx(i),downy(j),k)
     1          +map2(upx(i),upy(j),k)+map2(upx(i),downy(j),k)
     1          +map2(i,downy(j),upz(k))+map2(i,downy(j),downz(k))
     1          +map2(i,upy(j),upz(k))+map2(i,upy(j),downz(k))
     1          +map2(downx(i),j,upz(k))+map2(downx(i),j,downz(k))
     1          +map2(upx(i),j,upz(k))+map2(upx(i),j,downz(k)))

                caphi2(i,j,k)=c3*(map2(downx(i),downy(j),upz(k))
     1          +map2(downx(i),upy(j),upz(k))
     1          +map2(downx(i),downy(j),downz(k))
     1          +map2(downx(i),upy(j),downz(k))
     1          +map2(upx(i),downy(j),upz(k))
     1          +map2(upx(i),upy(j),upz(k))
     1          +map2(upx(i),downy(j),downz(k))
     1          +map2(upx(i),upy(j),downz(k)))


aphi2(i,j,k)=aaphi2(i,j,k)+baphi2(i,j,k)+caphi2(i,j,k)
              enddo
           enddo
        enddo


        do k=1,nz
           do j=1,ny
              do i=1,nx
                 pxi(i,j,k) = pxi(i,j,k)+deltat*
     1           (-map1(i,j,k)+apxi2(i,j,k))
             enddo
           enddo
        enddo

        do k=1,nz
           do j=1,ny
              do i=1,nx
                 phi(i,j,k) = phi(i,j,k) + deltat*(-map2(i,j,k)
     1           +aphi2(i,j,k)
     1           -alpha*(phi(i,j,k)-phim))
              enddo
           enddo
        enddo


c*************************************************************
c Customized output writing output to files is in the same way
c as given in Appendix A
c*************************************************************
```