Multilevel Numerical Algorithms for Systems of Nonlinear Parabolic Partial Differential Equations

by

Mashael Atallah Aljohani

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy

UNIVERSITY OF LEEDS

The University of Leeds
School of Computing
April 2020

The candidate confirms that the work submitted is her own, except where work which has formed part of a jointly authored publication has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Acknowledgements

First and above all, I praise and thank you Allah (Almighty God) for giving me the patience, strength, knowledge and well-being, in all my life and also for blessing me to successfully complete my PhD thesis.

My deepest appreciation and gratitude go to my supervisor Professor Peter Jimack, the Executive Dean of the faculty of Engineering, and Dr Mark Walkley, for their patient and friendly guidance, motivation, encouragement and invaluable support throughout my studies. I am grateful to both for helping me with to realise the significance of analytical, logical and structured thinking, and to develop me in a scholarly intellect. I am forever indebted to them for generously giving their valuable time and knowledge, as well as for instilling great confidence in both myself and my research.

My heartfelt thanks go to my lovely parents, my father Atallah and mother Salma, your prayers guided all the success in my life. I would like to thank my Mom and my Dad for their prayers, support, confidence and love in all my life. Without your support, I probably would not have made this achievement. Special thanks to my brothers and my sisters, for each time I would struggle, your support and your advice would always stand to inspire me to continue in the right path in my life and my PhD journey.

Most importantly, I wish to thank my beloved husband, Fawzi. Words cannot express how thankful I am to you for all the sacrifices that you have made on my behalf. You have always supported me and motivated me no matter what the limit. I would like to thank you also for always helping and encouraging me to finish my studies.

I would especially like to thank my children, your smiles and your prayers always remove all my tiredness and stress. I would like to express my thanks to you for being such good kids, always praying for me, encouraging me, and providing unending inspiration. I am so proud of you all.

I owe great thanks to my country Saudi Arabia, and also, a special thanks belongs to Taibah University, Medina, for awarding me the scholarship to continue my studies in the United Kingdom.

A huge thank you to my wonderful friends in the School of Computing, all colleagues and staff, who made my PhD experience so pleasant.

Abstract

This thesis is concerned with the development of efficient and reliable numerical algorithms for the solution of nonlinear systems of partial differential equations (PDEs) of elliptic and parabolic type. The main focus is on the implementation and performance of three different nonlinear multilevel algorithms, following discretisation of the PDEs: the Full Approximation Scheme (FAS), Newton-Multigrid (Newton-MG) and a Newton-Krylov solver with a novel preconditioner that we have developed based on the use of Algebraic Multigrid (AMG). In recent years these algorithms have been commonly used to solve nonlinear systems that arise from the discretisation of PDEs due to the fact that their execution time can scale linearly (or close to linearly) with the number of degrees of freedom used in the discretisation.

We consider two mathematical models: a thin film flow and the Cahn-Hilliard-Hele-Shaw model. These mathematical models consist of nonlinear, time-dependent and coupled PDEs systems. Using a Finite Difference Method (FDM) in space and Backward Differentiation Formulae (BDF) in time, we discrete the two models, to produce nonlinear algebraic systems. We are able to solve these nonlinear systems implicitly in computationally demanding 2D situations. We present numerical results, for both steady-state and time-dependent problems, that demonstrate the optimality of the three numerical algorithms for the thin film flow model. We show optimality of the FAS and Newton-Krylov approaches for the time-dependent Cahn-Hilliard-Hele-Shaw (CHHS) problem.

The main contribution is to address the question of which of these three nonlinear solvers is likely to be the best (i.e. computationally most effective) in practice. In order to asses this, we discuss the careful implementation and timing of these algorithms in order to permit a fair direct comparison of their computational cost. We then present extensive numerical results in order to make this comparison between these nonlinear multilevel methods. The conclusion emerging from this investigation is that it does not appear that there is a single superior approach, but rather that the best approach is problem dependent. Specifically, we find that our optimally preconditioned Newton-Krylov approach is best for the thin film flow model in the steady-state and time-dependent form, whilst the FAS solver appears best for the time-dependent CHHS model.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Objective of the Thesis

The objective of this thesis is to investigate and implement efficient and accurate multilevel algorithms for solving nonlinear parabolic and elliptic systems of partial differential equations (PDEs). Nonlinear multilevel schemes are well established in the literature as fast solvers for nonlinear PDEs of elliptic and parabolic type [5, 24, 28, 122], however, there appears to have been little attempt to make a systematic investigation of the relative performance of such algorithms for nonlinear systems.

We obtain numerical solutions which are robust and efficient using nonlinear multilevel methods for solving the nonlinear algebraic systems of equations that arise from the finite difference discretisation and implicit time integration of particular PDE systems. Specifically, we considered the application of the FAS, Newton-MG and preconditioned Newton-Krylov algorithms for two specific nonlinear problems: a thin film flow system and the Cahn-Hilliard-Hele-Shaw (CHHS) system. We especially seek numerical algorithms that give optimal time complexity for these nonlinear problems, since the advantage of such approaches becomes critically important as the problem size increases.

We present extensive numerical results for the efficient and accurate implementations of the nonlinear multilevel FAS, Newton-MG and Newton-Krylov algorithms (with a newly proposed preconditioner) for solving the thin film flow model. We also present numerical results for the efficient and accurate application of the FAS and Newton-Krylov algorithms (with a newly proposed preconditioner) for solving the CHHS model. These demonstrate that numerical results are efficient in the sense that they are either optimal in time (i.e. linear time complexity) or

very close to optimal.

The main contributions that have been made in this thesis can be summarised as follows:

- Development and implementation of three multilevel algorithms for the thin film flow system of equations: a nonlinear FAS algorithm, a Newton-MG algorithm and a preconditioned Newton-Krylov algorithm.
- The development of a novel preconditioner for the Newton-Krylov algorithm applied to the thin film flow problem. This uses algebraic multigrid (AMG) as a component of a more complex approximate block factorization.
- A detailed comparison of the three nonlinear multilevel algorithms for solving the thin film flow problem in both steady-state and time-dependent cases. We observed in both cases that the Newton-Krylov algorithm with our new preconditioner is the most efficient solver.
- Implementation of two optimal multilevel algorithms, a nonlinear FAS and a preconditioned Newton-Krylov algorithm for a two-phase model, namely the Cahn-Hilliard-Hele-Shaw (CHHS) system. We develop a new preconditioner based on an approximate block factorization plus AMG approach, similar in principle to that used in the thin film flow problem.
- A comparison of, and contrast between, the nonlinear multigrid FAS and the preconditioned Newton-Krylov approach for solving the Cahn-Hilliard-Hele-Shaw model in 2D. For this nonlinear system, we observed that the FAS algorithm is the more efficient solver.

These contributions are described in detail and discussed in depth in this thesis. We close the introduction by outlining the structure of the thesis.

1.2 Outline of the Thesis

This thesis has seven chapters, the first of which is this introduction. The remainder of the thesis is organized as follows.

- Chapter 2 gives background and introduction to the application area of elliptic and parabolic PDEs, including the particular nonlinear mathematical models that we are considering as model problems in this research.
- In Chapter 3 a review of literature relevant for this thesis is given, including an overview of the computational methods used in the thesis, such as the spatial discretisation, and temporal discretisation. We discuss numerical methods such as the Finite Difference Method (FDM) and the family of Backward Differentiation Formulae (BDF) used to approximate

the model problems. We review the definition and basic properties of nonlinear algebraic equation systems and Differential-Algebraic Equation (DAE) systems.

– In Chapter 4 we introduce several of the most significant methods for solving linear and nonlinear systems of algebraic equations. These include iterative methods for linear systems such as Multigrid and Krylov subspace methods. Numerical algorithms for solutions of nonlinear systems are presented based on the Newton, Newton-Krylov and Multigrid algorithms. Furthermore, we discuss the three nonlinear multilevel algorithms which form the core of this thesis: the FAS multigrid algorithm, Newton-multigrid and the preconditioned Newton-Krylov algorithm.

– In Chapter 5 we examine the performance of our numerical implementations of the three nonlinear multilevel algorithms for solving the thin film flow model in 2D. We consider these implementations in both steady-state (elliptic) and time-dependent (parabolic) cases. Detailed numerical results and comparison between these algorithms are presented in this chapter as well.

– In Chapter 6 we present the numerical solution of the Cahn-Hilliard-Hele-Shaw (CHHS) system of equations using two nonlinear multilevel algorithms, the FAS and preconditioned Newton-Krylov schemes. We present detailed numerical results and make a relative comparison of the approaches.

– In Chapter 7 our conclusions and suggestions for future avenues of research are presented.

– In Appendices A and B, we discuss technical details associated with the nonzero entries in the Jacobian matrix for degrees of freedom next to Dirichlet boundary conditions for the thin film flow model and the CHHS model, respectively. The main body of the thesis only discusses Jacobian entries away from the Dirichlet boundary in order to aid clarity for the reader.

Chapter 2

Mathematical Models

In this chapter, we introduce the mathematical models and concepts that are related to the research in this thesis. In the first section of this chapter, we will present an introduction to the application area of elliptic and parabolic PDEs. Thereafter we discuss certain mathematical models which will subsequently be used to demonstrate and assess the algorithms developed later.

2.1 Introduction to the Application Area of Elliptic and Parabolic PDEs

PDEs are differential equations that include unknown multivariable functions and their associated partial derivatives. They arise in many areas of science and engineering which govern many natural phenomena. The functions sought have independent variables often representing time and spatial directions. In simple cases, analytical techniques are utilized whereas computer algorithms have to be employed for more intricate equations. Let us take a look at a two dimensional, second-order linear partial differential equation which can be written as:

$$a\,\frac{\partial^2 u}{\partial x^2} + b\,\frac{\partial^2 u}{\partial x \partial y} + c\,\frac{\partial^2 u}{\partial y^2} + d\,\frac{\partial u}{\partial x} + e\,\frac{\partial u}{\partial y} + f u + g = 0, \tag{2.1}$$

according to [87,104,105,121,125]. Here a, b, c, d, e, f, and g may be functions of the independent spatial variables x and y. However for nonlinear problems they may also be functions of the dependent variable u. We can classify PDEs into three broad categories which can be determined by the characteristic polynomial of the highest order derivatives in Equation (2.1). PDEs are *elliptic* when $b^2 - 4ac < 0$, *parabolic* when $b^2 - 4ac = 0$ and *hyperbolic* when $b^2 - 4ac > 0$. We now discuss common examples of the three different types. One of the most common hyperbolic

PDEs discussed according to [45, 87, 105, 108, 121] is the one-dimensional wave equation:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \, \frac{\partial^2 u}{\partial x^2}. \tag{2.2}$$

Note that, although this is termed "one-dimensional", there are still two independent variables. The reason for this is that in most applications t represents time and x represents the single space variable.

We see in [87, 105, 121] that the simplest example of an elliptic equation is Laplace's equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \tag{2.3}$$

which is generally associated with steady-state behavior in two space dimensions. Furthermore a simple example of a parabolic equation is the one-dimensional heat equation:

$$\frac{\partial u}{\partial t} = k \, \frac{\partial^2 u}{\partial x^2}. \tag{2.4}$$

where $u(t, x)$ signifies the temperature at time t and position x. Usually k is considered constant, however, it can depend both on x and t.

In three dimensions, the function $u(x, y, z, t)$ of three spatial variables (x, y, z) and the time variable t, satisfies the heat equation:

$$\frac{\partial u}{\partial t} - k \, \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) = 0. \tag{2.5}$$

Conventionally we can write Equation (2.5) as follows,

$$\frac{\partial u}{\partial t} - k \, \Delta u = 0, \tag{2.6}$$

where Δ is the Laplace operator for the spatial derivatives.

For a unique solution, we require consistent initial and boundary conditions. Thus we define a bounded region Ω to be the spatial domain while we define $(0, T]$ to be the temporal region. Now the full domain for this PDE is $\Omega \times (0, T]$ and we require the values of u or its normal derivative at all points to be defined on the boundary of Ω ($\partial\Omega$) and the value of u at $t = 0$ (initial data) at all spatial points. We explore two broad categories of boundary conditions for models in this research: Dirichlet and Neumann. An example, of a Dirichlet boundary condition takes the form:

$$u = g \;\; on \quad \partial\,\Omega \times (0, T], \tag{2.7}$$

where g is a known function on the boundary $\partial\,\Omega$ of the spatial region Ω. On the other hand, a Neumann boundary condition takes the form:

$$\frac{\partial u}{\partial \tilde{n}} = f \;\; on \quad \partial\,\Omega \times (0, T], \tag{2.8}$$

where f is a known function, and $\tilde{n}$ is the outward normal to $\partial\,\Omega$. Therefore a positive value of f indicates outflow across the boundary $\partial\,\Omega$. As indicated in [32] and [108], PDEs can have purely Neumann boundary conditions or Dirichlet boundary conditions or have a mixture of both. It is possible to define Dirichlet boundary conditions on part of the domain with Neumann boundary conditions for the rest.

We have only described linear elliptic, hyperbolic and parabolic PDEs in one, two and three dimensions at this point. The purpose of our work is to study efficient methods to solve nonlinear PDEs. Hence the remainder of this section is used to introduce some nonlinear PDEs. In particular, we restrict our attention to nonlinear parabolic and elliptic problems. The interested reader should consult [2,45,87,105,108,121] for discussions on nonlinear hyperbolic PDEs.

There are various forms of nonlinearity to consider. For instance, we can turn the linear Equation (2.4) into a nonlinear equation by defining the coefficient $c(u)$, which is a known nonlinear function, as follows:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(c(u)\;\frac{\partial u}{\partial x}\right). \tag{2.9}$$

The Equation (2.9) is termed the nonlinear diffusion equation (or nonlinear heat equation) in [2, 121], and is an example of a nonlinear parabolic PDE.

We can vary this nonlinear equation in several ways, for example by [72]:

– appending more spatial variables giving,

$$\tfrac{\partial u}{\partial t} \;\; = \;\; \tfrac{\partial}{\partial x}\;(c(u)\;\tfrac{\partial u}{\partial x}) + \tfrac{\partial}{\partial y}\;(c(u)\;\tfrac{\partial u}{\partial y}) + \tfrac{\partial}{\partial z}\;(c(u)\;\tfrac{\partial u}{\partial z}), \tag{2.10}$$

where $u = u(x, y, z, t)$,

– adding a nonlinear forcing term f such as

$$\tfrac{\partial u}{\partial t} \;\; = \;\; \tfrac{\partial}{\partial x}(c(u)\,\tfrac{\partial u}{\partial x}) + \tfrac{\partial}{\partial y}(c(u)\,\tfrac{\partial u}{\partial y}) + f(u, x, y), \tag{2.11}$$

where $u = u(x, y, t)$.

By introducing the divergence operator $\nabla.v$ which acts on an arbitrary vector function v, and

the gradient operator ∇v, the PDE Equation (2.10) can be expressed as:

$$\frac{\partial u}{\partial t} = \nabla . \left(c(u) \ \nabla u \right). \tag{2.12}$$

Now that we have introduced the basic definitions, it is possible to introduce specific applications where we can model the phenomena as a system of nonlinear PDEs. These models will be used to investigate several numerical algorithms later in the thesis. In particular, we have two models in mind. Firstly a thin film model for fully developed flows over a defined topography in one and two dimensions. The second of these models is used to investigate the mixing of fluids, namely a two-phase field model called the Cahn-Hilliard-Hele-Shaw (CHHS) model.

2.2 Thin Film Flow Model

Thin film flow approximations are used in a wide variety of applications from biomedical sciences (e.g. thin tear films in the eye or fluid linings in the lungs of animals, see [23,31,64,68]) through to engineering and physics (e.g. coating processes in manufacturing or modelling the behaviour of a raindrop along a window under the action of gravity, see [40, 53, 76, 103, 114, 115, 145]). Generally, thin film flows involve a liquid that is bounded between a solid substrate and a free surface with another fluid. In many applications, this can just be air. The distinguishing features of the applications being discussed is that the motion perpendicular to the substrate may be neglected.

2.2.1 Derivation

A mathematical model of fully-developed flow is illustrated in the following subsections in one and two-dimensions. We introduce briefly the derivation of the thin film flow model [99,115,145]. The motion of a general flow is governed by the time-dependent Navier-Stokes equations which are:

$$\rho \left(\frac{\partial \underline{\mathrm{U}}}{\partial T} + \underline{\mathrm{U}} . \nabla \underline{\mathrm{U}} \right) = \ -\nabla P + \mu \nabla^2 \underline{\mathrm{U}} + \rho \ \underline{\mathrm{g}}, \tag{2.13}$$

$$\nabla . \underline{\mathrm{U}} \ = \ 0. \tag{2.14}$$

Here we have assumed that the fluid is incompressible, of density ρ, and of viscosity μ. The fluid occupies a given domain in its initial condition, and we denote pressure as P and fluid velocity as $\underline{\mathrm{U}} = (U, V, W)$. For the flow of a fluid down a plane inclined at an angle α, the body force $\underline{\mathrm{g}} = g(sin\,\alpha, 0, -cos\,\alpha)$, where the acceleration due to gravity is $g = 9.81ms^{-2}$. Using characteristic lengths H_0 as the thin film flow thickness and L_0 as the extent of the substrate, to have a valid thin film approximation, the ratio $\epsilon = \frac{H_0}{L_0}$ is required to be sufficient small [51,53], and T_0 is the time which is proportional to that derived by Orchard [99] for the levelling of surface disturbances. Use of this assumption in order to simplify the Navier-Stokes equations (2.13) and (2.14) is known as the lubrication approximation or long-wave approximation. This

approach works in terms of the corresponding non-dimensional (lower case) variables (for more details see these references [51, 53, 99]).

$$h(x,y,t) = \frac{H(X,Y,T)}{H_0}, \;\; s(x,y) = \frac{S(X,Y)}{H_0}, \;\; (x,y) = \frac{(X,Y)}{L_0}, \;\; z = \frac{Z}{H_0},$$

$$p(x,y) = \frac{2P(X,Y)}{\rho\, g\, L_0\, sin\alpha}, \;\; t = \frac{T}{T_0}, \;\; T_0 = \frac{\mu L_0}{\sigma \epsilon^3}, \;\; (u,v,\frac{w}{\epsilon}) = (U,V,W)\frac{T_0}{L_0}. \tag{2.15}$$

Here $h(x,y,t)$ and $p(x,y,t)$ are the thin film flow thickness and the pressure variable respectively and $s(x,y)$ is the bed shape topography of the substrate. The fluid in Figure 2.1 is supposed to be Newtonian and incompressible, of fixed density ρ and viscosity μ, with surface tension σ. Furthermore, (x,y,z) are non-dimensional Cartesian spatial coordinates, (u,v,w) is the non-dimensional fluid velocity, t denotes time non-dimensionalised against a time scale T_0 which is obtained by Orchard [99]. Further simplifications associated with thin film flow for the governing Equation (2.13) assume the convection $\rho\,(\underline{U}.\nabla\underline{U})$ to be negligible. Applying this assumption to Equation (2.13), reduces to the Stokes equations and further simplifications, in which high order terms in ϵ are neglected, leads to the following system of equations for the fluid thickness, $h(x,y,t)$ and the pressure field, p. We do not describe the derivation of this pressure here, however, the outcome is to yield the thin film equation as follows:

$$\frac{\partial h}{\partial t} = \frac{\partial}{\partial x}\left[\frac{h^3}{3}(\frac{\partial p}{\partial x} - 2)\right] + \frac{\partial}{\partial y}\left[\frac{h^3}{3}(\frac{\partial p}{\partial y})\right]. \tag{2.16}$$

During this fully-developed flow, the pressure is expressed as follows,

$$p = -\frac{\epsilon^3}{Ca}\triangle(h+s) + 2\epsilon\,(h+s-z)\,cot\,\alpha, \tag{2.17}$$

where Ca is Capillary number which indicates the ratio of viscosity μ to surface tension σ and $N = Ca^{\frac{1}{3}}\,cot\,\alpha$ measures the relative significance of the normal component of gravity (see [13,53,115] for more details). Equation (2.17) indicates that z is independent to Equation (2.16) has no influence on the evolution of the film thickness and it is therefore neglected from the next Equation (2.19). Consequently, this term is neglected in the following results. According to the derivation in the paper by Gaskell et al. [53], the choice of length scale L_0 to be equal to the capillary length L_c,

$$L_0 = L_c = \left(\frac{\sigma\, H_0}{3\,\rho\, g\, sin\alpha}\right)^{\frac{1}{3}} = \frac{H_0}{(6\,Ca)^{\frac{1}{3}}}, \tag{2.18}$$

then the pressure in Equation (2.17) can be rewritten as

$$p = \; -6\,\triangle(h+s) + 2\,\sqrt[3]{6}\,N(h+s). \tag{2.19}$$

Full details of this derivation may be found in the paper by Gaskell et al. [53].

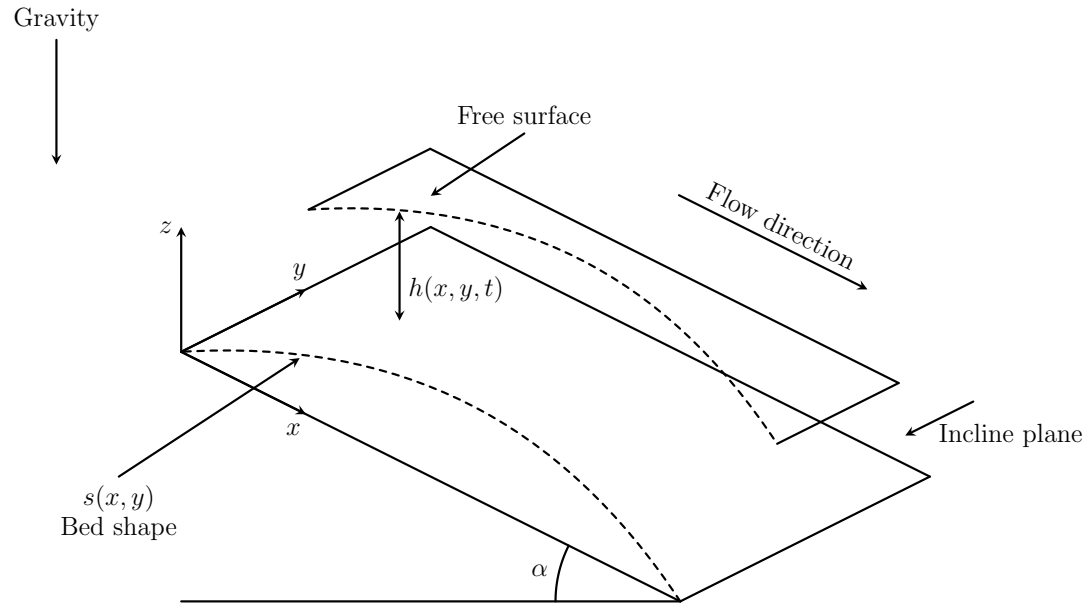


Figure 2.1: Two-dimensional thin film flow $h(x, y, t)$ over a bed shape topography $s(x, y)$ on a substrate inclined at an angle α.

Note that the thin film flow illustrated in Figure 2.1 is displayed for a two-dimensional flow $h(x, y, t)$ over a bed shape topography $s(x, y)$ on a substrate inclined at an angle α, showing the coordinate system and parameters determining the bed shape topography.

For completeness of this model, we need to define the boundary and initial conditions for both dependent variables $h(x, y, t)$ and $p(x, y, t)$. For simplicity, we have used a constant Dirichlet boundary condition. This requires that the changes to the free surface are sufficiently far from the boundaries.

In brief, the system of PDEs for fully-developed thin film flow consists of Equations (2.16) and (2.19) together with a numerical description of the bed shape topography $s(x, y)$, and appropriate initial and boundary conditions. In the following subsections, we will introduce two numerical approaches for the thin film flow model: the first one is Kalliadasis's model [76], described in one-dimension, and the second one is Sellier's model [53], in one and two dimensions.

2.2.2 Kalliadasis's Model

In this subsection, we are interested in the steady-state case, which means that we assume $\frac{\partial h}{\partial t} = 0$ in Equation (2.16). We consider here Kalliadasis's model which is applied to the

coating of a surface feature through free surface flow over a cavity. For the complete description of this system of equations, one can read [76]. The following assumptions are made: steady-state; $\alpha = 90^0$; and one-dimensional flow down the vertical plane (which means there is nothing varying in y direction, including the bed shape topography). We assume boundaries are sufficiently far from the topographic feature. Therefore, by applying this simplification to Equation (2.16) and Equation (2.19), these equations become as follows,

$$0 = \frac{d}{dx}\left[\frac{h^3}{3}(\frac{dp}{dx} - 2)\right], \tag{2.20}$$

and

$$p = -6\frac{d^2}{dx^2}(h + s). \tag{2.21}$$

By substituting Equation (2.21) into Equation (2.20) we obtain

$$0 = \frac{d}{dx}\left[\frac{h^3}{3}\left(-6\frac{d^3}{dx^3}(h + s) - 2\right)\right]. \tag{2.22}$$

Following Kalliadasis et al. [76], we scale Equation (2.22) to obtain the fourth-order mathematical model which is

$$0 = \frac{d}{dx}\left[h^3(3\frac{d^3}{dx^3}(h + s) + 1)\right]. \tag{2.23}$$

This is solved for $x \in [X_1, X_2]$, along with the boundary conditions $h = 1$ when $x = X_1$ and $x = X_2$ and $\frac{dh}{dx} = 0$ when $x = X_1$ and $x = X_2$. We then reduce Equation (2.23) from a fourth-order ODE to a third-order ODE for the film thickness $h(x)$, and then to three coupled first-order equations. By integrating Equation (2.23) once with respect to x we obtain this form,

$$h^3(3\frac{d^3}{dx^3}(h + s) + 1) = c_1. \tag{2.24}$$

Without producing a justification [76] assumes uniform flow at $x = X_1$, so that $3\frac{d^3}{dx^3}(h+s) = 0$ at $x = X_1$. This allows them to obtain $c_1 = 1$ in (2.24) to get,

$$h^3(3\frac{d^3}{dx^3}(h + s) + 1) = 1. \tag{2.25}$$

As noted by [76], we will consider here Equation (2.25) and reduce it from one third-order ODE to three coupled first-order equations. Equation (2.25) may be rearranged (with modified notation so that $(\frac{d^3h}{dx^3})$ is replaced by (h_{xxx})) to the form,

$$h_{xxx} = -s_{xxx} + \frac{1 - h^3}{3h^3}. \tag{2.26}$$

To transform this into three coupled first-order equations define unknowns as follows,

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} h \\ h_x \\ h_{xx} \end{pmatrix}. \tag{2.27}$$

From this we obtain that

$$\begin{pmatrix} u_x \\ v_x \\ w_x \end{pmatrix} = \begin{pmatrix} h_x \\ h_{xx} \\ h_{xxx} \end{pmatrix} = \begin{pmatrix} v \\ w \\ -s_{xxx} + \frac{1-u^3}{3u^3} \end{pmatrix}, \tag{2.28}$$

such that we have a system of equations in the form:

$$\begin{aligned} u_x &= v \\ v_x &= w \\ w_x &= -s_{xxx} + \tfrac{1-u^3}{3u^3}. \end{aligned} \tag{2.29}$$

Kalliadasis [76] defines 6 boundary conditions in order to ensure a unique solution to this discrete system. We follow Kalliadasis in stating the boundary conditions from his paper [76].

$$\begin{aligned} u_{(X_1)} &= u_{(X_2)} = 1, \\ v_{(X_1)} &= v_{(X_2)} = 0, \\ w_{(X_1)} &= w_{(X_2)} = 0. \end{aligned} \tag{2.30}$$

Now we have a nonlinear system of three equations and three unknowns which we will refer to as Kalliadasis's model. We will solve this system numerically in Chapter 5.

2.2.3 Sellier's Model

There is an alternative approach to Kalliadasis's model which is used by Sellier [51, 53] therefore we named it here as Sellier's model. In this system, Sellier used two coupled second-order equations rather than three coupled first-order equations as in Kalliadasis's model. The mathematical formulation of Sellier's system of equations is described in more detail in [51, 53] and [145].

We considered Sellier's model in the time-dependent case in one space dimension, which is as follows:

$$\frac{\partial h}{\partial t} = \frac{\partial}{\partial x}\left[\frac{h^3}{3}\left(\frac{\partial p}{\partial x} - 2\right)\right]. \tag{2.31}$$

We may allow any angle α, however, we simplify Equation (2.19) by assuming the angle $\alpha = 90^0$

which gives us that,

$$p = -6\,\frac{\partial^2}{\partial x^2}\,(h+s). \tag{2.32}$$

In one-dimension boundary conditions are defined as the following on the domain $x \in [X_1, X_2]$:

$$\begin{array}{rcl} h_{(X_1)} & = & 1, \\ p_{(X_1)} & = & 0, \\ \frac{\partial h}{\partial x}(X_2) & = & \frac{\partial p}{\partial x}(X_2) \quad = \quad 0, \end{array} \tag{2.33}$$

also we can use Dirichlet boundary condition at $x = X_2$. The initial conditions for the time-dependent case are provided at $t = 0$, which implies that $u(x, 0) = u_0(x)$ in one-dimension. It is essential to specify appropriate boundary conditions and initial conditions in order to solve this model in the time-dependent case. This system of PDEs for time-dependent thin film flow in one-dimension consists is solved numerically in Chapter 5.

The steady-state case of Sellier's model in one-dimension is given by the following expression,

$$0 = \frac{\partial}{\partial x}\left[\frac{h^3}{3}(\frac{\partial p}{\partial x} - 2)\right]. \tag{2.34}$$

together with the pressure Equation (2.32).

To complete this model, for both dependent variables, h and p, the boundary conditions have to be defined, as we described in the time-dependent case above.

We may also consider here the equations already introduced as Equations (2.16) and (2.19) which is Sellier's model in the time-dependent case in two space dimension, with some simplification to the pressure in Equation (2.19) by assuming the angle $\alpha = 90^0$ which is stated as follows,

$$p = -6\,\triangle(h+s), \tag{2.35}$$

where $\triangle$ is the two-dimensional Laplacian,

$$\triangle = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \tag{2.36}$$

The steady-state case of Sellier's model of the thin film flow system of equations in two-dimensions is given by the following expression,

$$0 = \frac{\partial}{\partial x}\left[\frac{h^3}{3}(\frac{\partial p}{\partial x} - 2)\right] + \frac{\partial}{\partial y}\left[\frac{h^3}{3}(\frac{\partial p}{\partial y})\right], \tag{2.37}$$

together with the pressure Equation (2.35).
Moreover, the time-dependent case of Sellier's model of the thin film flow system of equations

in two-dimensions is given by the following expression,

$$\frac{\partial h}{\partial t} = \frac{\partial}{\partial x}\left[\frac{h^3}{3}(\frac{\partial p}{\partial x} - 2)\right] + \frac{\partial}{\partial y}\left[\frac{h^3}{3}(\frac{\partial p}{\partial y})\right], \tag{2.38}$$

together with the pressure Equation (2.35). In two-dimension, we have defined the Dirichlet boundary condition again here on the domain $(x, y) \in [X_1, X_2] \times [Y_1, Y_2]$ as the following:

$$\begin{array}{rclcrclcl} h_{(X_1,Y)} & = & h_{(X_2,Y)} & = & h_{(X,Y_1)} & = & h_{(X,Y_2)} & = & 1, \\ p_{(X_1,Y)} & = & p_{(X_2,Y)} & = & p_{(X,Y_1)} & = & p_{(X,Y_2)} & = & 0, \end{array} \tag{2.39}$$

We define the initial conditions for the time-dependent case in two-dimension for each variable as $h(x, y, t = 0) = h_0(x, y)$ and $p(x, y, t = 0) = p_0(x, y)$. In the following section, we have described another nonlinear system which is phase-field models.

2.3 Phase-Field Models

2.3.1 Introduction

In many scientific fields, models may arise which involve the evolution of interfaces between materials. We introduce here the notion of phase-field models that are used to approximate interface problems based upon a diffuse interface. Phase-field models are not the only technique to compute the evolution of interfaces. Other approaches include modelling as a free boundary problem where a sharp interface [75] is used and explicitly tracked. A further technique is called the level set method which is described in [80] for example. We consider a phase-field application as a further example of a nonlinear parabolic system of PDEs.

The application of a diffuse interface approximation based upon a supplementary phase variable (ϕ) is described in [84]. The phase-field variable is smooth (differentiable) and is nearly constant valued in most of the domain, but taking a different constant value in each phase. However, between phases, at the interface regions, the value of the phase-field variable changes smoothly but rapidly between the two bulk values [36]. There are many different applications for phase-field models which are presented in the literature [36,50,88,89,120], but they all based on the diffuse interface approximation. The phase-field equations themselves are typically derived from physical energy minimisation principles [84]. Moreover, in [4, 34, 38, 59, 84] there are many applications which are presented for phase-field models.

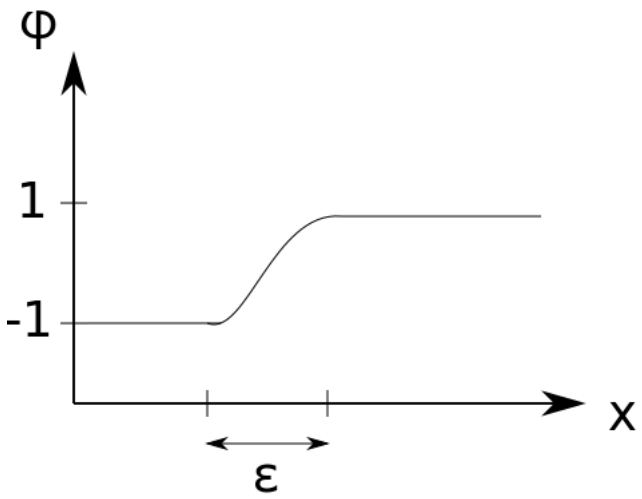


Figure 2.2: Description of phase variable ϕ in a two-phase-field system, describing two phases by values of $\phi = 1$ and $\phi = -1$.

Figure 6.2 displays a phase variable ϕ in a two-phase system: $\phi = 1$ describes the first phase (e.g. solid) and $\phi = -1$ describes the second phase (e.g. liquid). These two phases are isolated via a diffuse interface with a thickness ε. The selection of ε is important in the phase-field models because if this value becomes very small then the models will become difficult to solve numerically due to sharp interface issues. If the interface is too diffuse the model will not be accurate. The variable for the phase-field is coupled with physical variables, such as energy, temperature and velocity to form the governing equations [84]. Note that no sharp interface tracking is required as the position of the interface is recovered from the values of ϕ (e.g. the contour $\phi = 0$ in this case).

2.3.2 A Two-Phase Model: The Cahn-Hilliard-Hele-Shaw Model

The Cahn-Hilliard (CH) equation begins from the work in [35,36], and plays a vital role in many application including the study of multi-phase flows such as [141]. After that, Shinozaki and Oono [120] applied a variation of the CH equations to simulate a binary fluid in a Hele-Shaw cell and the result of this model is named as the Cahn-Hilliard-Hele-Shaw (CHHS) system of equations. The CHHS is derived by Lee et al. in [89]. Since the most of the mathematical models concentrate on different aspects, this system of equations is important and represent some of these aspects this is because of that it appears in several models such as spinodal decomposition of a binary fluid in a Hele-Shaw cell, cell sorting, and tumour growth and flows in porous media [88, 89, 120, 140, 141]. We present here a simplification of the CHHS system,

which was presented in Wise's paper [139] and Yang's thesis [145]:

$$\frac{\partial \phi}{\partial t} = \Delta \mu - \nabla.(\phi\ \mathbf{u}), \tag{2.40}$$

$$\mu = \phi^3 - \phi - \varepsilon^2\, \Delta \phi, \tag{2.41}$$

$$\mathbf{u} = -\nabla\, p - \gamma \phi \nabla \mu, \tag{2.42}$$

$$\nabla.\mathbf{u} = 0. \tag{2.43}$$

Here $\mu(x, y, t)$ is a chemical potential, $\mathbf{u}$ is the fluid advective velocity, ε is the phase-field interface thickness, $\gamma > 0$, $\phi(x, y, t) = \pm 1$ denotes pure fluids, and $p(x, y, t)$ is a pressure. The CHHS model that we used in this research requires $\gamma > 0$, but if we use $\gamma = 0$, then Equations (2.40), (2.42) and (2.43) will be reduced to the CH equations [35].

In practice, when substituting Equation (2.42) into Equation (2.40) and Equation (2.43), we obtain three coupled equations for ϕ, μ, and p:

$$\frac{\partial \phi}{\partial t} = \nabla.((1 + \gamma\phi^2)\nabla\mu) + \nabla.(\phi \nabla p), \tag{2.44}$$

$$\mu = \phi^3 - \phi - \varepsilon^2\, \Delta \phi, \tag{2.45}$$

$$-\Delta p = \gamma\, \nabla.(\phi \nabla \mu). \tag{2.46}$$

In [139] Wise solves on $[-L_x, L_x] \times [-L_y, L_y]$ where $L_x = L_y = 3.2$, with a zero Neumann boundary condition on all variables. We therefore complete the above system with the boundary conditions as follows,

$$\frac{\partial \phi}{\partial n} = \frac{\partial \mu}{\partial n} = \frac{\partial p}{\partial n} = 0\ on\ \partial\Omega, \tag{2.47}$$

where n here is the unit outward normal of the boundary $\partial\Omega$. Note that the pressure variable p appears only in gradient form in the CHHS system, consequently, it has no unique solution and we will deal with this issue in Chapter 6.

A consistent initial condition is also required. Hence, given $\phi(x, y, t = 0)$, we evaluate Equation (2.45) to get $\mu(x, y, t = 0)$. Given $\phi(x, y, t = 0)$ and $\mu(x, y, t = 0)$ it is then necessary to solve Equation (2.46) to obtain $p(x, y, t = 0)$. We will consider the numerical solution of the CHHS model again in more detail in Chapter 6.

In this chapter, we have introduced the mathematical models that we will solve numerically in this thesis. In the next chapter, we move to discuss some of the most broadly applied methods for the discretisation of PDE's in space and time as well as the numerical methods for solving the discrete linear and nonlinear system of equations that arise. Following that, we can discretise and solve the mathematical models that have been presented in this chapter.

Chapter 3

Introduction to Discretisation Techniques

In the previous chapter, we discussed a range of elliptic and parabolic PDE systems and we presented two particular mathematical models that we will consider in more detail in this thesis: a thin film model and a phase-field model in two-dimensions. In this chapter, we introduce some of the fundamental concepts behind the discretisation of PDEs in space and time. The discretised PDEs which form systems of linear and nonlinear algebraic and differential algebraic equation standard solution algorithms for these problems are presented.

Scientific computing and numerical approximation techniques are used to solve mathematical problems for which it is not possible to obtain exact solutions by analytical methods. The choice of a particular numerical scheme is subjective as each has advantages and disadvantages, which may depend on the PDE or the domain on which it is defined. The goal of this chapter is to introduce the discretisation techniques that are later applied to the one-dimensional mathematical models defined in Chapter 2 and their generalization to two-dimensional problems.

A discretisation scheme for a PDE, or a system of PDEs, is a process of approximating a continuous mathematical solution as the solution of a discrete system of algebraic equations. One of the most common, and simplest, schemes is the finite difference method which is discussed in Section 3.1.1. There are other discretisation schemes that may also be applied, and some of these are outlined briefly in Section 3.1.2. Each of these discretisation methods requires the approximation of the spatial parts of the PDEs with a discrete representation based on a finite number of parameters. In the case of an elliptic PDE, or system, this results in an algebraic system of equations to determine these parameters.

In the case of a parabolic PDE, or system, application of these spatial discretisation methods reduces the problem to an initial value system of Ordinary Differential Equations (ODEs). Approximation in space by finite difference or finite element methods will result in sparse systems of algebraic equations. This is important here for improving efficiency of our solution algorithms and is considered in Section 3.2. The spatial approximation is followed by a discretisation in time to complete the reduction to algebraic form. When the PDE system contains a mixture of parabolic and elliptic equations the spatial discretisation results in a system of *Differential Algebraic Equations (DAEs)* which must then be solved (see Section 3.4).

3.1 Spatial Discretisation

Spatial discretisation methods approximate the spatial derivatives of the PDEs with a finite number of degrees of freedom. In many approaches, this approximation is determined based on decomposing the spatial domain Ω into a set of points or cells, as in finite difference or finite element methods for example. We introduce in this section various options for the spatial discretisation. In Section 3.1.1, we introduce the finite difference method and we present a selection of other spatial methods in Section 3.1.2.

3.1.1 Finite Difference Method

Finite difference methods (FDM) are commonly used in engineering and science research applications (see, amongst others, [81,87,117,121]) due to their simplicity, especially on rectangular geometries. FDM depend on the approximation of a derivative as the difference in the dependent variable over a finite set of points [72,121]. We limit our discussion here to a fixed spacing between adjacent nodes. In other words, the spatial domain of the model is divided into a set of uniformly spaced points. This means that the spatial domain $[X_1, X_2]$ is approximated as a set of $N+1$ points $\{x_1, x_2, x_3, \ldots, x_{N+1}\}$, where

$$x_i = X_1 + (i-1)\ \Delta x, \quad 1 \leq i \leq N+1, \tag{3.1}$$

$$\Delta x = \frac{X_2 - X_1}{N}. \tag{3.2}$$

Here Δx indicates the mesh size. The goal is to approximate the dependent variables through estimation of their values over this set of points. That is, we seek to compute their value at each node i.

Spatial derivatives are approximated at a given point x by using Taylor series expansions to link the derivative at that point to values of the dependent variable at neighbouring nodes. The

Taylor expansion of $f(x + \Delta x)$ is as follows:

$$f(x + \Delta x) = f(x) + \frac{f'(x)}{1!}\Delta x + \frac{f^{(2)}(x)}{2!}\Delta x^2 + \cdots + \frac{f^{(n)}(x)}{n!}\Delta x^n + \cdots \tag{3.3}$$

and for $f(x - \Delta x)$,

$$f(x - \Delta x) = f(x) - \frac{f'(x)}{1!}\Delta x + \frac{f^{(2)}(x)}{2!}\Delta x^2 - \cdots + (-1)^n\frac{f^{(n)}(x)}{n!}\Delta x^n + \cdots \tag{3.4}$$

The *forward difference approximation* of the first derivative f', is derived from Equation (3.3) as:

$$f'(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x}. \tag{3.5}$$

This results in truncation error terms from the Taylor series approximations which describes the accuracy of these expressions. For the first derivative,

$$\frac{f(x + \Delta x) - f(x)}{\Delta x} = f'(x) + \frac{f^{(2)}(x)}{2}\Delta x + \cdots \tag{3.6}$$

or

$$\frac{f(x + \Delta x) - f(x)}{\Delta x} = f'(x) + \mathcal{O}(\Delta x). \tag{3.7}$$

We use the $\mathcal{O}(\Delta x)$ symbol to illustrate the linear dependence of the truncation error on the mesh spacing. This means that the error of the approximation is proportional to Δx as $\Delta x \longrightarrow 0$. Likewise, the *backward difference approximation* of the first derivative of f' is derived from Equation (3.4) as:

$$\frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) + \mathcal{O}(\Delta x). \tag{3.8}$$

These approximations of f' at point x are said to be consistent at first-order [2, 121]. By subtracting Equation (3.4) from Equation (3.3) we obtain the *central difference approximation* to the first derivative,

$$\frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} = f'(x) + \mathcal{O}(\Delta x^2). \tag{3.9}$$

This gives a second-order consistent approximation to f'. Similarly, we can derive an approximation of the second derivative $f^{(2)}$, by adding Equations (3.3) and (3.4), which allows one to derive the expression,

$$\frac{f(x + \Delta x) - 2f(x) + f(x - \Delta x)}{\Delta x^2} = f^{(2)}(x) + \mathcal{O}(\Delta x^2). \tag{3.10}$$

This is called the central difference approximation for the second derivative. On our finite difference grid we can apply these approximations at each point x_i. For instance, using a grid

such as that illustrated in Figure 3.1, we may approximate the differential equation,

$$\frac{d^2u}{dx^2} = f, \tag{3.11}$$

with the algebraic system,

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} = f_i, \tag{3.12}$$

for i=2,...,N, and $u_i = u(x_i)$ (where we assume Dirichlet boundary conditions define u_1 and u_{N+1}).

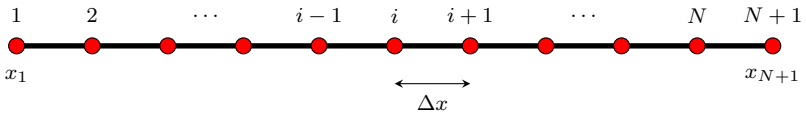


Figure 3.1: An example of a 1D mesh with uniform spacing.

Next we present a simple parabolic PDE in one space dimension. This is the heat conduction equation (or the diffusion equation as in chemical problems) which takes the following form:

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2}, \tag{3.13}$$

where $u = u(x,t)$, a^2 is the diffusivity, [108]. We define the spatial domain here to be a region Ω, whilst the temporal domain is defined to be $(0,T)$. On the region Ω let us create a mesh as in Figure 3.1.

To complete this problem, we define the initial and boundary conditions as follows,

$$u(x,0) = g(x), \quad X_1 < x < X_{N+1}, \tag{3.14}$$

$$u(X_1,t) = f_1(t), \quad 0 < t < T, \tag{3.15}$$

$$u(X_{N+1},t) = f_2(t), \quad 0 < t < T, \tag{3.16}$$

where the functions $g(x)$, $f_1(t)$ and $f_2(t)$ are given and this process starts when $t = 0$.

The generalisation of finite difference schemes to two or three dimensions is straightforward. Following the same approach as in one-dimension, we may use the second order central differences formula for each second derivative on the uniform grid to obtain

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta y^2} = \mathcal{O}(\Delta x^2, \Delta y^2), \tag{3.17}$$

where $u_{i,j} \approx u(x_i, y_j)$ for $2 \leq i \leq N$ and $2 \leq j \leq M$. For example, consider the discretisation

of the 2D Poisson equation,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \tag{3.18}$$

on the domain $[X_1, X_2] \times [Y_1, Y_2]$ with Dirichlet boundary conditions. We may approximate this domain by a set of points (x_i, y_i) where

$$x_i = X_1 + (i-1)\ \Delta x, \quad 1 \leq i \leq N+1, \tag{3.19}$$

$$y_j = Y_1 + (j-1)\ \Delta y, \quad 1 \leq j \leq M+1, \tag{3.20}$$

$$\Delta x = \frac{X_2 - X_1}{N}, \tag{3.21}$$

$$\Delta y = \frac{Y_2 - Y_1}{M}. \tag{3.22}$$

Here Δx indicates the mesh size in x direction and Δy indicates the mesh size in y direction. The functions are approximated over these sets of points, or mesh, in terms of their values at each node (x_i, y_i) [121]. A section of this finite difference grid in 2D is shown in Figure 3.2.

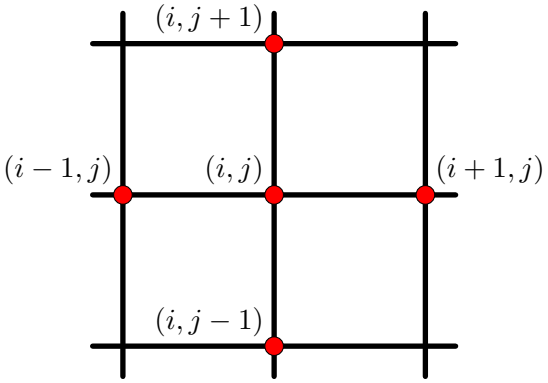


Figure 3.2: Part of a 2D mesh with each component of the five-point stencil marked as a red circle.

In Figure 3.2, we have displayed a vertex mesh of each component of the five-point stencil marked as a red circle in 2D. We can define the Dirichlet boundary condition on boundary points directly.

These finite difference strategies will be used in the following chapters to approximate the solutions of nonlinear parabolic PDEs. The goal of each of these spatial discretisation methods is to reduce the parabolic problems to initial value systems of ordinary differential equations. These systems are discrete in space, however, they are still continuous in time [32].
Temporal discretisation schemes will be discussed in Section 3.3, however before this in the next subsection, we will introduce some other spatial discretisation schemes that are commonly

used elsewhere.

3.1.2 Other Spatial Discretisation Schemes

There are several different alternatives for spatial discretisation schemes. Two common choices are the finite volume and the finite element methods, although such schemes will not be considered in this research. In the finite volume scheme a solution is approximated on a grid using the cell average of the integral over that element. This approach is very common for hyperbolic systems of equations which often arise from methods of advective flows, as in [90]. Finite volume schemes can also be used to find the solution of PDEs of elliptic and parabolic type [94]. However, for elliptic and parabolic problems finite difference and finite element discretisations are much more widespread. For more detail on finite volume methods we refer the reader to [90, 133].
Next, we briefly mention finite element methods (FEM), which are used in various areas of computational science and engineering to find approximate solutions to PDEs [47, 146]. These schemes are often used for problems in which the domain is geometrically complex. The FEM relies on the weak form of PDEs which makes it well suited for problems with Neumann boundary conditions. As with the finite difference method, the spatial domain is covered with a set of points.

In one dimension the solution of the governing differential equations is approximated over the set of intervals between the points by low order polynomials between each pair of neighbouring points (a spatial element). The set of N non-overlapping elements, or mesh, totally covers the domain. The precise details are not described here, but the method is based upon representing the approximate solution in terms of a set of local basis functions that interpolate the approximation over the mesh. The approximated weak form is integrated over the set of elements to convert the PDE system to a sparse system of algebraic equations which can then be solved for the coefficients of the basis functions. There are many textbooks that focus on the FEM theory and for more information, the interested reader is referred to the literature [107] and [72, 81, 87, 106, 117] for a focus on the practical side of this method.

In this thesis, we shall focus solely on the FDM as this technique is sufficient to investigate multilevel techniques based on local stencils. Similar multilevel solution techniques based on local stencils from FEM or FVM are possible, however, we will not consider them in this thesis. An interested reader can learn more about these techniques in [18, 79, 90, 124].

Furthermore, there are other types of discretisation schemes which are based on global stencils. For example, spectral methods use infinitely differentiable basis functions defined globally (e.g. a Fourier basis) which are orthogonal. These schemes are less suitable for multilevel solvers as it is not generally possible to define a suitable smoother. We direct the reader to [37] for

further detail on the study of these methods. An example of another discretisation method is the collocation method (see [72]).

3.2 Sparse Matrices

A *sparse matrix* is a matrix with the majority of its components zero. More precisely, an $N \times N$ matrix B is said to be sparse if the number of non-zero entries grows like $O(N)$ as N grows. Conversely, if the majority of its components are non-zero, then we say that this matrix is *dense*. The dense matrix always requires a large amount of memory and time in order to solve large problems, since it is necessary to store the full matrix. However, this issue does not appear when we use sparse matrices. The FDM or FEM methods typically produce sparse matrices. Sparse matrix algorithms can be employed for common matrix operations, and also when storing and manipulating matrices on a computer. They typically require specialised data structures. Sparse matrices require much less storage than dense matrices of the same size. The advantage of a sparse matrix is that produces enormous savings in memory and floating point operations when we are storing only the nonzero entries in some format.

There are two common sorts of sparse matrices: structured (regularly structured) and unstructured (irregularly structured). The first type is comprised of regularly structured matrices whose non-zero entries form a regular known pattern, usually with a small number of non-zero diagonals. The simplest examples of regularly structured matrices are matrices that consist of only a few diagonals (e.g. bi-diagonal, tri-diagonal or penta-diagonal). The second type is comprised of irregularly structured matrices, which are matrices with irregularly located non-zero entries.

The efficient algorithms have to work with these sparse matrices with data stored in the specific storage format. There are different storage formats of sparse matrices, for example, the simplest storage format for a sparse matrix is known as the coordinate format, COO, which is a very flexible format. Usually, this format stores a listing of (row, column, value). Where the entries are sorted first by row and then by column, to improve memory access times. The constructing of the data depends on three arrays:

- an array that stores all real values (or complex) of nonzero elements of the matrix, AA,
- a corresponding integer array including their row numbers, JR,
- a corresponding integer array including their column numbers, JC.

When NNZ indicates the total number of nonzero elements, then all three arrays are NNZ length, the number of nonzero elements.

For example, consider the matrix whose entries are given as follows, [111, 123]:

$$A = \begin{pmatrix} 16. & 0. & 0. & 21. & 0. \\ 36. & 24. & 0. & 15. & 0. \\ 9. & 0. & 8. & 7. & 6. \\ 0. & 0. & 20. & 1. & 0. \\ 0. & 0. & 0. & 0. & 11. \end{pmatrix}.$$

Then the COO format gives:

AA	16.	15.	36.	24.	21.	9.	8.	11.	6.	20.	1.	7.

JR	1	2	2	2	1	3	3	5	3	4	4	3

JC	1	4	1	2	4	1	3	5	5	3	4	4

This method is very simple storage format. Another storage format is the Compressed Sparse Row (CSR) format which represents a matrix A by three one-dimensional arrays, that respectively contain nonzero values, the starting location of rows, and column indices. This format allows fast row access. In this format, the new data construction has three arrays with the following functions:

- A real array AA contains the real values a_{ij} stored row by row, from row 1 to n and the length of AA is NNZ.
- An integer array JA contains the column indices of the elements a_{ij} as stored in the array AA and the length of JA is NNZ.
- An integer array IA contains the pointers to the beginning of each row in the arrays AA and JA. Therefore, the content of $IA(i)$ is the location in arrays AA and JA where the i^{th} row begins. The length of IA is $n+1$ with $IA(n+1) = NNZ + 1$.

AA	16.	21.	36.	24.	15.	9.	8.	7.	6.	20.	1.	11.

JA	1	4	1	2	4	1	3	4	5	3	4	5

IA	16	36	9	20	11

This scheme is preferred over the COO format because it is usually more helpful for implementing standard computations. Therefore, CSR is most commonly applied as the entry format in sparse matrix software packages. An alternative format to the Compressed Sparse Row CSR format is storing the columns instead of the rows. This format is called the Compressed Sparse Column (CSC) format but is less commonly used.

Furthermore, sparse matrix operators and algorithms should be efficient, in terms of computational work as well as storage requirements, as mentioned above (cf [65] and [111]). In the following section, we will consider temporal discretisation schemes.

3.3 Temporal Discretisation

The spatial discretisations explained in the previous section are used for the nonlinear systems of differential equations which are considered in detail in Section 5.2.1 and Section 5.2.3 in Chapter 5. In this thesis, our foremost interest is not only in solving steady-state problems but also to solve time-dependent problems. Consequently one has to think about suitable temporal discretisation schemes. The schemes mainly used here are implicit multi-step methods which will be presented in Section 3.3.2.

Now, we introduce the temporal discretisation schemes in the following subsections. Integration methods can be divided into implicit and explicit. Explicit methods are cheap per step but subject to some stability conditions, see [32] and [58]. These conditions tend to be especially severe for parabolic problems and so explicit methods are not usually used. For example, in [121] Smith describes that $\Delta t < C\,(\Delta x)^2$ for second-order parabolic systems and $\Delta t < C\,(\Delta x)^4$ for fourth-order parabolic systems, for some constant C [12]. Implicit methods are more costly per step, but have improved stability, allowing fewer, larger time steps, and consequently, are widely used for such problems.

3.3.1 Implicit Versus Explicit Integration Methods

In Subsection 3.1.1 we described a semi-discrete system that is discrete in space and continuous in time, (i.e. takes the form of an initial value system of ordinary differential equations ODEs). Therefore, we need to obtain a fully discrete system that can be used for numerical computations. In this section, we proceed with the study of temporal discretisation schemes. Several numerical schemes of the implicit and explicit schemes have been discussed in the literature to solve various mathematical models in engineering and in science, see [33], [32], [85], [71] and [56].

There are many classes of explicit methods that can provide a higher order of accuracy, such as the explicit Runge-Kutta methods which are classified as one-step (multi-stage) methods [118, 129]. There are also explicit multi-step methods (e.g. Adams-Bashforth method, finite difference methods [33, 72]). Explicit schemes, in general, are calculated at the current time step only from known values from the previous time steps.

The simplest example of an explicit scheme is the Forward Euler scheme. The explicit methods are simple to solve, and this is the main advantage of these methods. Nevertheless, explicit methods are generally only conditionally stable. In other words, the numerical solution is stable only when the time step size is adequately small. Therefore, implicit schemes can be used to replace explicit ones in cases where the stability requirements of the latter impose stringent conditions on the maximum time step size.

As implied above, implicit methods are more expensive to implement, especially for nonlinear problems, but the advantages of these schemes are to achieve satisfactory accuracy with larger time steps. Taking into account that one needs to solve a system of nonlinear equations at each time step the increase in the time step size over explicit schemes may have to be very significant to make this approach worthwhile. A simple example of an implicit scheme is the backward Euler scheme. The choice between the explicit or implicit schemes is based upon the problems itself that we want to solve and the accuracy that we require. Moreover, to control the issues of conditional stability, we used the implicit techniques since these techniques have much greater regions of stability than explicit techniques. We direct the reader to [71, 72, 121, 139] for further details.

We present here more detail of the simplest examples of an explicit and an implicit scheme, known as the Forward and the Backward Euler scheme, respectively. We will be concerned with the following general ODE initial value problem IVP:

$$\frac{dy}{dt} = f(t, y) \text{ in } (0, T], \tag{3.23}$$

with the initial condition at time $t = t_0$ as $y(t_0) = y_0$.

The Forward Euler method which is also called the explicit Euler method is probably the simplest method to solve the IVP in Equation (3.23). To solve the IVP in Equation (3.23) by the Forward Euler method, we replace the time derivative by

$$\frac{dy}{dt}\Big|_{t=t_n} = \frac{y_{n+1} - y_n}{\Delta t} + \mathcal{O}(\Delta t). \tag{3.24}$$

The IVP in Equation (3.23) then reduces to

$$\frac{y_{n+1} - y_n}{\Delta t} = f(t_n, y_n) + \mathcal{O}(\Delta t).$$

This gives us

$$y_{n+1} \cong y_n + \Delta t\, f(t_n, y_n), \tag{3.25}$$

for $n = 0, ..., M$. We can see that given an initial condition y_0, all the quantities on the right-hand side of the equation (3.25) are known from the previous step. We say that this method is of order one, because of the truncation error in Equation (3.24).

The Backward Euler method is an implicit scheme for solving first-order ODEs. Here y_{n+1} is presented only in terms of an implicit equation. To solve the IVP in Equation (3.23) by the

Backward Euler method, we replace the time derivative by

$$\frac{dy}{dt}\Big|_{t=t_{n+1}} = \frac{y_{n+1} - y_n}{\Delta t} + \mathcal{O}(\Delta t). \tag{3.26}$$

The IVP in Equation (3.23) then reduces to

$$y_{n+1} \cong y_n + \Delta t\, f(t_{n+1}, y_{n+1}). \tag{3.27}$$

The θ-scheme is an implicit numerical scheme given here as follows:

$$y_{n+1} = y_n + \Delta t\, [\theta f(t_{n+1}, y_{n+1}) + (1-\theta) f(t_n, y_n)], \tag{3.28}$$

where $\theta \in [0, 1]$. It is worth pointing out that when $\theta = 0$ the scheme is the explicit forward Euler scheme Equation (3.25) and when $\theta = 1$ this scheme is the implicit backward Euler scheme Equation (3.27) [72] and [32]. When $\theta = \frac{1}{2}$ we obtain the second-order CrankNicolson scheme:

$$y_{n+1} = y_n + \frac{\Delta t}{2}\, [\, f(t_{n+1}, y_{n+1}) + f(t_n, y_n)]. \tag{3.29}$$

For example, an application of this scheme can be seen in [51]. There schemes are limited in accuracy to at most order two ($\theta = \frac{1}{2}$).

In the next section, we introduce a family of backward differentiation formulae. In this case, the accuracy can have arbitrary high order.

3.3.2 A Family of Backward Differentiation Formulae

We will proceed with the study of the most popular implicit multi-step method known as the Backward Differentiation Formulae BDF [5, 6]. These are implicit methods known to have excellent stability [63, 85]. The general formula (or the p-step formula) for the BDF schemes can be given as follows [6],

$$y_{n+1} = \sum_{j=0}^{p-1} \alpha_j\, y_{n-j} + \Delta t\, \beta f(t_{n+1}, y_{n+1}), \tag{3.30}$$

where the superscript $n+1$ denotes the unknown solution at the current time step, Δt is the size of the time step (which is supposed constant for this description), $n-j$ indicates the known solution from the previous time step, and $t_n = t_0 + n\,\Delta t$. The coefficients α_j and β are parameters chosen so that the method achieves order p accuracy, (i.e. p denotes the order of accuracy in time). More features of these methods can be found in [5,63,85]. The first 6 formulae of this family are tabulated in Table 3.1 [6].

The algebraic system arising from an application of a fully implicit scheme is nonlinear if the

PDE is nonlinear.

Table 3.1: The coefficients of the family of BDF methods in Equation (3.30) for order up to $p = 6$.

p	β	α_0	α_1	α_2	α_3	α_4	α_5
1	1	1					
2	$\frac{2}{3}$	$\frac{4}{3}$	$-\frac{1}{3}$				
3	$\frac{6}{11}$	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$			
4	$\frac{12}{25}$	$\frac{48}{25}$	$-\frac{36}{25}$	$\frac{16}{25}$	$-\frac{3}{25}$		
5	$\frac{60}{137}$	$\frac{300}{137}$	$-\frac{300}{137}$	$\frac{200}{137}$	$-\frac{75}{137}$	$\frac{12}{137}$	
6	$\frac{60}{147}$	$\frac{360}{147}$	$-\frac{450}{147}$	$\frac{400}{147}$	$-\frac{225}{147}$	$\frac{72}{147}$	$-\frac{10}{147}$

3.4 Differential Algebraic Equations

Differential Algebraic Equations DAEs can arise from the spatial discretisation of mixed PDEs where some are time-dependent and some are not. This can occur in a broad variety of problems from engineering and science such as mechanics, electrical circuits, fluids, dynamics, and chemical process control [5].

The most general DAE form is given by the equation

$$F\left(\dot{x}, x, t\right) = 0,\ t_0 \leq t \leq t_f, \tag{3.31}$$

where $x = x(t) = (x_1(t), \ldots, x_n(t))$ is a vector of dependent variables and suitable initial conditions are defined, $x(t_0) = x_0$. The DAEs are not directly solvable for the derivatives of all components of the function x because some equations are algebraic equations. Hence they are more general than the ODE systems of the form

$$\dot{x} = F(x, t). \tag{3.32}$$

A typical way to present a DAE system is to let x denote the variables described by ODEs and y denote the algebraic variables. The DAE system may be then rewritten as follows:

$$\dot{x}(t) = f(x(t), y(t), t), \tag{3.33}$$

$$0 = g(x(t), y(t), t). \tag{3.34}$$

Additional information about DAE systems can be found in [24]. Note that the spatial discretisation for the time-dependent thin film flow model of Equations (2.31) and (2.32) takes the

form of Equations (3.32) and Equation (3.33).

This may be expressed as,

$$\begin{aligned} \frac{\partial Y}{\partial t} &= F_h\left(t, Y, Z\right), \\ 0 &= F_p\left(Y, Z\right), \end{aligned} \tag{3.35}$$

where

$$\begin{aligned} Y &= \left(h_1, h_2, h_3, ..., h_{n+1}\right)^T, \\ Z &= \left(p_1, p_2, p_3, ..., p_{n+1}\right)^T. \end{aligned} \tag{3.36}$$

Note that we present this example in more detail in Chapter 5, where the precise definition of F_h and F_p are discussed in the more details. In order to solve the system (3.35) by using the θ-method, we write,

$$\begin{aligned} \frac{Y_{n+1} - Y_n}{\triangle t} = &\ \theta\, F_h\left(t_{n+1}, Y_{n+1}, Z_{n+1}\right) + \left(1-\theta\right) F_h\left(t_n, Y_n, Z_n\right), \\ 0 = &\ F_p\left(Y_{n+1}, Z_{n+1}\right). \end{aligned} \tag{3.37}$$

In the following chapter, some basic solution algorithms are introduced for the discrete algebraic systems that arise from implicit temporal discretisation schemes.

Chapter 4

Solution Algorithms for Algebraic Systems

4.1 Introduction

In this chapter, we introduce various standard solution methods for the algebraic systems arising from the discretisation of linear and nonlinear elliptic and parabolic PDEs and systems. We provide an introduction to some of the most important methods in order to solve these systems of algebraic equations. We conclude with a description of three nonlinear multilevel algorithms, which are the focus of this thesis.

This chapter begins with a short summary in section 4.1.1 of the application of direct solvers for linear systems of equations. In section 4.1.2, a brief overview of the variety of iterative methods for linear and nonlinear problems is given. In section 4.2, we discuss some of the iterative methods for linear systems in more detail. In this section, we also introduce the basic computations required to apply multigrid algorithms for solving appropriate linear algebraic systems. Furthermore, Krylov subspace methods are discussed, along with the discussion of their preconditioning. We present some iterative methods for nonlinear algebraic systems in section 4.3, including Newton's method and some important nonlinear iterations: FAS, Newton-Multigrid and Newton-Krylov.

4.1.1 Direct Methods

Let us consider the following linear system of n equations:

$$A\ u = b, \tag{4.1}$$

where $A \in \mathbb{R}^{n \times n}$ is a coefficient matrix, $u \in \mathbb{R}^n$ is the unknown vector and $b \in \mathbb{R}^n$ is known. If the matrix A is nonsingular then one can find a unique solution for this system which is given by $u = A^{-1}\, b$. The action of the matrix A^{-1} is needed in an efficient way however, as this matrix is not commonly explicitly formed [55]. These solvers produce the exact solution (up to rounding error) in a finite and predictable number of steps. Standard methods include Gaussian elimination, LU decomposition and Cholesky decomposition (cf [5, 32, 55, 72, 111]). For dense matrix systems, their expense is $O(n^3)$ making them impractical for very large problems.

The discretisation of linear PDE systems most often requires a large sparse matrix system to be solved. The advantages of this sparse matrix system is that it can be stored on the computer with much decreased storage needs over a dense matrix, therefore, it is important to exploit the properties of the sparse matrix in practice. Direct methods tend to be quite memory intensive since these methods construct a decomposition of matrix A, which may not be sparse even if A is. Therefore, for large problems, such direct methods can be prohibitive because they allow the fill-in of matrix entries that change initially zero entries to non-zero values through the execution of the algorithm.

By making careful use of re-ordering of the equations and the unknowns, the best sparse direct solvers for large sparse matrices are able to provide solution methods with asymptotic time complexity $O(n^{3/2})$, where n is the number of unknowns [54]. This has been accomplished in a number of computer packages for sparse matrix computation on sequential and parallel architectures, notably MUMPS, a MUltifrontal Massively Parallel sparse direct Solver [3] and SuperLU [91]. In this thesis, we have exploited the MATLAB "backslash" function as an efficient direct solver for solving large sparse linear systems such as Equation (4.1). This function is used for solving the simultaneous linear equations in MATLAB, expressed by $u = A \setminus b$, which relies on the structure of the coefficient matrix A, which is equivalent to LU factorizing the matrix. However, there are many other software libraries that efficiently implement sparse direct methods in serial and parallel [3, 14, 44, 91].

A sparse direct solver based upon LU factorisation includes four stages:

- Applied reordering stage, which reorders the rows and columns in an attempt to approximately minimize fill-in at the factorization stage.
- Determine the non-zero structure of the factors and allocate memory efficient data structures for the factors by applying a symbolic analysis of the factorisation stage.

– The numerical factorisation stage in which the L and U factors are formed.

– The solution stage, using the factors to execute the forward and back substitution.

Note that for a non-symmetric matrix numerical stability may also be an issue, so the numerical factorization could cause the ordering and symbolic factorization stages to have to be revisited. Typically, sparse direct algorithms are very good solvers and are very efficient for medium to large problem sizes. However, for very large sparse systems we need an algorithm to have asymptotic complexity that is as close to optimal as possible. Such linear time complexity of $O(n)$ is not generally achievable using direct methods. Consequently, we will not consider sparse direct solvers in more detail, but we will consider alternative solvers (which are the iterative methods).

In the next subsection, we continue the discussion of methods for solving linear systems of equations that are suitable for use with large sparse problems with an alternative approach by iterative solvers.

4.1.2 Iterative Methods

In contrast to direct solvers which obtain an exact solution (apart from rounding errors) in a finite number of steps, with iterative methods we seek to obtain an acceptable approximation with a finite number of iterations. A series of approximate solutions which converge to the exact solution of the linear system are produced by iterative methods. Iterative methods may be employed for linear or nonlinear problems. Iterative methods start with an initial guess for the solution and successively update it until the solution is as accurate as required.

From a theoretical point of view, an infinite number of iterations could be needed to converge to the exact solution, but, from a practical point of view, termination of the iteration occurs when some norm of the residual (or any suitable measure of error) is as small as required. Iterative methods for the solution of linear systems of equations abound (cf [55, 77, 111]). Section 4.2.2 outlines stationary iterative methods such as Jacobi, weighted Jacobi and Gauss-Seidel [55, 72]. Relaxation methods for solving large linear systems are rarely used separately now. Nevertheless, if we combine these basic methods with more efficient algorithms such as Multigrid and Krylov subspace methods, we can obtain new successful methods. We will describe the important multigrid techniques in section 4.2.4 in detail. In section 4.2.5, we also describe non-stationary iterative methods such as Krylov subspace methods [24, 111], as well as introducing preconditioning techniques. In section 4.3 we introduce some iterative solution algorithms for nonlinear systems, which include FAS, Newton's method, Newton-MG and Newton-Krylov [49, 78, 100].

4.2 Iterative Methods for Linear Systems

In this section, we briefly discuss some iterative methods for solving linear systems of equations. We introduce the standard Jacobi and Gauss-Seidel methods in subsection 4.2.2 as a solver. Then, we describe the linear multigrid method in subsection 4.2.4. In the next subsections, we explain some different types of iterative methods, namely the Krylov subspace methods, including the Conjugate Gradient method (CG) in subsection 4.2.5.2 and the Generalized Minimal RESidual method (GMRES) in subsection 4.2.5.3.

4.2.1 Introduction

To solve systems of linear equations, we can use alternative solvers from direct methods based upon iterative methods. The earliest iterative methods used for solving large linear systems are the relaxation methods. These methods are based upon providing an approximate solution and then systematically modifying the components of the approximation in a particular order until the convergence is reached [111]. There are numerous such iterative methods available to solve large sparse linear algebraic systems that arise from the discretisation of elliptic PDEs. For further information, we refer the reader to the broad literature on this field (e.g. [5, 18, 32, 47, 55, 72, 77, 111] and references therein).

In this section, we will present some iterative methods that can be used to solve large sparse linear systems that will appear later in this thesis. However, the primary interest of this thesis is the computation of a solution of nonlinear PDEs, and an important step towards that goal is to compute the solutions of linear systems which arise as a key ingredient in, for instance, Newton-type iterative algorithms. Discretisation of a linear differential equation or of a linearised nonlinear differential equation (through spatial discretisation techniques such as finite elements or finite differences) leads to a finite dimensional system given by Equation (4.1).

In sections 4.2.2 and 4.2.3, we introduced Jacobi and Gauss-Seidel methods, these typically require the matrix A to be strictly diagonally dominant or an irreducibly diagonally dominant to guarantee convergence. In section 4.2.4, where multigrid is introduced, it is assumed that A matrix is the result of either direct discretization linear elliptic operator or the linearization of discretization of the nonlinear elliptic operator. Finally, in section 4.2.5, Krylov subspace methods that introduced for both general nonsymmetric (GMRES) and symmetric positive definite (CG) matrices. In the following subsection, we introduce the important linear Jacobi and Gauss-Seidel iteration methods.

4.2.2 Jacobi and Gauss-Seidel Methods

We start by introducing *stationary iterative (relaxation) methods* for solving linear systems, specifically the *Jacobi, weighted Jacobi, Gauss-Seidel and Red-Black-Gauss-Seidel methods.*

Following [28] and [130], we begin by describing these methods as a solver represented by the matrix A from Equation (4.1) in split form as:

$$A = D - L - U, \tag{4.2}$$

where D is the diagonal of A, and L and U are strictly lower and upper triangular matrices respectively.

Let

$$R_J = D^{-1}\,(L + U). \tag{4.3}$$

Then *Jacobi* iteration in matrix form is given by

$$u^{k+1} = R_J\,u^k + D^{-1}\,b, \tag{4.4}$$

where u^k and u^{k+1} are the approximations of the solution u in iterations $k = 0, 1, 2, ..., n$. In this method, we only use the values from the previous iteration which means that we cannot use the components of the new approximation as soon as they are updated.

Let

$$R_\omega = (1 - \omega)I + \omega\,R_J. \tag{4.5}$$

Then *weighted or (damped) Jacobi* iterations in matrix form are given by:

$$u^{k+1} = R_\omega\,u^k + \omega D^{-1}\,b, \tag{4.6}$$

where $\omega \in \mathbb{R}$ is the weighting factor that is usually chosen as $\omega \in (0, 1]$. If $\omega = 1$ we yield the original Jacobi iterations. We will perform experiments with different values of the ω parameter in the following chapter, where this choice can be important with the algorithms used in this thesis.

Now we define the *Gauss-Seidel* method which is an effective alternative method to the Jacobi method. This method can use the components of the new approximations as soon as they are computed, which means that we can improve the iterations by using the most up-to-date values of the approximation. Starting from Equation (4.2) we let

$$R_G = (D - L)^{-1}U, \tag{4.7}$$

and then the *Gauss-Seidel* iterations are given by:

$$u^{k+1} = R_G\,u^k + (D - L)^{-1}\,b. \tag{4.8}$$

We define the *Successive Over-Relaxation* (SOR) method, which is the *weighted-Gauss-Seidel* method, as follows. We choose $\omega > 0$ and let

$$R_{SOR} = (D - \omega L)^{-1}. \tag{4.9}$$

In matrix form the iterations are given by:

$$u^{k+1} = R_{SOR}\left[(1-\omega)\, D + \omega\, U\right] u^k + R_{SOR}\, \omega\, b, \tag{4.10}$$

which reduces to the *Gauss-Seidel* method, when $\omega = 1$. When a different weighting parameter $\omega \in (0, 2)$ is used this method is also known as *Successive Over-Relaxation* (SOR) method. The convergence (or not) of these iterations relies on the properties of the matrix A in the linear operator Equation (4.1) (cf [8, 61, 67, 77, 111]).

4.2.3 Red-Black-Gauss-Seidel

The strategy of the *Red-Black-Gauss-Seidel* (RB-GS) iteration is concerned with the order of the unknowns. The advantage of this method (the Red-Black ordering or odd-even ordering) is not immediately apparent, however, for the Gauss-Seidel method, the order in which the variables are updated is relevant, particularly if there is an underlying spatial discretisation, such as FDM. Each red node only has black neighbours in all directions; furthermore, each black node only has red nodes in all directions. If we are doing Gauss-Seidel with this ordering, then, we will update the solution at all the red nodes first, followed by all the black nodes. Because no red node depends on data from any other red node, the red nodes can be treated in any order without changing results; they just have to be treated before the black nodes. Similarly, the black nodes can be treated in any order, followed by the red nodes. This makes Red-Black-Gauss-Seidel method popular for parallel computer implementation as it reduces communication overheads. Our implementations are always sequential, however, we still apply the Gauss-Seidel method with this order (Red-Black) in this thesis as it turns out to perform within multigrid in an efficient manner, as we will see in the later chapters.

For example we illustrate the *Red-Black-Gauss-Seidel* method by considering a simple model as follows [28, 130],

$$-u^{k-1} + 2u^k - u^{k+1} = h^2 b^k, \quad 1 \leqslant k \leqslant n-1$$

$$u^0 = u^n = 0.$$

This system of $n-1$ algebraic equations arises from the finite difference discretisation of a 1D

Laplacian, which we can write in component form as,

$$u^k = \frac{1}{2}(u^{k-1} + u^{k+1} + h^2 b^k),\ k = 1, ..., n-1. \tag{4.11}$$

We present here the *Red-Black-Gauss-Seidel* (RB-GS) iteration, in component form first by using the updating of all even components:

$$u^{2k} = \frac{1}{2}(u^{2k-1} + u^{2k+1} + h^2 b^{2k}),\ k = 1, ..., \frac{n-1}{2}; \tag{4.12}$$

and after that we update all the odd components by

$$u^{2k+1} = \frac{1}{2}(u^{2k} + u^{2k+2} + h^2 b^{2k+1}),\ k = 0, ..., \frac{n-1}{2}. \tag{4.13}$$

Figure 4.1 illustrates the red-black ordering for 1D and 2D models respectively.

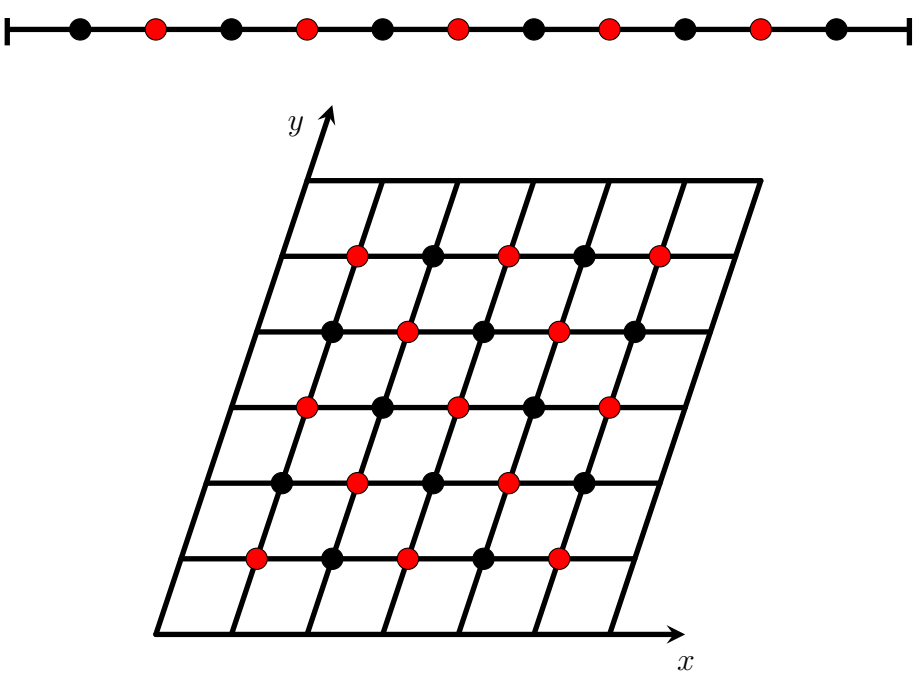


Figure 4.1: A sketch displays on the top a 1D mesh and on the bottom a 2D mesh with the red points (•) and the black points (•) for representation of the Red-Black-Gauss-Seidel method. The red points correspond to points whose (i, j) index sum is even in 1D and 2D.

The iterative methods considered to this point typically converge very slowly. In most cases, the reason behind this is that the high-frequency components of the error are damped very quickly but the low-frequency components very slowly. Such iterative methods have a beneficial feature known as the *smoothing* property. This smoothing property means that it is possible to reduce

the higher frequency error components very quickly within a few iterations. In the next section, we will further discuss the smoothing property, where its use becomes valuable within the context of multigrid methods. It is not generally efficient to use these relaxation methods as solvers for systems arising from elliptic and parabolic PDE discretisations because of their slow convergence. Nevertheless, these methods remain popular as components of multigrid methods because of their smoothing properties (cf [28, 47, 121, 130, 137, 138]).

Consequently, next, we will describe the essential techniques that we will use in this thesis, namely the multilevel algorithms, where we will use these relaxation methods not as solvers but as smoothers.

4.2.4 Multigrid Methods

Southwell described in his 1946 paper [122] the idea of applying multiple grids to reduce overall computational work by solving a system on a coarse grid and after that interpolating the solution to develop the initial guess on a fine grid. Multigrid methods begin from this idea but make greater use of the available grids. In 1977, Brandt published a paper in which he described an effective geometric multigrid method for solving linear and nonlinear boundary value problems. This seminal paper was entitled "Multi-Level adaptive solutions to boundary-value problems" [19]. The fundamental contribution in Brandt's Multigrid method was the use of a series of coarser grids to accelerate the solution on the finest grid. In Brandt's paper, there are three main concepts. Firstly, he demonstrated the use of the Multigrid method on linear boundary value problems using a sequence of coarser grids, known as a V-cycles. Secondly, Brandt shows the use of the Multigrid method with the Full Approximation Scheme (FAS) algorithm for nonlinear boundary value problems. That is, rather than solving the error equation on the coarse grid; he solves a modified version of the whole problem on each grid to account for the nonlinear nature. Ultimately, he incorporated the idea of adaptive mesh refinement, which extends the FAS algorithm with the Multi-Level Adaptive Technique MLAT [19]. Since then multigrid principles have become widely used, not just for the application of a numerical solution of differential equations and optimisation but also, in control theory, computational tomography and particle physics. For more details see [17, 19, 83, 130, 137, 145], for example. Furthermore, there is a notable introductory book [28] explaining the principles of multigrid, which describes in detail linear and nonlinear problems, and how to apply the finite difference multigrid method for solving PDE systems.

In this section, we describe some of the fundamentals of multigrid methods for solving linear systems of PDEs of elliptic type. Multigrid methods are iterative solvers and very efficient for solving the sparse systems of algebraic equations that arise from discrete PDEs. Multigrid methods are termed scalable methods since these methods can solve a linear system with n unknowns with only $O(n)$ computational operations, so-called optimal efficiency. Furthermore,

the multigrid algorithm is directly suitable for both linear and nonlinear systems and can show behaviour that is independent of mesh and the types of boundary conditions.

Multigrid methods use simple iterative methods, such as weighted Jacobi and weighted Gauss-Seidel to reduce a high-frequency component of the error. This process is called *(pre or post) smoothing*, and it can be exploited in the multigrid methods. Therefore, we will use these methods as a smoother throughout this thesis. These classical iterative methods are effective when the error has a high frequency. Consequently, through using a few iterations of the iterative method on a fine grid, we can obtain a large decrease in the highest frequency components of the error. Then, through transferring data to a coarse grid, where the error again is of a higher-frequency (relative to the grid spacing), the iterative method can again quickly reduce the highest frequency components of the error. In other words, once the error is smooth on a grid, it is possible to move to the coarse grid to reduce the lower frequency error components more quickly (since further components of the error appear high-frequency on the coarse grid). When the coarsest grid is eventually reached, we then use a coarse grid solver to get an exact solution on that grid and prolongate (interpolate) these results back to the finest grid to get the final solution. This process overall is much less costly than solving only on the fine grid because the coarser grids have fewer grid points and the convergence rate is accelerated. The overall algorithm is called a Multigrid (MG) method.

There are two broad classes of multigrid methods: one is the Geometric MultiGrid (GMG), and the other is the Algebraic MultiGrid (AMG). The GMG method operates on a hierarchy of defined grids, whereas the AMG method does not need a hierarchy of grids to be produced, which means that instead of using geometric data on a sequence of grids, it uses algebraic data of the matrix associated with the discrete system of equations. Some of the development, analysis and applications of AMG methods are presented in [27, 93, 132, 143]. Furthermore, multigrid methods can be used as preconditioners for other iterative methods [83]. We will consider this in later sections. We continue first with an introduction to the linear GMG method as a solver.

4.2.4.1 Geometric Multigrid (GMG)

Multigrid methods are motivated by a couple of basic considerations: first, many iterative methods have a strong error smoothing effect if they are employed for discrete elliptic (and parabolic) problems, and second, the smooth error can be well expressed on the coarser grid where its approximation is computationally cheaper. In this subsection, we will introduce some common notation and describe the linear GMG method. We first introduce the discrete problems on the fine grid (which we refer to here as grid L, see below) such that

$$A^L \, u^L = b^L, \tag{4.14}$$

Algorithm 1 Linear Geometric Multigrid GMG V-cycle

Function $v^L = \textbf{LMG V-cycle}\,(A^L, v^L, b^L, L)$

1: **Update the solution** v^L **by applying** ν_1 **iterations of the selected smoother on the fine grid (Pre-smoother)**
2: **Calculate the residual**, $r^L = b^L - A^L\, v^L$
3: **Restrict the residual**, $r^{L-1} = I_L^{L-1}(r^L)$
4: **Set the initial guess** e^{L-1} to be 0
5: **if** $(L-1 = Lmin)$ **then** Solve the problem on the coarsest grid $(A^{L-1}e^{L-1} = r^{L-1})$
6: **else** $e^{L-1} = \textbf{LMG V-cycle}\,(A^{L-1}, e^{L-1}, r^{L-1}, L-1)$
7: **Interpolate the error** e^{L-1} **from coarse grid to fine grid** $e^L = I_{L-1}^L(e^{L-1})$
8: **Calculate the correction** $v^L = v^L + e^L$
9: **Update the solution** v^L **by applying** ν_2 **iterations of the selected smoother on the fine grid (Post-smoother)**

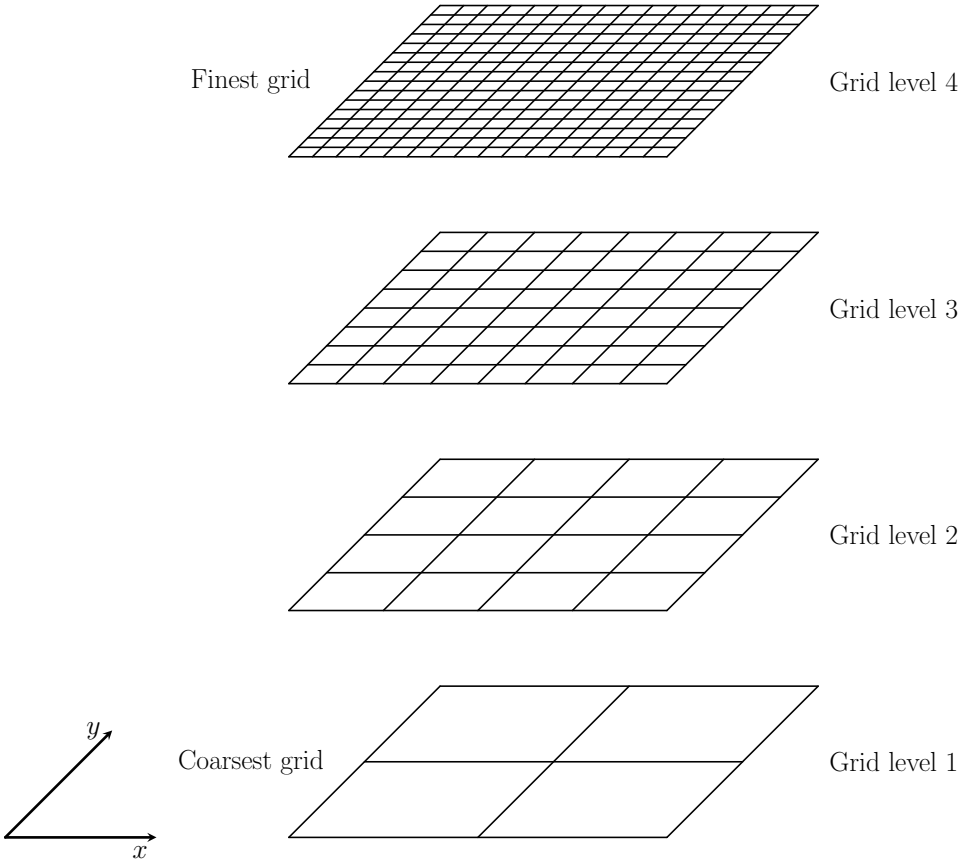


Figure 4.2: An illustration of the geometric multigrid grid hierarchy using four grid levels in two dimensions.

where A^L is the coefficient matrix, u^L is the exact solution vector to the system of equations, b^L is the vector of the right-hand side, and the superscript L indicates the fine grid (or level). We seek v^L which indicates an approximate solution to the system of equations. The error e^L, is the difference between the true solution u^L and the approximate solution v^L, and is given by

$$e^L = u^L - v^L. \tag{4.15}$$

We will then compute the residual (or defect) r^L on the fine grid which is defined as follows:

$$r^L = b^L - A^L v^L, \tag{4.16}$$

and satisfies the error equation as follows,

$$A^L e^L = r^L. \tag{4.17}$$

Importantly, the residual r^L may be computed, however, the error e^L is usually unknown. Note that the relation between the error and the residual in the linear multigrid, can be recognized from:

$$\begin{aligned} A^L e^L &= A^L (u^L - v^L) \\ &= A^L u^L - A^L v^L \\ &= b^L - A^L v^L \\ &= r^L. \end{aligned} \tag{4.18}$$

In a GMG we use a sequence (Ω_L) of finite-dimensional (coarser) grids such that

$$\Omega_{L-1} \subset \Omega_L, \ L = 2, ..., J, \tag{4.19}$$

where Ω_J is the finest grid, on which the solution is desired. Hence, we choose the grids such that they form a hierarchy

$$\Omega_1 \subset \Omega_2 \subset \Omega_3 \subset ... \subset \Omega_J. \tag{4.20}$$

Let V_L represent the sequence of finite-dimensional function spaces which can be represented on grid Ω_L: then

$$V_1 \subset V_2 \subset V_3 \subset ... \subset V_J, \tag{4.21}$$

where each V^L, $L = 1, ..., J$ is defined by grid Ω_L. We consider the discretised system of equations (4.14). To develop a linear multigrid V-cycle recursively, we apply two operators to move functions between the different function spaces. These operators are the *restriction* operator I_L^{L-1}, and *interpolation (or prolongation)* operator I_{L-1}^L, where the subscripts and

superscripts L and $L-1$ indicate the fine and coarse level, respectively:

$$I_L^{L-1}\colon V^L \to V^{L-1}, \qquad L = 2, \ldots, J, \tag{4.22}$$

$$I_{L-1}^{L}\colon V^{L-1} \to V^{L}, \qquad L = 2, \ldots, J. \tag{4.23}$$

We want to solve the problem (4.14) on grid Ω_L, when $L \neq 1$ for the exact discrete solution $u^L \in V^L$. After a number of iterations of linear geometric multigrid has been implemented; we obtain $v^L \in V^L$ which is the approximate solution to the exact solution u^L. We can obtain the error approximation e^{L-1} via the error Equation (4.17) but on a coarser grid. To solve the problem

$$A^{L-1}\, e^{L-1} = r^{L-1} \tag{4.24}$$

is much less expensive than on the fine level L. There are two ways of forming the matrix A^{L-1} which are: to discretize the PDEs on the coarse grid (which is what we are doing in this thesis) or to use Galerkin projection approach as described in [28, 130, 137]. We then compute an updated approximation of v as follows:

$$v^{\text{update}} = v^L + I_{L-1}^{L}\, e^{L-1}.$$

This is an *exact coarse grid correction*, where v^L denotes the approximation to (4.14) on grid Ω_L, and e^{L-1} denotes the exact solution to the residual equation on grid Ω_{L-1}. If there are only two grids in the hierarchy, in this case, the coarse grid problem will be solved exactly (though its interpolation onto the fine grid will not be exactly e^L). The linear geometric multigrid algorithm (Algorithm 1) does not solve the error Equation (4.24) exactly, however, instead it computes an approximate update to the solution on the fine grid by recursively solving the coarse grid equation by the same linear multigrid algorithm. Figure 4.2 displays a hierarchy of four uniform grids in 2D, where the coarsest grid is grid level 1 and the finest grid is grid level 4.

Now we discuss the linear multigrid method in more detail, where the superscripts L and $L-1$ indicate here the fine and coarse level respectively. We must define three operators on the grid as part of our MG algorithm: restriction, interpolation and smoothing. I_L^{L-1} denotes the *restriction* operator. We choose the *full weighting* operator in two-dimensions as a restriction operator from [28] given by

$$\begin{aligned} v_{i,j}^{L-1} = \frac{1}{16}\,[&v_{2i-1,2j-1}^{L} + v_{2i-1,2j+1}^{L} + v_{2i+1,2j-1}^{L} + v_{2i+1,2j+1}^{L} \\ &+ 2\,(v_{2i,2j-1}^{L} + v_{2i,2j+1}^{L} + v_{2i-1,2j}^{L} + v_{2i+1,2j}^{L}) + 4\,v_{2i,2j}^{L}], \end{aligned} \tag{4.25}$$

for $1 \leq i, j \leq \frac{n}{2} - 1$. The *interpolation (or prolongation)* operator I_{L-1}^{L} transfers the error approximation e^{L-1} from the coarse grid to the fine grid, where $I_{L-1}^{L} v^{L-1} = v^L$. We consider

here *the linear interpolation* operator in two-dimensions given by

$$\begin{aligned} v^L_{2i,2j} &= v^{L-1}_{i,j}, \\ v^L_{2i+1,2j} &= \frac{1}{2}(v^{L-1}_{i,j} + v^{L-1}_{i+1,j}) \\ v^L_{2i,2j+1} &= \frac{1}{2}(v^{L-1}_{i,j} + v^{L-1}_{i,j+1}) \\ v^L_{2i+1,2j+1} &= \frac{1}{4}(v^{L-1}_{i,j} + v^{L-1}_{i+1,j} + v^{L-1}_{i,j+1} + v^{L-1}_{i+1,j+1}), \end{aligned} \tag{4.26}$$

where $0 \leq i, j \leq \frac{n}{2} - 1$.
Several standard iterative (relaxation) methods, such as the *weighted Jacobi, the Gauss-Seidel and the weighted Red-Black-Gauss-Seidel methods*, have the smoothing property, which is very important for the multigrid methods as we have mentioned earlier. We can state that iterative methods possess the smoothing property if these methods are effective at reducing high-frequency components of the error. The purpose of smoothing is to allow the numerical resulting iteration error to be approximated well on a coarser grid. According to reference [47], a formal definition of smoothing property is:

$$\| A\,(I - P^{-1}A)^k x \| \leq \eta(k) \;\| x \|_A \quad \text{with } \eta(k) \longrightarrow 0 \text{ as } k \longrightarrow \infty, \tag{4.27}$$

for all vectors $x \in \mathbb{R}^n$, where $\| x \|_A = (x,x)^{\frac{1}{2}}_A = \| x^T A\, x \|_2$, η must be independent of the grid size and P is the preconditioning which is approximate the coefficient matrix A which we will discuss in Subsection 4.2.5.4. In practice, these means that we can obtain a substantial decrease of the high-frequency components of the error by applying a few sweeps of these iterative methods on the fine grid (cf [16, 17, 28, 130]).

We have to use a number ν of smoothing steps before and after the coarse grid correction step; it is not necessary for the number of steps before and after to be the same number. We implement ν_1 steps of an iterative smoothing method on the fine grid (pre-smoothing step), calculate the residual of the current fine grid approximation, and then restrict the residual to the coarse grid with the restriction operator I^{L-1}_L. We then solve the coarse grid residual equation (or approximately solve it using a recursive call). Next, we interpolate the correction applying a prolongation (interpolation) operator I^L_{L-1} for the error to the fine grid, then add the interpolated correction to the current fine grid approximation (coarse grid correction step). We implement ν_2 steps of an iterative method on the fine grid (post-smoothing step). One MG iteration (V-cycle) of the linear geometric multigrid algorithm is described in Algorithm 1. This iteration can be repeated until a suitable norm of the residual is achieved, or a user-defined stopping criterion is satisfied.

Obtaining an optimal efficiency may be problem dependent but it is often the case that these

procedures converge well. Such V-cycle procedures involve exact solutions on the coarsest grid, with the application of a suitable smoother on each fine grid (cf [16, 17, 28, 111, 138, 145]).

One essential merit for multigrid methods is that the number of V-cycles that are required for convergence is independent of the fine mesh level J. Hence, the V-cycle represented in Algorithm 1 can be repeated until a user-defined stopping criterion is met, usually a suitable norm of the residual. In the following subsection, we describe multigrid cycle strategies in more detail.

4.2.4.2 Multigrid Cycle Strategies

There are many cycle strategies for multigrid such as V-cycle, W-cycle and Full-cycle [130,145]. In the previous subsection, we presented a V-cycle approach, which includes one coarse grid correction only at each level of the multigrid cycle. In other words, the multigrid V-cycles start by using the pre-smoother on the finest grid and finish with the post-smoother on the finest grid with only one coarse grid correction within each level of V-cycle. Various other potential strategies for cycling between the different grid levels are also possible. We describe here some different multigrid cycle strategies.

Another strategy is the W-cycle, where after we have applied the smoother at every level we then apply two coarse grid corrections successively. This means that one additional coarse grid correction is required after each new interpolation. The development of the W-cycle increases the number of coarse grid solves used in one cycle, simultaneously with raising the total number of smooths undertaken at each grid level. In some situations, this can make the algorithm converge faster (i.e. in fewer cycles than the V-cycle). However, the W-cycle has additional cost for each cycle. An example of the application of the W-cycle, we refer the reader to [92].

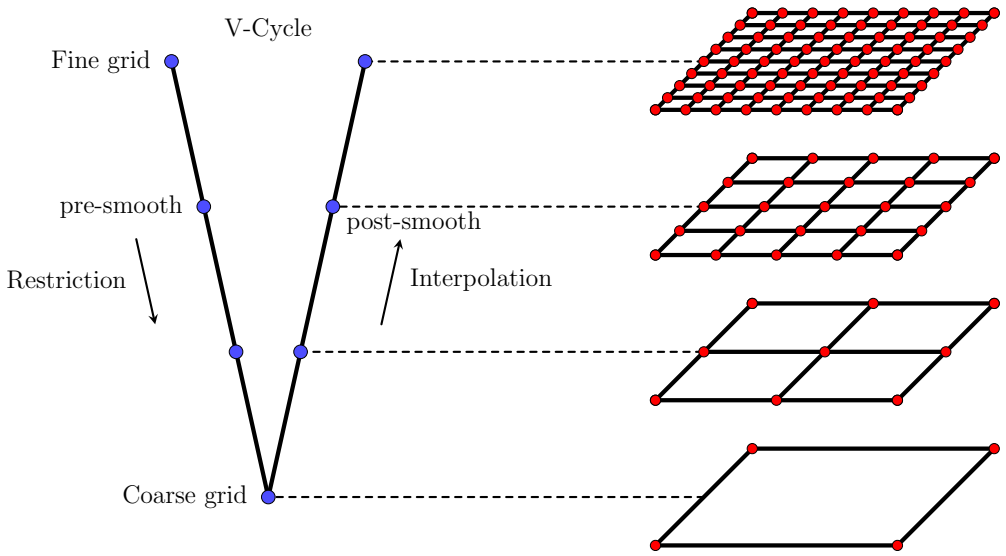


Figure 4.3: A representation of the V-cycle for 2D used in the multigrid iteration of the linear Equation (4.1) using four grid levels [67].

The last multigrid cycle strategy that we will describe here is named the Full-cycle. It is different from the two types described above in that it starts on the coarsest grid and then interpolates the solution to the next finer level. Then a single V-cycle (or W-cycle) is used followed by further interpolation of the solution to the next finer level. On reaching the finest grid a standard cycle is repeated [130,145]. It is worth noting that the Full-cycle is just applied once at the start of the calculation. For more details, the reader is referred to [39] where an application that includes the Full-cycle and the W-cycle can be found. In Figure 4.3, one V-cycle is illustrated for a multigrid method. Figures 4.4 and 4.5 illustrate two V- and W-cycles respectively, whilst Figures 4.6 and 4.7 illustrate the Full-cycle using V- and W-cycles respectively.

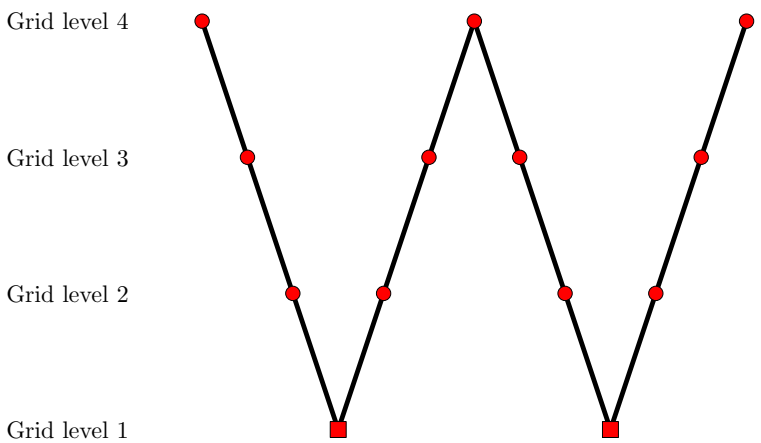


Figure 4.4: Structure of two V-Cycle iterations of the four-grid multigrid method. Within the cycles, • displays the smoothing that is used; ■ indicates the exact solution on the coarsest grid.

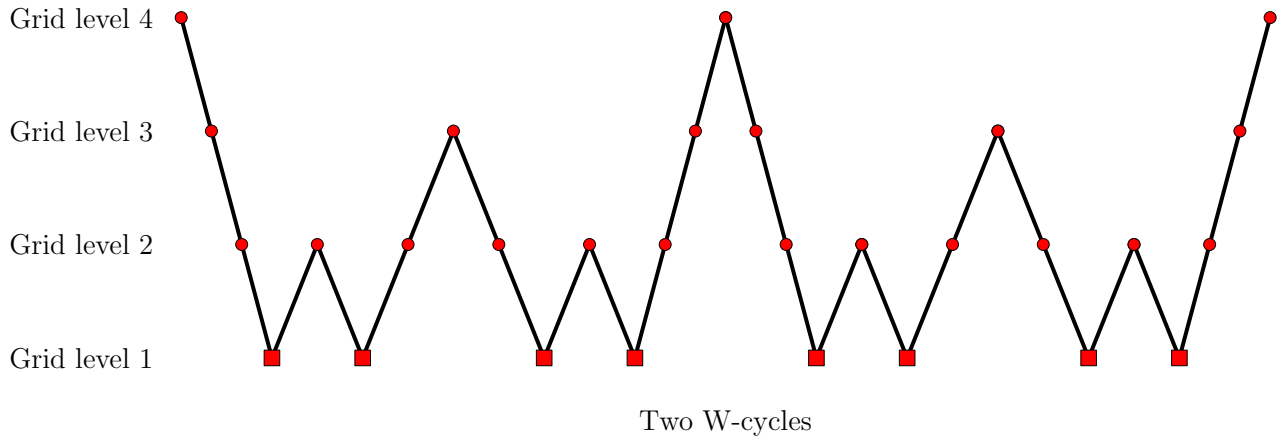


Figure 4.5: Structure of two W-Cycle iterations of the four-grid multigrid method. Within the cycles, • displays the smoothing that is used; ■ indicates the exact solution on the coarsest grid.

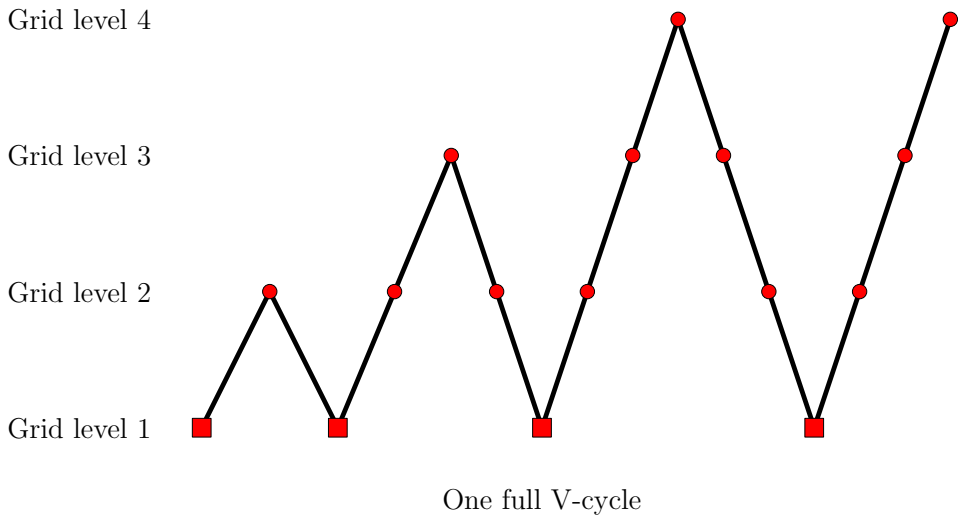


Figure 4.6: Structure of a full V-Cycle iteration for the four-grid multigrid method. Within the cycles, • displays the smoothing that is used; ■ indicates the exact solution on the coarsest grid.

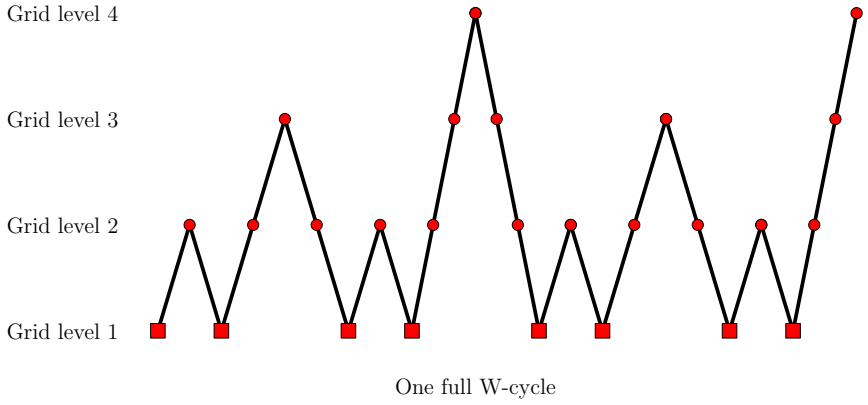


Figure 4.7: Structure of a full W-Cycle iteration for the four-grid multigrid method. Within the cycles, • displays the smoothing that is used; ■ indicates the exact solution on the coarsest grid.

In the following subsection, we will describe the Algebraic Multigrid AMG method.

4.2.4.3 Algebraic Multigrid (AMG)

In subsection 4.2.4.1 we have described the GMG method which uses a grid hierarchy and therefore is suitable, in particular, for sequences of structured grids. In this subsection, we consider a more general approach, namely Algebraic MultiGrid (AMG). This approach is suitable for both structured and unstructured grids and therefore problems with complicated spatial domains. Consequently, we can use this technique when we cannot use the GMG methods, i.e.

when there is no hierarchy of grids available. This method was first introduced in the early 1980s [20–22] since when a lot of studies have been developed. There are many books and papers that have been written on AMG such as [26, 28, 95, 126, 127, 130].

Algebraic Multigrid approaches may also be used in the form of a preconditioner for other iterative methods instead of as a solver in itself. In this thesis, we will use AMG methods as a part of our preconditioner in order to accelerate the convergence of our outer iteration. In particular, we will use this method with a Krylov subspace approach to achieve optimal efficiency [70]. However, we start with a brief introduction to AMG as a solver.

Algebraic multigrid methods generate a coarse grid problem from a fine grid problem in an algebraic way. These methods can, therefore, be efficient in solving large systems derived from discretisation undertaken on unstructured grids, e.g. [20, 28, 48, 93, 128, 131, 132]. Algebraic multigrid requires solely the coefficients in the matrix A of problem Equation (4.1), without explicit data of the problem geometry mesh or discretisation. This method defines inter-grid transfer operators, coarse grids, and coarse grid equations depending only on the matrix coefficients. The coarse grid system is determined by division of the set of unknowns (nodes) into two disjoint sets: the fine and coarse unknowns, based on the properties of the matrix and an appropriate algebraic smoothness condition. Following this, an approximation is formed that only involves the coarse unknowns, along with transfer operators between levels. Once defined, multigrid cycles can be used in a similar way to GMG, but now using only the algebraically defined grid operations. The advantages for AMG, are that it can be used when only the matrix information is provided or when the GMG is hard to apply. Nevertheless, there can be disadvantages for AMG methods: they can require a large computational expense while building the matrix hierarchy and the transfer operators [101]; and they may be less robust than GMG (i.e. the coarsening process may break down).

We have introduced in the previous subsections some of the fundamentals of linear multigrid methods. In the following subsection, we will turn to the Krylov subspace methods, another family of iterative methods, which are also very successful in practice.

4.2.5 Krylov Subspace Methods

In this subsection we will describe some of the different types of *Krylov subspace* methods or *non-stationary iterative* methods. Moreover, we provide a short description of preconditioning techniques.

4.2.5.1 General Technique

Krylov subspace methods describe a large class of methods that have been very successful and powerful for solving linear equations, including, those arising from discretisations of partial differential equations. Krylov subspace methods belong to the family of subspace correction methods [142] and projection methods [111]. In the particular case that an operator is symmetric positive definite, we describe the subspace correction structure. The first use of the Krylov subspace method was in the 1950s [69]. These methods are also very well established in the literature (cf [8, 49, 55, 77, 102, 110, 111, 113] amongst many others).

Krylov methods are formed as a minimisation problem over the span of a set of Krylov basis vectors generated by matrix-vector products of the matrix under consideration. The idea of these methods is that we are working in a high-dimensional space and the goal is to find an appropriate increasing sequence of low-dimensional subspaces in which to find approximate solutions. Krylov subspace methods work by forming a basis from the sequence of successive matrix powers multiplied by the initial residual. The approximations to the solution are then made by minimising the residual over the subspace created by the span of each basis set. There are various types of Krylov subspace methods, such as for solving a sparse symmetric positive definite system, one uses Conjugate Gradient (CG), for solving a sparse symmetric indefinite system, one uses Minimal Residual Method (MINRES), and for a general sparse nonsymmetric system, one uses Generalized Minimal RESidual (GMRES) method [111]. We will discuss two of the most common Krylov subspace methods in the following sections: the CG method and the GMRES method.

Let A be a matrix which we assume to be sparse and nonsingular (e.g. from Equation (4.1)) so that the evaluation of $r^0 \to A\,r^0$ is $O(n)$. Let u_0 be the initial approximation to the solution of problem (4.1). For a given vector b consider the k^{th} Krylov subspace

$$K_k = \mathit{span}\{r^0, A\,r^0, .., A^{k-1}\,r^0\}, \quad k = 1, 2, 3, \ldots \tag{4.28}$$

where $r^0 = b - A\,u_0$ is a nonzero vector. Clearly $K_k \subset K_{k+1}$ and the objective is simply to find for each k the solution $u_k \in K_k$ which best solves the equation $A\,u_k = b$, that is the one that minimises the appropriate norm (for instance the A-norm) $\parallel A\,u_k - b \parallel$, and show that as $k \to n$ the solution u_k tends to the true solution u. Note that although such an approach should yield the exact solution after n iterations (in the absence of rounding errors), in practice we treat Krylov subspace methods as iterative techniques and seek a suitable approximation u_k in fewer iterations.

In practice for general problems, it is unavoidable that the given span of K_k becomes almost linearly dependent for large k. This is highly undesirable when using floating point arithmetic,

therefore an important modification, for an effective implementation of such methods, is to also compute an orthogonal basis for K_k iteratively. This is performed by applying Arnoldi's method, in which some orthogonalisation procedure is executed. Common choices are Gramm-Schmidt, Householder and Lanczos orthogonalisation [111]. In the following sections, Krylov subspace methods for the solution of linear systems are discussed, covering various types of Krylov subspace methods and the preconditioning technique.

4.2.5.2 The Conjugate Gradient (CG) Method

The *Conjugate Gradient (CG)* method is an iterative method and is defined as an orthogonal method on the Krylov subspace. This method was discovered in the 1950s [69], as a direct method. Over the last 40 years, the CG method has developed into a broad application as an iterative method and has generally replaced the family of fixed point iterative methods. We use this method if matrix A in Equation (4.1) is an $n \times n$ *symmetric positive definite (SPD)* matrix only. The CG method is an effective mechanism for solving large sparse symmetric positive definite systems. For further details, the reader is referred to articles such as [69, 77, 111, 119].

There is a relation between the CG method and the Lanczos method for estimating eigenvalues of symmetric matrices [86]. Specifically, like the CG method, the Lanczos method (for symmetric matrices) has the great property that orthogonal basis can be calculated for Krylov subspaces using just a two-term recurrence relation. Therefore, they provide very efficient solution algorithms for linear systems. Features of the connection between conjugate gradient and Lanczos method are discussed in detail in [47, 55, 111].

It is noteworthy that the CG method is very efficient and its iteration can continue without storing a basis for the whole Krylov subspace in this method. In the case of an SPD matrix A, we can reduce the orthogonalisation step to a 2-term recurrence relation. This makes the cost of each CG iteration strictly $O(n)$ when A is sparse. For a general Krylov method (e.g. GMRES) the cost grows with the iteration number k.

4.2.5.3 The Generalized Minimal RESidual (GMRES) Method

The Generalized Minimal RESidual (GMRES) method implements the residual minimization idea by constructing u_k from the least squares method based on the constructed orthonormal basis for K_k which is derived from an Arnoldi-based method. The main issue to be addressed in this approach is to implement this most efficiently via QR decomposition and Hessenberg matrices (cf [8, 112]).

In the GMRES method, all of the k basis vectors that define the subspace must be stored. The orthogonalisation step requires the use of all previous basis vectors. In each iteration of the GMRES method, a matrix-vector product is needed to make a new basis vector. The method

terminates based upon a calculation of the residual (i.e. when the norm of the residual is sufficiently small). Hence, in the GMRES method, as for other iterative techniques, it is not important to form matrix A explicitly: the method only requires the matrix-vector product. From a theoretical point of view, the GMRES algorithm's storage requirements per iteration grow linearly as the iteration progresses. Therefore, in practice, it is important to use restarts based upon a maximum Krylov dimension to control storage requirements. For a detailed description of the algorithm, see [8, 111, 142, 144].

When we distinguish between the CG and GMRES methods, we observe that one advantage of the CG method is that this method obtains the solution without the need to store the basis vectors, as is the situation for GMRES. The CG method is very efficient and attractive computationally. Therefore, the CG method is less memory intensive and is computationally less expensive than GMRES for symmetric positive definite matrices, (cf [8, 55, 111]). The GMRES method is more expensive but can be applied to general non-symmetric and/or indefinite matrices, for which CG will not converge. We have used the GMRES method in this thesis since we deal with non-symmetric matrices in the mathematical models that are solved here. For a discussion regarding an efficient algorithm and implementation of GMRES scheme, the reader is referred to [111].

In the following section, we introduce the main concepts behind preconditioning techniques.

4.2.5.4 Preconditioning

The speed of convergence of the solution methods described above can often be enhanced by a process called preconditioning. In principle, this is the selection of a matrix or an operator P that transforms the equation $A\,u = b$ to one of the standard forms:

$$(P^{-1}A)\,u = P^{-1}\,b$$

which is termed left preconditioning, or

$$(AP^{-1})(Pu) = b$$

which is termed right preconditioning, or

$$(P_1^{-1}\,A\,P_2^{-1})u = P_1^{-1}b$$

which indicates both left and right preconditioning (where $P = P_1\,P_2$). P is chosen such that a faster rate of convergence is obtained, thus reducing the number of iterations required. It is most common to select P by one of two general ways:

– P is an approximate factorisation of A.

– P^{-1} is an approximate inverse of A.

It is often the case that such matrices P can be found by a careful analysis of the problem [10]. An example is given in the case when one is solving the Laplace equation with spatially varying coefficients, and where P corresponds to the discretisation of the Laplacian with constant coefficients where the action of the inverse P^{-1} can be computed efficiently. In these methods, one never computes the full matrix $P^{-1}A$, only vector evaluations of $P^{-1}Au$ are ever computed. The design of a good preconditioner P should be guided by the following conditions:

– The matrix P should be a good approximation to A and computationally cheap to compute.

– The action of the matrix P^{-1} should computationally cheap to compute.

– The eigenvalues of AP^{-1} should be restricted away from zero and infinity and should be clustered into a small number of sets.

The development of preconditioners is a vast, open research area. For the CG method, which applies to an SPD matrix A, the convergence rate is determined by the spectral condition number κ of the matrix A, which is the ratio

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}} \tag{4.29}$$

between the maximum and the minimum eigenvalues of A. The convergence of the CG method relates the error before and after n steps of the CG iteration (in [77,111]) and may be bounded by

$$\|u - u_n\|_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^n \|u - u_0\|_A, \tag{4.30}$$

where the A-norm is defined as follows: $\|u\|_A^2 = (u, u)_A = u^T \, A \, u$. The CG method converges very quickly in the A-norm if the spectral condition number of the symmetric positive definite operator A is approximately equal to one which means that $\kappa = \frac{\lambda_{max}}{\lambda_{min}} \approx 1$. The drawback of CG (and GMRES) iterations is that the convergence rate relies on the spectrum of the operator A. We state that a matrix A is poorly conditioned if the spectrum is not bounded as $n \longrightarrow \infty$ (cf [9, 55, 111, 119]).

For some systems of equations, it is natural to write the linear system Equation (4.1) in block form as follows,

$$A \, u = \begin{pmatrix} \mathbf{K} & \mathbf{D}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{4.31}$$

where $\mathbf{K}$ is $\in \mathbb{R}^n \times \mathbb{R}^n$ and $\mathbf{D}, \mathbf{B} \in \mathbb{R}^n \times \mathbb{R}^m$. This equation, named a saddle point system, can arise in practical applications such as the discretisation of Stokes equation. We can solve the Equation (4.31) iteratively using any iterative methods, especially Krylov subspace methods

such as MINRES (e.g [135]) or GMRES (e.g [112]) which are appropriate in the symmetric (if $\mathbf{D} = \mathbf{B}$) and nonsymmetric (if $\mathbf{D} \neq \mathbf{B}$) problems, respectively. We will consider the nonsymmetric problem in this research. We can precondition the coefficients of matrix A by,

$$P_D = \begin{pmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}, \tag{4.32}$$

where P_D is a block diagonal preconditioner and the Schur Complement is $S = B\,K^{-1}\,D^T$. Equation (4.32) is an example of a block diagonal preconditioner [135]. Another preconditioner for the nonsymmetric problem uses the form as follows,

$$P_T = \begin{pmatrix} \mathbf{K} & \mathbf{D}^T \\ \mathbf{0} & \mathbf{S} \end{pmatrix}, \tag{4.33}$$

where P_T is the block upper triangular preconditioner and the Schur Complement is again $S = B\,K^{-1}\,D^T$. The Krylov method with the P_D block diagonal and the P_T block triangular preconditioning requires only three and two iterations to converge respectively [47,97,135,136]. These preconditioners, with exact Schur complement matrices, are theoretically important as they have a bounded convergence rate independent of the problem size. Nevertheless, the exact Schur complement is impractical to compute due to requiring involves an inverse matrix, making it expensive to compute and requiring large memory and storage. Consequently, practical preconditioners replace S with cheaper sparse approximations. The advantage of these preconditioners is to accelerate the algorithm and to decrease the number of iterations required for convergence while not growing significantly the amount of computational work required for each iteration.

In general, there is no single best preconditioner, however, to obtain a good preconditioner for the problems that we are dealing with we try to remain close to the original problem structure, and use our knowledge of the problem [136]. Overall, preconditioned Krylov-subspace methods represent another class of potentially optimally efficient iterative methods. The final efficiency depends crucially on the efficiency of the preconditioning process.

This concludes the overview of solution algorithms for linear systems and provides a basis for the discussion of solution algorithms for nonlinear systems presented in the next section. We direct the readers for more details of preconditioning techniques and algorithms for preconditioned Krylov methods to read [47,97,111,119,135,136].

4.3 Iterative Methods for Nonlinear Systems

In this section, we consider two classes of methods: nonlinear multigrid methods and variations of Newton's method. The first method presented is the nonlinear multigrid FAS method in subsection 4.3.2. The standard Newton method is introduced in subsection 4.3.3. We then discuss two extensions to the basic Newton's method: the Newton-MG method in subsection 4.3.3.2 and the Newton-Krylov method in subsection 4.3.3.3.

4.3.1 Introduction

This subsection is devoted to a description of nonlinear algebraic systems in order to define the notation used. A typical nonlinear algebraic system of n equations in n variables (e.g. resulting from the discretisation of a nonlinear PDE) (see [5, 24]) can be written as:

$$\begin{aligned} f_1(u_1, u_2, \ldots, u_n) &= 0 \\ f_2(u_1, u_2, \ldots, u_n) &= 0 \\ f_3(u_1, u_2, \ldots, u_n) &= 0 \\ &\vdots \\ f_n(u_1, u_2, \ldots, u_n) &= 0. \end{aligned} \tag{4.34}$$

This nonlinear system can be formulated in vector notation as:

$$\mathbf{F}(\mathbf{u}) = \mathbf{0}, \tag{4.35}$$

where

$$\mathbf{u} = [u_1, u_2, ..., u_n]^T,$$

is a vector of n unknown values that we are seeking and

$$\mathbf{F}(\mathbf{u}) = [f_1(u_1, u_2, ..., u_n), f_2(u_1, u_2, ..., u_n), ..., f_n(u_1, u_2, ..., u_n)]^T, \tag{4.36}$$

is a set of nonlinear equations.

Note that, in general, we do not know in advance if a solution for Equation (4.35) exists, and, if it does, whether it is unique. However, we proceed on the assumption that at least one solution does exist and return to multigrid methods in order to search for such a solution.

4.3.2 Nonlinear Multigrid FAS

The linear multigrid scheme, described in Section 4.2.4, can be generalised to nonlinear systems. In this section, we will be using multigrid methods for solving nonlinear systems of equations

based upon *the Full Approximation Scheme (FAS)*. The first description of FAS was in [19]. Since then, it has been used in many studies and was shown to lead efficiently to the solution of some systems of nonlinear equations (e.g. [16, 17, 145] and many others). FAS seeks to apply the ideas from the linear multigrid iteration and use them directly in a nonlinear multigrid iteration. The general nonlinear Equation (4.35) may also be written in the form:

$$A\,(u) \;=\; b, \tag{4.37}$$

where A is nonlinear in the unknown a, and b does not depend on u, and $u, b \in \mathbb{R}^n$. Given an estimate v of the solution u, we can define the error $e = u - v$, as we have defined earlier for a linear system. We start by explaining that the important difference between linear and nonlinear algebraic systems is in the relationship between the residual and the error. In the standard linear multigrid case, the error equation is $A\,e = r$ and we can determine this error by solving the error equation. Now the residual is $r = b - A(v)$, so when we subtract Equation (4.37) from this residual we obtain,

$$A(u) - A(v) = r. \tag{4.38}$$

In general $A(e) = A(u - v) \neq r$, which means that we cannot solve the simple error equation as we have done in the linear multigrid. Therefore, we must consider Equation (4.38) as a residual equation. We can write Equation (4.37) in the following equivalent form,

$$A(v + e) - A(v) = r, \tag{4.39}$$

i.e.

$$A(v + e) = r + A(v). \tag{4.40}$$

The idea behind the coarse grid correction part of the *Full Approximation Scheme FAS* is to approximate Equation (4.40) on a coarser grid. Consequently, in this scheme the coarse grid problem is solved for the full solution, $v + e$, rather than the error, e, but with a modified right-hand side.

In order to solve the nonlinear problem using FAS, we need to implement (pre or post) nonlinear smoothing as well as the coarse grid correction. We will describe here the nonlinear smoothing first, then we will describe the overall FAS algorithm after that.

We start by smoothing the solution using a few sweeps from a pre-smoother on the fine grid in the FAS algorithm. To do this, we use the *nonlinear Jacobi or nonlinear Gauss-Seidel* iterations which are described in [28]. An approximation solution v^k is updated by using the nonlinear

Jacobi iteration as follows,

$$v_i^{k+1} = v_i^k + \frac{F_i(\mathbf{v}^k)}{J_i(\mathbf{v}^k)},\ i = 1, ..., n, \tag{4.41}$$

where the superscripts $k+1$ and k are the current and previous iterations of the nonlinear Jacobi iteration, respectively, $J_i(\mathbf{v}^k)$ is the first derivative of $F_i(\mathbf{v}^k)$ with respect to v_i^k. For the nonlinear Gauss-Seidel iteration one simply applies the most up-to-date components of v in the evaluation of F_i and J_i on the right-hand side. The smoothing properties of the nonlinear ω-weighted Jacobi and the nonlinear ω- weighted Gauss-Seidel will depend upon the particular nonlinear algebraic discrete system. Consequently, optimal choices for ω cannot be obtained easily but may be estimated after a sequence of numerical experiments. To describe the FAS algorithm, we return to the residual Equation (4.38) and we will use it to obtain the coarse grid correction. We suppose that u^L is the exact solution on grid L,

$$A^L(u^L) = b^L, \tag{4.42}$$

and that v^L is an approximate solution. We now compute the residual on the fine grid as follows:

$$r^L = b^L - A^L(v^L). \tag{4.43}$$

Then, we restrict both the residual r^L and the approximate solution v^L from the fine grid to the coarser grid:

$$r^{L-1} = I_L^{L-1}(r^L), \tag{4.44}$$

$$v^{L-1} = I_L^{L-1}(v^L), \tag{4.45}$$

using the standard restriction operations [28, 130]. Therefore, the Equation (4.40) is approximated on the coarser grid with the modified right-hand side as follows:

$$A^{L-1}(v^{L-1}) = b^{L-1}, \tag{4.46}$$

$$b^{L-1} = r^{L-1} + A^{L-1}(v^{L-1}). \tag{4.47}$$

If Equation (4.46) is solved for v^{L-1} then the error approximation e^{L-1} can be computed on the coarse grid as follows:

$$e^{L-1} = v^{L-1} - v^{L-1}. \tag{4.48}$$

Then we interpolate the error e^{L-1} from the coarse grid to obtain e^L on the fine grid.

$$e^L = I_{L-1}^L(e^{L-1}). \tag{4.49}$$

This gives the correction vector which is applied to the fine grid solution.

$$v^L = v^L + e^L. \tag{4.50}$$

Then we smooth this updated solution with a few sweeps from the post-smoother on the fine grid, see Algorithm 2. The nonlinear multigrid FAS is a popular implementation of a nonlinear multigrid method and has been applied effectively for many systems of nonlinear equations, e.g. [51–53, 62, 74, 96, 109, 116].

We have presented the first of the nonlinear multilevel algorithms, the FAS algorithm, in this subsection. In the following subsection 4.3.3, we will introduce the Newton method which is often used to linearise nonlinear systems, and we will use this as a basis for further algorithms, namely Newton-Multigrid 4.3.3.2 which is a Newton iteration that uses a multigrid solver for each linear solve, and the Newton-Krylov method.

Algorithm 2 Nonlinear multigrid (FAS) algorithm for the solution of the nonlinear system.

Function: $v^L = FASVcycle(v^L, b^L, L)$

1: Update the solution v^L by applying ν_1 iterations of the selected smoother on the fine grid (Pre-smoother)
2: **Calculate the nonlinear residual** $r^L = b^L - A^L\left(v^L\right)$.
3: **Restrict the residual** $r_0^{L-1} = I_L^{L-1} r^L$ on the coarse grid $L-1$.
4: **Restrict the solution** $v_0^{L-1} = I_L^{L-1} v^L$ on the coarse grid $L-1$.
5: **Calculate the modified right-hand side** $b^{L-1} = r_0^{L-1} + A^{L-1}(v_0^{L-1})$ on the coarser grid.
6: **if** $(L-1 = Lmin)$ **then** Solve the problem on the coarsest grid $(A^{L-1}(v^{L-1}) = b^{L-1})$
7: **else** $v^{L-1} = FASVcycle(v_0^{L-1}, b^{L-1}, L-1)$
8: **Calculate the error approximation** e^{L-1} **on the coarse grid** $e^{L-1} = v^{L-1} - v_0^{L-1}$**.**
9: **Interpolate the correction equation from coarse grid to the fine grid** $e^L = I_{L-1}^L e^{L-1}$.
10: **Calculate the correction equation to the fine grid** $v^L = v^L + e^L$**.**
11: Update the solution v^L by applying ν_2 iterations of the selected smoother on the fine grid (Post-smoother)

4.3.3 Newton's Methods

In this section, we will introduce different types of inexact Newton methods as well as two multilevel algorithms to solve the linear algebraic system of equations that arise at each Newton step. In subsection 4.3.3.1, we introduce the Newton method in general. Newton's method provides a global linearisation of nonlinear systems. That means that the result is a linear system at each nonlinear iteration which requires some appropriate iterative linear solver. Therefore, we will present two different types of multilevel algorithms. The first one, presented in subsection 4.3.3.2 is the combination of the Newton method (outer-iteration) with a multigrid method

(inner-iteration) for the linear system. The second algorithm, presented in subsection 4.3.3.3, is the combination of the Newton method (outer-iteration) with a Krylov method (inner-iteration) for the linear system.

4.3.3.1 General Algorithm

Newton's method is very well established and one of the most popular methods for solving nonlinear equations and systems (cf [42,43,49,77,78,100]). In this section, we describe how this method works. Suppose we need to solve the scalar nonlinear equation $F(u) = 0$. By using a Taylor expansion with an initial guess u_0 we can write, [11],

$$F(u_0 + \delta) = F(u_0) + \delta F'(u_0) + \frac{\delta^2}{2} F''(\xi), \tag{4.51}$$

for some ξ between $u_0 + \delta$ and u_0. By neglecting the second order term, assuming that $F(u_0 + \delta) = 0$ (i.e. $u = u_0 + \delta$) and solving for δ we can obtain an improved approximated solution $u_0 + \delta$, where

$$\delta = -F'(u_0)^{-1} F(u_0). \tag{4.52}$$

Now we can extend the Newton method for solving a system of n nonlinear equations with n unknowns straightforwardly from the scalar Newton method. We assume $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ is a differentiable function and that all the partial derivatives of $\mathbf{F}$ with respect to $\mathbf{u}$ exist. The iterative approach converging on a solution assumes that we have some current approximation $\mathbf{u}^k$, $k = 0, 1, 2, ...$ We can obtain the following formula, analogous to (4.52), for an updated approximation $\mathbf{u}^{k+1}$:

$$\mathbf{u}^{k+1} = \mathbf{u}^k - \left(\frac{\partial \mathbf{F}}{\partial \mathbf{u}} (\mathbf{u}^k) \right)^{-1} \mathbf{F}(\mathbf{u}^k). \tag{4.53}$$

In order to solve the nonlinear system (4.35), we are using Equation (4.53) to generate a sequence of approximations $u^1, u^2, ..., u^k$; where $(\frac{\partial \mathbf{F}}{\partial \mathbf{u}}(\mathbf{u}^k))$ is the $n \times n$ Jacobian matrix of partial derivatives,

$$J_{ij} = \frac{\partial F_i}{\partial u_j}, \ i = 1, 2, \ldots, n, \ j = 1, 2, \ldots, n,$$

evaluated at the current solution $\mathbf{u}^k$. We can write this matrix in full as:

$$J(\mathbf{u}^k) = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \frac{\partial f_1}{\partial u_3} & \cdots & \frac{\partial f_1}{\partial u_n} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \frac{\partial f_2}{\partial u_3} & \cdots & \frac{\partial f_2}{\partial u_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \frac{\partial f_n}{\partial u_2} & \frac{\partial f_n}{\partial u_3} & \cdots & \frac{\partial f_n}{\partial u_n} \end{bmatrix}. \tag{4.54}$$

Solving Equation (4.53) requires the inversion of an $n \times n$ matrix at every Newton iteration (in practice the solution of an $n \times n$ linear system at each Newton iteration). For implementing the system (4.53), we divide the iterative method into two steps as follows:

– We find the vector δ by solving the $n \times n$ linear system

$$J(\mathbf{u}^k)\,\delta = -\mathbf{F}(\mathbf{u}^k). \tag{4.55}$$

We can solve the system (4.55) for δ by using a standard numerical method for a system of n linear equations.

– Following this we use the equation

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \delta, \tag{4.56}$$

in order to update the solution $\mathbf{u}^k$. Then we apply Equation (4.55) and Equation (4.56) iteratively until we meet some suitable convergence or failure criteria, for example, until the following convergence condition is satisfied:

$$\|\mathbf{F}(\mathbf{u}^{k+1})\| < Tol, \tag{4.57}$$

where Tol is a user provided value.

The Newton method is extremely popular due to its local quadratic convergence. However, this performance requires a good initial guess and precise convergence properties are problem dependent (see [42, 78, 100]).

There are a wide variety of Newton-type methods, some of which will be reviewed here. In the monograph [42] Deuflhard divided Newton methods into various classes. The first one is that described above, termed the exact Newton method for general nonlinear problems. In this form, the linear system requires the exact numerical solution of the linear equations (4.55). However, the drawback of this method is that at each step we require to solve a system of linear equations by applying a direct method. This can be costly especially when the number of unknowns is large.

Consequently, we will consider a second class of Newton methods: the inexact Newton methods. These methods solve the linear system approximately, or approximate the Jacobian (one can approximate the Jacobian in different ways), or both. For instance, when $\mathbf{F}(\mathbf{u}) = \mathbf{0}$ represents a system of nonlinear equations, which is obtained from the discretisation of a PDE, the Jacobian matrix is often sparse. As a result, sparse iterative solvers may be applied to produce a Newton method that is more efficient applying a dense direct solver. The Newton iteration is defined to be the outer iteration, and the linear iterative steps applied to obtain an approximate update are defined to be the inner iteration. The full scheme then consists of an approximate inner solve at each Newton step k in the inexact Newton method.

There are various types of inexact Newton methods. For example, an inexact Newton method that computes an approximate solution to the Newton equations in a form such that

$$J(u^k)\,\delta^k = -F(u^k) + r^k, \quad k = 0, 1, \ldots, \tag{4.58}$$

$$u^{k+1} = x^k + \delta^k, \tag{4.59}$$

$$\frac{\| r^k \|}{\| F(u^k) \|} \leq \eta_k, \tag{4.60}$$

where r^k is the residual, $\{\eta_k\}$ is the (decreasing) forcing sequence which is used to control the level of accuracy; we assume that this sequence is uniformly less than one. If $\eta_k = 0$, this represents the exact Newton's method. In this method, for example, we solve the linear equation approximately for the Newton step, rather than approximate the Jacobian (cf [41, 42, 46]).

Another type of inexact Newton method is the simplified Newton method [42]. This method is defined by keeping the Jacobian fixed (frozen) during the full iterations:

$$J(u^0)\;\delta^k = -F(u^k), \quad k = 0, 1, \ldots \tag{4.61}$$

$$u^{k+1} = x^k + \delta^k. \tag{4.62}$$

Since we use the old Jacobian approximation the computational expense per iteration is reduced at the potential cost of growing the number of iterations and probably reducing the convergence domain of the Newton iterations.

A further enhancement to Newton's method is the so-called *globalised Newton method*. In this thesis, we apply Newton's method to linearise the nonlinear system of equations that arise from the finite difference discretisation of nonlinear PDEs. We are interested in finding optimal ways to solve these linear systems. For boundary value PDE problems, an initial approximation is often not given, and it may be hard to obtain a good initial approximation. However, if the convergence relies on the quality of the initial solution, it may be that the initial approximation is not close enough to guarantee convergence and consequently, even the exact Newton's method diverges. In this circumstance, a global Newton method may be more suitable. This method is able to compensate for an initial poor approximation to the solution by using damping strategies. As an example, a global Newton method is defined using a damping parameter λ^k, (see [7, 42, 43]).

The global Newton method is given as,

$$J\;\delta^k = -F(u^k), \tag{4.63}$$

$$u^{k+1} = u^k + \lambda^k\,\delta^k \;\;, k = 0, 1, \ldots \tag{4.64}$$

where the parameter $\lambda^k > 0$ is chosen ideally such that

$$\|F(u^{k+1})\| \leq \|F(u^k)\|. \tag{4.65}$$

This cannot be guaranteed, and a separate algorithm for choosing λ^k must be used at each Newton iteration.

We now present a final type of inexact Newton method which are Newton-like methods. In this approach, we replace the Jacobian by some approximation such that

$$M(u)\ \delta^k = -F(u^k),\ \ k = 0, 1, \ldots \tag{4.66}$$

$$u^{k+1} = u^k + \delta^k, \tag{4.67}$$

or replace the Jacobian by some fixed Jacobian $J(v)$ with $v \neq u^0$. For example, we may consider "sparsing" a large Jacobian, e.g. ignoring fill-in during factorization, to allow the use of a direct sparse solve for the Newton-like corrections and thus decrease the work per iteration.

4.3.3.2 Newton-Multigrid

In this subsection, we consider an inexact Newton method combined with a multigrid solver applied to a general nonlinear problem. This method is common in practice, and called Newton-multigrid, this algorithm includes the linear multigrid which is used as a solver for the linear system of equations. We use this algorithm to solve the discrete nonlinear system arising from the discretisation of a nonlinear PDE. We can apply the Newton method which we introduced in Subsection 4.3.3, to linearise the nonlinear system, which requires a linear system to be solved at each Newton step. Hence, we apply the multigrid method for solving the linear system. Therefore, we will use a combination of Newton's method with the linear multigrid method in order to solve the linear system that appears at each Newton iteration. Let

$$F(u) = 0, \tag{4.68}$$

be a nonlinear problem, where F is a set $F = \{F_1, F_2, ..., F_n\}$ of n nonlinear equations and u is a vector $u = \{u_1, u_2, .., u_n\}$ of n unknown values (see [78,100]). In order to solve this system using the *Newton-MG* method, we have to consider two different types of systems and iterations: the nonlinear system that we solve with Newton iterations, which is the outer iteration, and the linear system that we solve with the MG iterations, which is the inner iteration. We start by solving Equation (4.68) using Newton method as follows:

$$J(v)\,\delta = -F(v), \tag{4.69}$$

$$v = v\ + \delta. \tag{4.70}$$

where

$$J(v) = \frac{\partial F(v)}{\partial u},$$

$F(v)$ is the nonlinear residual of the discrete PDE, and v is an approximate solution to the exact solution u. We now solve the system (4.69) using Algorithm 1.

At the next Newton iteration v is changed, therefore the Jacobian has changed on the finest grid, and all of the coarser grids as well. Hence, once we have updated v,

$$v = v + \delta, \tag{4.71}$$

we must restrict this value to the coarse grid as follows:

$$v^{L-1} = I_L^{L-1}(v^L), \tag{4.72}$$

so, as to compute the new Jacobian on each grid. We repeat the whole process at each Newton iteration. In this subsection, we have described how one may apply Newton's method to linearize the nonlinear system of equations that arise from the finite difference discretisation of a nonlinear PDE, and then we show how to apply the MG method to the linear system. For the interested reader, the Newton-MG approach is successful in practice (see [29,60,66,73,82] where one can find that the results give mesh-independent convergence). Now we turn our attention to the last nonlinear multilevel approach in this thesis that uses multilevel principles which is a Newton-Krylov algorithm with a multilevel preconditioner.

4.3.3.3 Newton-Krylov

Newton's method is an attractive method to solve nonlinear systems of PDEs due to its quadratic convergence. When we apply this method for the nonlinear system, we require a linear solver to solve the linear system at each Newton iteration. It is reasonable to solve the linearised system approximately using an iterative method such as Multigrid, as in the Newton-MG algorithm in Subsection 4.3.3.2, or using Krylov methods as in the Newton-Krylov algorithm. In this subsection, we consider the *Newton-Krylov* algorithm.

We have already introduced the overall idea of Krylov methods in Subsection 4.3.3.1. Here we describe the Newton-Krylov method which has the inexact Newton methods as the outer iteration for the nonlinear system and a Krylov subspace solver as the inner iteration for the linear system at each nonlinear iteration. The combination of these two methods that make the Newton-Krylov method one of the most powerful methods to solve nonlinear algebraic systems. The Krylov subspace approach produces a range of iterative linear algebra methods that include the standard conjugate gradient CG method for symmetric positive-definite SPD systems [69,77,111] and methods for nonsymmetric linear systems such as the generalized minimal

residual method (GMRES) [8,112]. Which method can be used is determined by the properties of the linearised system, which in turn depends on the nonlinear system itself. Here we solve with an iterative Krylov method using GMRES since the Jacobian system is nonsymmetric for the problems we consider.

Krylov-subspace methods are commonly used with a preconditioner that accelerates the convergence of the linear iterations. In order to reduce the number of GMRES iterations and speed this algorithm up, several preconditioners have been developed in this research. One important choice of preconditioner is the application of a single V-cycle of multigrid or algebraic multigrid. This can work well for a problem arising from the discretisation of a single nonlinear elliptic PDE but may not be suited to more general problems [16,47,97,136]. An interested reader may find more details in the literature (cf [9, 16, 30, 42, 78, 82, 98] and [144] amongst many others). For nonlinear systems, we can consider block-based preconditioners, where multilevel algorithm may be applicable to part of the preconditioning process. Such preconditioners will be designed and discussed as part of this work.

4.4 Summary

This thesis is essentially concerned with the development and application of nonlinear multilevel approaches to the solution of nonlinear elliptic and parabolic systems of PDEs. Therefore, we have presented three common nonlinear multilevel approaches: the first approach is the nonlinear multilevel algorithm (which is the FAS algorithm), then, a linear multigrid method in combination with Newton's method gives the second approach, and the third multilevel approach is Newton-Krylov with a multilevel preconditioner. In the remainder of this thesis, we consider how to apply these three sophisticated nonlinear multilevel algorithms to nonlinear systems arising from discretising systems of nonlinear PDEs.

All these three approaches can deliver optimal efficiency for certain nonlinear systems as we will see in the later chapters. It is the purpose of this thesis to investigate which is the most efficient. Furthermore, for the Newton-Krylov approach, we will develop novel preconditioners that are suited to problems involving PDE systems.

In Brabazon's thesis and paper [16,17] a comparison was made between the Newton-Multigrid and FAS methods for a single PDE. We extend this work to the comparison between these two methods for solving a nonlinear system of PDEs of equations and add the third nonlinear multilevel method for solving the nonlinear system of PDEs.

For the purposes of this thesis, we are interested in implementing these different nonlinear multilevel algorithms for two examples of nonlinear system of PDEs, which will be done in the

later chapters. In the next chapter, we describe their application to the first model in this thesis, that is the thin film flow system of equations introduced in Sections 5.5 and 5.9. After that, we apply them to the second model, namely the Cahn-Hilliard-Hele-Shaw system of equations introduced in Section 6.3.

Chapter 5

Thin Film Flow System

5.1 Introduction

The focus of this chapter is on developing and contrasting efficient and accurate numerical solvers for systems of nonlinear parabolic partial differential equations: specifically, those modelling time-dependent thin film flows in two dimensions. The cases of steady-state flows and one-dimensional problems are also considered. A further aim of this chapter is to examine the performance of detailed numerical implementations for three different nonlinear multilevel algorithms. We consider these implementations in both steady-state (elliptic) and time-dependent (parabolic) cases, and we make comparisons between them using MATLAB. In all cases, our computational approach uses the finite difference method FDM in space, as we have described in Chapter 3, and implicit time integration with backward differentiation BDF1 in time, which is also described in Chapter 3, for the transient cases.

To solve nonlinear time-dependent problems we must use a nonlinear solver at each time step. Therefore, at each time step the resulting fully discrete nonlinear algebraic system is solved, with the fsolve MATLAB function in the one-dimensional case, or by using three different nonlinear multilevel schemes in the two-dimensional case. The numerical results in 1D and 2D are shown for the computation of steady-state profiles, allowing comparison and validation against previously published work in the one-dimensional case (cf [53,76,145]). We discuss two different mathematical models in the one-dimensional case, as a prelude to the two-dimensional case, selecting the better of these models to then use in two dimensions.

We begin by considering two different mathematical models for the same physical problem

(thin film flow) in the one-dimensional case. We refer to these as Kalliadasis's model, [76], and Sellier's model, [51, 53, 145]. Kalliadasis's model is a third-order PDE given by Equation (2.26), whereas Sellier's model is given by two coupled second-order PDEs (2.32) and (2.34) in the steady-state case. We approximate both models by using the FDM on regular grids in Section 5.2. In the first instance, we solved both discrete problems using the MATLAB fsolve function. This is used to demonstrate the superiority of Sellier's scheme hence only this scheme is considered for subsequent investigation. In these investigations, we contrast three different nonlinear multilevel schemes in 2D. Moreover, we have solved Sellier's model in 2D for both the steady-state and the time-dependent cases with all three of the nonlinear multilevel schemes.

The multilevel schemes considered in this chapter are able to solve a nonlinear algebraic system of equations, arising from discretization of our elliptic and parabolic problems, optimally with a computational expense of $O(N)$ for N unknowns. We will compute the 2D case with each nonlinear multilevel solution method for both steady-state and time-dependent problems and present their results in sections 5.4 and 5.8. Finally, we will discuss and compare computational performance for all cases individually.

5.2 Steady-State Thin Film Flow Solving in 1D

In this section the discretisation methods described in the previous chapter are applied to two mathematical models for the thin film flow in one-dimension:

- The Kalliadasis's model, Equation (2.26).
- The Sellier's model, Equations (2.32) and (2.34).

We are interested in computing the steady-state solution, which means that this solution depends only on the spatial discretisation. In other words, the steady-state solution requires us to set $\frac{\partial h}{\partial t} = 0$ in the mathematical model and obtain an ordinary differential equation in x to be solved.

5.2.1 Kalliadasis's Model

In this subsection, we study the FDM approximation of the solution of the Kalliadasis's model, formulated as a third-order ODE, Equation (2.26) which is described in Chapter 2. We take a particular topography bed shape s as follows,

$$s(x) = D \left[1 + \frac{1}{\pi}\left(-\tan^{-1}\left(\frac{x}{\delta}\right) + \tan^{-1}\left(\frac{x-W}{\delta}\right)\right)\right], \tag{5.1}$$

where x is the Cartesian coordinate. Also, D and W are the trench (or bump) depth and width, respectively, and δ is the parameter that controls the steepness of the trench (or bump) sides

(see Figure 5.1). This equation allows computation of the steady-state one-dimensional thin film flow over a topography of shape $s(x)$. The depth of the film is the dependent variable $h(x)$. The free surface shape is $h(x) + s(x)$. For more details, the interested reader may consult [76] and [53].

The Kalliadasis's model Equation (2.22) may be rearranged to the form of the system of Equations (2.26) as we have described previously in Chapter 2. After that, we discretise by using a finite difference method FDM as we will see in this section.

We define a function $F\colon \mathbb{R}^{3(N+1)} \to \mathbb{R}^{3(N+1)}$ consistent with Equation (2.27) such that the unknowns are grouped together as:

$$U = \begin{pmatrix} u_1 \\ \vdots \\ u_{N+1} \\ v_1 \\ \vdots \\ v_{N+1} \\ w_1 \\ \vdots \\ w_{N+1} \end{pmatrix}. \tag{5.2}$$

We want to create the discrete nonlinear system of equations $F(U) = 0$, by applying the finite difference scheme. We define a uniform grid $x_i = X_1 + (i-1)\Delta x$, for $i = 1, 2, \ldots, N+1$ and $\Delta x = \frac{X_2 - X_1}{N}$, and the domain is $X_1 < x < X_2$ where every point x_i in the grid has 3 unknowns u_i, v_i and w_i. In other words, every point x_i in the grid requires a solution (u_i, v_i, w_i). Then for $j = 1, \ldots, N$ we discretise as follows:

$$\begin{aligned} F_j(U) &= \frac{u_{j+1} - u_j}{\triangle x_j} - \frac{1}{2}(v_j + v_{j+1}), \\ F_{N+j}(U) &= \frac{v_{j+1} - v_j}{\triangle x_j} - \frac{1}{2}(w_j + w_{j+1}), \\ F_{2N+j}(U) &= \frac{w_{j+1} - w_j}{\triangle x_j} - \frac{1}{2}\left[(-s_{xxx})_j + \frac{1 - u_j^3}{3u_j^3} + (-s_{xxx})_{j+1} + \frac{1 - u_{j+1}^3}{3u_{j+1}^3}\right], \end{aligned} \tag{5.3}$$

where

$$(s_{xxx})_{(j)} = \frac{(s_{(j+2)} - 2s_{(j+1)} + s_{(j)}) - (s_{(j)} - 2s_{(j-1)} + s_{(j-2)})}{2(\triangle x)^3}. \tag{5.4}$$

Boundary conditions described in Equation (2.30) are imposed as constraints of the form:

$$\begin{aligned} F_{3N+1}(U) &= (u_1 - 1)^2 + (u_{N+1} - 1)^2 - \lambda = 0, \\ F_{3N+2}(U) &= v_1^2 + v_{N+1}^2 - \lambda = 0, \\ F_{3N+3}(U) &= w_1^2 + w_{N+1}^2 - \lambda = 0. \end{aligned} \tag{5.5}$$

Here λ is a prescribed penalty parameter. In practice, we require a small λ for an accurate solution. Note that this is the exact scheme used in [76] with boundary conditions that Kalliadasis imposed.

Now, we have a system with $3(N+1)$ nonlinear algebraic equations and $3(N+1)$ unknowns (u_j, v_j, w_j) for $j = 1, 2, \ldots, N+1$. In other words, we have a potentially solvable nonlinear system. We can now attempt to solve this system by using the MATLAB function fsolve [78] with a chosen initial guess.

In practice, we require a continuation approach (to obtain convergence) that defines a transformation from a simple case to the desired nonlinear problem. We have used the continuation parameter λ to define the transformation, and we solve a sequence of nonlinear equations using the previous solution as the initial data for the next solution. We started with $\lambda = 1$ and then gradually reduced it. We have also used the parameter δ for a continuation indicating the steepness of the topographical features from Equation (5.1). We started with a large δ (shallow slope) and then gradually reduced it. In practice (see Table 5.1) we reduce λ first until sufficiently small, then reduce δ to produce the designed geometry of the bed. Consequently, we introduced a better initial guess every time and then solved the discrete system for this model by using the MATLAB fsolve function. In the following subsection, we discuss the results obtained using this solution strategy for this model.

5.2.2 Empirical Results and Discussion for Kalliadasis's Model

In this subsection, we present our numerical results for Kalliadasis's model solved using MATLAB's fsolve function. We show nine test situations with trenches of widths 1, 5 and 10 and depth D=1 in Figures 5.1, 5.2 and 5.3, respectively. Different numbers of grid points were used as well. The topography s is defined by Kalliadasis et al. in [76] and afterwards by Gaskell et al. in a similar way in [53] and presented in Equation (5.1).

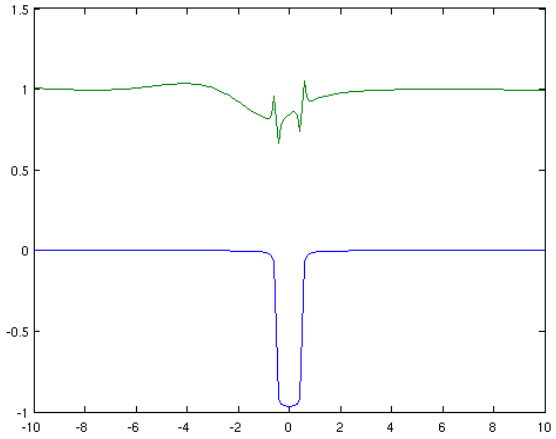
Figures 5.2b and 5.3b display non-physical solutions of the nonlinear system Equations (5.3) and (5.5) which implies that these systems may have multiple solutions. The solutions shown in these figures are solutions of this system but are not the ones of physical interest. Figures 5.1c, 5.2c and 5.3c depict the results of Kalliadasis's model with large values of δ (i.e. a shallow sloping wall). It is clear from these figures that this model can work well with a fine grid for

large values of δ. However, according to the results in Figures 5.1, 5.2 and 5.3 we have also found that Kalliadasis's model does not work well in every situation, which means that Kalliadasiss model does not give satisfactory accuracy in every situation. This may be due to the presence of the third derivative s_{xxx} term in Equation (2.26). This term is not approximated smoothly, and it requires more grid points in order to be accurate. It is worth to note that this model can be accurate but needs increasing numbers of grid points for a small value of δ.

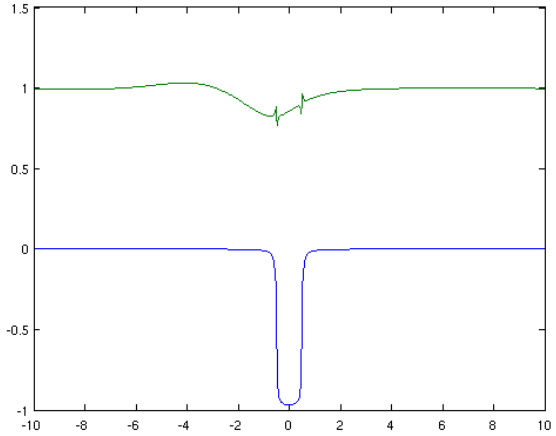
In order to provide further details of the continuation strategy employed to obtain the results in Figures 5.1 to 5.3, we show in Table 5.1 the intermediate cases for solving Kalliadasis's model with 401 mesh points, $D = 1$ and $W = 1$. We solve this model with the starting value of $\lambda = 1$ and $\delta = 0.5$ and then we reduce both. We introduce a better initial guess at each stage through continuation and then solve the discrete system for this model.

Table 5.2 shows the numerical calculations based on Kalliadasis' model with 401 mesh points and different values of λ, δ and $W = 1, 5$ and 10. Each row corresponds to the equivalent row in Table 5.1 and the "iterations" column gives the number of iterations taken by fsolve. Table 5.3 shows the equivalent results of numerical calculations for Kalliadasis's model with 801 mesh points. It is apparent that it is very difficult to get convergence as $\lambda \to 0$ for small δ. Consequently, it can be seen from the data displayed in Table 5.4 that the model gives much better results if $\delta = 0.5$ (and so the bed topography is artificially smooth).

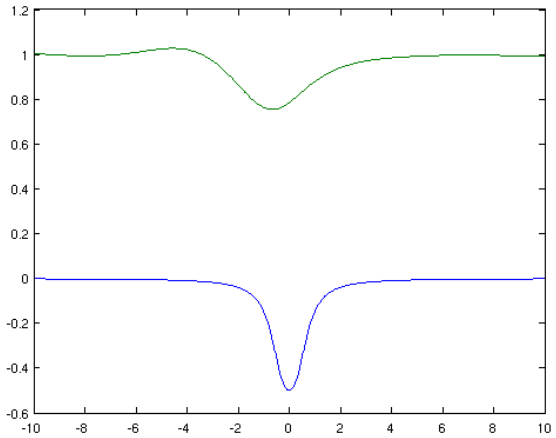
To summarise, the scheme introduced in [76] proved to be expensive, unreliable and also can converge to non-physical solutions as well. Therefore, we now consider an alternative model to solve the same problem more efficiently, without the problematic third derivative terms.



(a) $N = 400$ and $\delta = 0.025$.

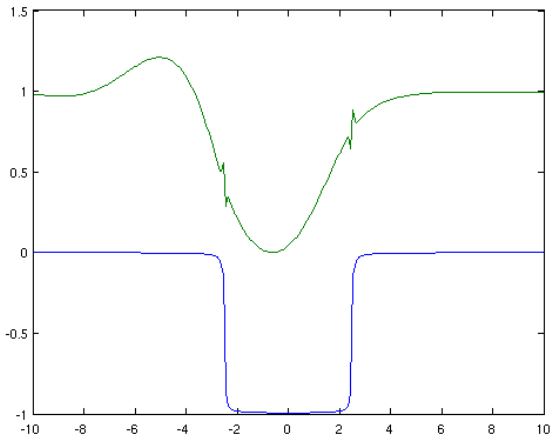


(b) $N = 800$ and $\delta = 0.025$.

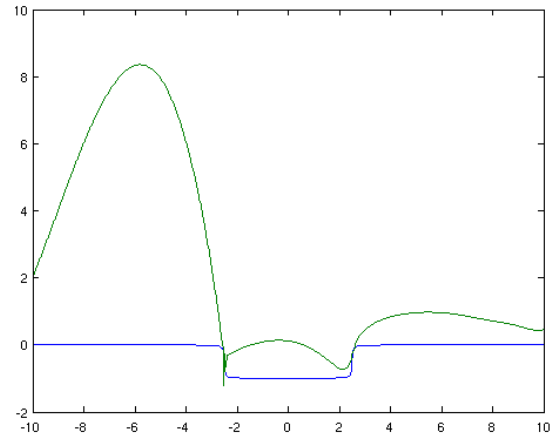


(c) $N = 800$ and $\delta = 0.5$.

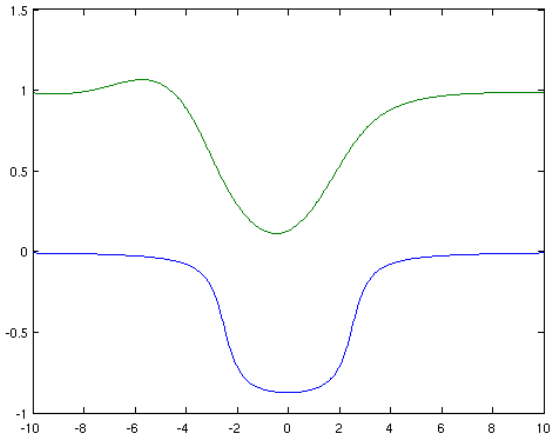
Figure 5.1: The bed shape s is shown in blue (bottom curve), and the numerical solution in green (top curve), for $D = 1$ and $W = 1$. We see that the numerical solution requires large values of N unless the bed shape is artificially smooth.



(a) $N = 400$ and $\delta = 0.025$.

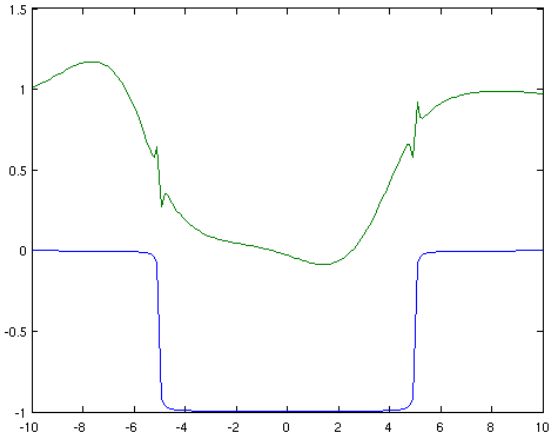


(b) $N = 800$ and $\delta = 0.025$.

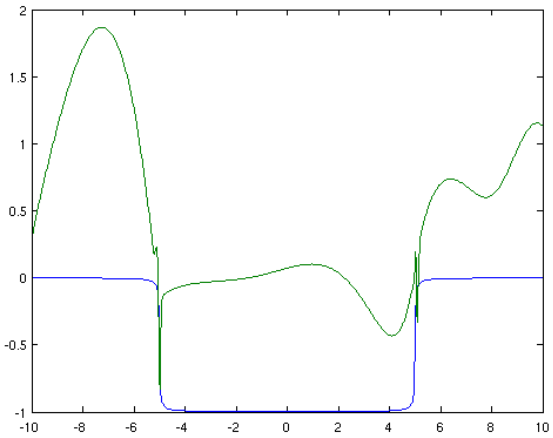


(c) $N = 800$ and $\delta = 0.5$.

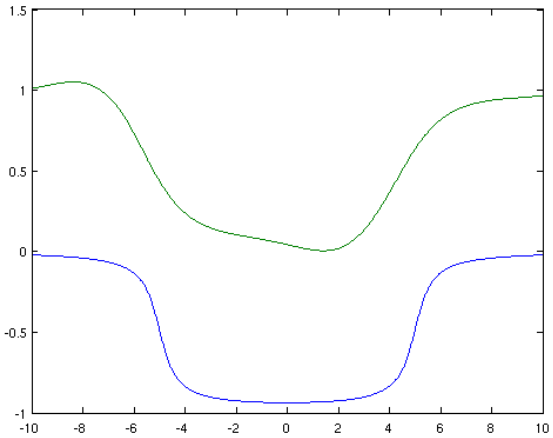
Figure 5.2: The bed shape s is shown in blue (bottom curve) and the numerical solution in green (top curve), for $D = 1$ and $W = 5$. We see that the numerical solution requires large values of N unless the bed shape is artificially smooth.



(a) $N = 400$ and $\delta = 0.025$.



(b) $N = 800$ and $\delta = 0.025$.



(c) $N = 800$ and $\delta = 0.5$.

Figure 5.3: The bed shape s is shown in blue (bottom curve) and the numerical solution in green (top curve), $D = 1$ and $W = 10$. We see that the numerical solution requires large values of N unless the bed shape is artificially smooth.

Table 5.1: The continuation process with values $\lambda = [1, 0.1, 0.01, .001, 0.0001]$ and $\delta = [0.5, 0.4, 0.2, 0.05, 0.025]$.

λ	δ
1	0.5
0.1	0.5
0.01	0.5
0.001	0.5
0.0001	0.5
0.0001	0.4
0.0001	0.2
0.0001	0.05
0.0001	0.025

Table 5.2: Newton iterations for different cases for Kalliadasis's model with 401 mesh points, $D = 1$, $W = 1, 5$ and 10, $\lambda = [1, 0.1, 0.01, 0.001, 0.0001]$ and $\delta = [0.5, 0.4, 0.2, 0.05, 0.025]$ (see Table 5.1).

W=1		W=5		W=10	
Iterations	*Converge*	*Iterations*	*Converge*	*Iterations*	*Converge*
9	Yes	9	Yes	9	Yes
6	Yes	6	Yes	6	Yes
5	Yes	5	Yes	5	Yes
5	Yes	5	Yes	103	Yes
4	Yes	6	Yes	104	Yes
4	Yes	104	Yes	102	Yes
6	Yes	104	Yes	103	Yes
31	Yes	104	Yes	108	Diverge
7	Yes	102	Yes	106	Diverge

Table 5.3: Newton iterations for different cases for Kalliadasis's model with 801 mesh points, $D = 1$, $W = 1, 5$ and 10, $\lambda = [1, 0.1, 0.01, 0.001, 0.0001]$ and $\delta = [0.5, 0.4, 0.2, 0.05, 0.025]$(see Table 5.1).

W=1		W=5		W=10	
Iterations	*Converge*	*Iterations*	*Converge*	*Iterations*	*Converge*
10	Yes	10	Yes	9	Yes
6	Yes	6	Yes	6	Yes
5	Yes	5	Yes	5	Yes
5	Yes	5	Yes	103	Yes
4	Yes	6	Yes	104	Yes
4	Yes	103	Yes	104	Yes
6	Yes	102	stall	112	stall
15	Yes	103	stall	111	stall
11	Yes	109	Diverge	132	Diverge

Table 5.4: Newton iterations for different cases for Kalliadasis's model with 801 mesh points, $W = 1, 5$ and 10, $D = 1$ and δ=[0.5].

λ	W=1		W=5		W=10	
-	*Iterations*	*Converge*	*Iterations*	*Converge*	*Iterations*	*Converge*
1	10	Yes	10	Yes	9	Yes
0.1	6	Yes	6	Yes	6	Yes
0.01	5	Yes	5	Yes	5	Yes
0.001	5	Yes	5	Yes	103	Yes
0.0001	4	Yes	6	Yes	104	Yes

5.2.3 Sellier's Model

Our motivation here is to find the numerical solution for the fully-developed thin film flow model described by Equations (2.31) and (2.32) and to contrast this approach with that of Kalliadasis's, described in the previous subsections. To make this comparison, we begin by considering a simple case, namely, Sellier's model to solve the same problem in 1D [53, 115]. This model is described by two coupled second-order ODEs for $p(x)$ and $h(x)$ where:

$$p - \frac{\partial^2}{\partial x^2}(h + s) = 0, \tag{5.6}$$

$$\frac{\partial}{\partial x}\left(h^3\left(3\frac{\partial p}{\partial x} + 1\right)\right) = 0. \tag{5.7}$$

We will use the FDM approximation to approximate Equations (5.6) and (5.7). This results in a nonlinear system $F(U) = 0$ involving the vector of unknowns U, defined as follows:

$$U = \begin{pmatrix} h_1 \\ \vdots \\ h_{N+1} \\ p_1 \\ \vdots \\ p_{N+1} \end{pmatrix}. \tag{5.8}$$

We considered in 1D the equivalent version of Sellier's model which are Equations (5.6) and (5.7) these are scaled slightly differently from the other version in 2D, as we described early in Chapter 2 in Section 2.2.2. We considered this version due to the fact that this model is consistent with Kalliadasis's model and we can make a direct comparison between these two models in 1D. The version of Sellier's model in 2D that we will carry on with are in Equations (2.31) and (2.32).

We now approximate Equations (5.6) and (5.7) with the FDM for the derivatives in the x direction. In this discretisation, we replace the derivatives by central difference approximations on a uniform grid, where every point x_i in the grid has 2 unknown values p_i and h_i. In other words, every point x_i in the grid requires a solution (p_i, h_i). The resulting discrete problem is as follows:

$$p_i - \left[\frac{h_{i+1} - 2h_i + h_{i-1}}{(\triangle x)^2}\right] - \left[\frac{s_{i+1} - 2s_i + s_{i-1}}{(\triangle x)^2}\right] = 0, \qquad i = 2, \ldots, N, \tag{5.9}$$

$$\frac{1}{\triangle x}\left(\left(\frac{h_i + h_{i+1}}{2}\right)^3\left(3\left(\frac{p_{i+1} - p_i}{\triangle x}\right) + 1\right) - \left(\frac{h_i + h_{i-1}}{2}\right)^3\left(3\left(\frac{p_i - p_{i-1}}{\triangle x}\right) + 1\right)\right) = 0, \qquad i = 2, \ldots, N. \tag{5.10}$$

We define the following nonlinear equations

$$\begin{aligned}
[F_p(U)]_i &= p_i - \left[\frac{h_{i+1} - 2h_i + h_{i-1}}{(\triangle x)^2}\right] - \left[\frac{s_{i+1} - 2s_i + s_{i-1}}{(\triangle x)^2}\right] = 0, \\
[F_h(U)]_i &= \frac{1}{\triangle x}\left(\left(\frac{h_i + h_{i+1}}{2}\right)^3 \left(3\left(\frac{p_{i+1} - p_i}{\triangle x}\right) + 1\right) - \right. \\
&\qquad \left. \left(\frac{h_i + h_{i-1}}{2}\right)^3 \left(3\left(\frac{p_i - p_{i-1}}{\triangle x}\right) + 1\right)\right) = 0, \\
&\qquad\qquad i = 2, \ldots, N.
\end{aligned} \tag{5.11}$$

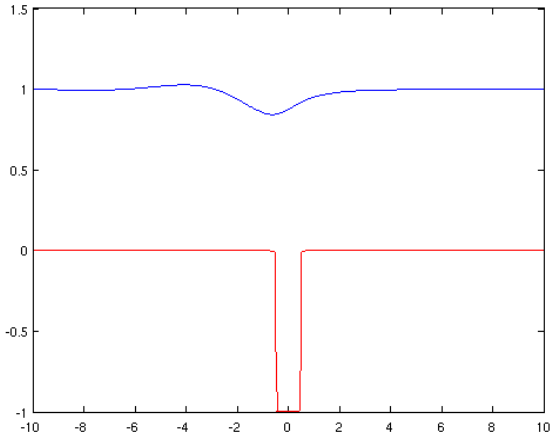
These equations together represent a nonlinear system of $2(N-1)$ equations with $2(N+1)$ unknowns. We set four boundary conditions to complete the system at $i = 1$ and $i = N+1$ as follows,

$$\begin{aligned}
h_1 &= 1, \\
\frac{h_{N+1} - h_N}{\triangle x} &= 0, \\
p_1 = p_{N+1} &= 0.
\end{aligned} \tag{5.12}$$

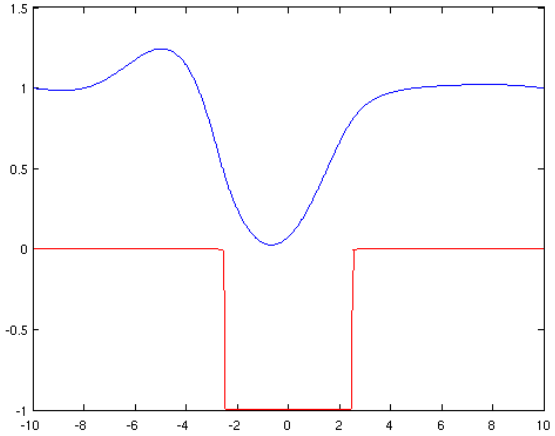
In the following subsection, we will solve this model with the MATLAB fsolve function.

5.2.4 Empirical Results and Discussion of Sellier's Model

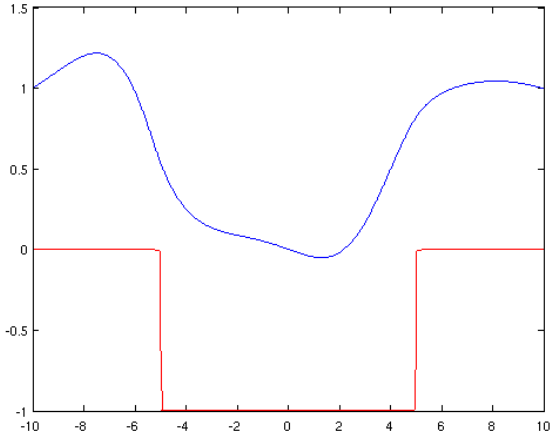
We present the numerical solutions of Sellier's model computed with MATLAB's fsolve function in this subsection. The numerical solutions will use different values of width $W = 1$, 5 and 10. Table 5.5 shows that the numerical solutions converge well for Sellier's model, with a small number of Newton iterations, for various choices of δ and $D = 1$, $W = 1, 5$ and 10. Figure 5.4 shows plots of the numerical solution of Sellier's model with $\delta = 0.001$. The bed shapes s are shown in red (bottom curve), and the numerical solutions are in blue (top curve). We see that the numerical solutions are always smooth with different values for the trench.



(a) The trench with $D = 1$ and $W = 1$.



(b) The trench with $D = 1$ and $W = 5$.



(c) The trench with $D = 1$ and $W = 10$.

Figure 5.4: The bed shape s is shown in red (bottom curve) and the numerical solution with $N = 400$ is in blue (top curve). We see three different cases for the trench when solving Sellier's model.

Table 5.5: Newton iterations using fsolve MATLAB function for Sellier's model with $D = 1$, $W = 1, 5$ and 10 with different values of δ.

δ	W=1		W=5		W=10	
-	*Iterations*	*Converge*	*Iterations*	*Converge*	*Iterations*	*Converge*
0.7	5	Yes	5	Yes	5	Yes
0.5	4	Yes	5	Yes	5	Yes
0.3	4	Yes	4	Yes	4	Yes
0.1	5	Yes	5	Yes	5	Yes
0.05	4	Yes	5	Yes	5	Yes
0.01	4	Yes	4	Yes	4	Yes
0.001	4	Yes	4	Yes	4	Yes

A comparison of Tables 5.2 and 5.5 demonstrates that the convergence of the solutions for Sellier's model is much better than the convergence of the solutions for Kalliadasis's model, with a smaller number of iterations and greater reliability.

5.2.5 Discussion and Comparison

In the previous subsections, numerical experiments with Kalliadasis's and Sellier's models in one-dimension were presented; both used MATLAB's fsolve function in the steady-state case. From these experiments, we see that one of the clear advantages of Sellier's model is that we do not need to use continuation to get a suitable initial guess since it is converging quickly and reliably from the basic initial guess $h_0 = 1$ and $p_0 = 0$. Hence, the nonlinear thin film flow system can be solved far more reliably using Sellier's model rather then Kalliadasis's model.

We have validated our results against existing published results in [76, 145], (see Figures 5.20-5.21 in pages (164-166) in [145], against Figure 5.4 in this chapter). What we have found is that Sellier's model is more robust than the model that Kalliadasis used, therefore, we will only use Sellier's model for the rest of this thesis. In the following section, we will describe the time-dependent discretisation with fixed time step for Sellier's model of thin film flow in 1D.

5.3 Time-Dependent Thin Film Flow Solving in 1D

We have described spatial discretisation with uniform grids for solving Kalliadasis's and Sellier's models in the former subsections; we have also considered the steady-state numerical solution of the nonlinear thin film flow system in 1D for both models. In this section, we will discretise Sellier's model to obtain the time-dependent numerical solution of the nonlinear parabolic PDEs in one-dimension. We will consider the backward Euler BDF1 implicit method with first-order accuracy to approximate the time derivative.

The purpose of this section is to provide a step towards the more challenging 2D problem. Consequently, we introduce the full discretisation in 1D in the next subsection and discuss this in the following subsection. Detailed numerical results are not provided in 1D however as the 2D case, discussed subsequently, is our primary interest.

5.3.1 The Fully Discrete System for the Thin Film Flow Model in 1D

The goal of this subsection is to produce discretisation schemes that can be applied to the one-dimensional mathematical model of Sellier's Equations (2.32) and (2.34) and then be extended to the two-dimensional cases. We have applied the FDM in space and a fully implicit scheme in time for the thin film flow system. At each time step, the resulting discrete nonlinear algebraic system must be solved. We have considered the first-order BDF1 implicit method to approximate the time derivative leading to a fully discrete system in space and time at each time step.

We have explained how the thin film steady-state equations could be discretised in Equations (5.9) and (5.10). In this part, the time derivative will also be discretised using the BDF1 scheme. The time dependence appears only in the term $\frac{\partial h}{\partial t}$. Therefore, we apply to this term the BDF1 scheme, i.e. the backward Euler method in time. It is worth pointing out that we require this temporal discretisation in Equation (2.31) only (and not Equation (2.32)). The fully discrete thin film Equation (2.31) is therefore given as:

$$\begin{aligned}\frac{h_{i,j}^{n+1}-h_{i,j}^{n}}{\triangle t} = \frac{1}{\triangle x}\Bigg[&\frac{1}{3}\left(\frac{h_{i,j}^{n+1}-h_{i+1,j}^{n+1}}{2}\right)^{3}\left(\left(\frac{p_{i+1,j}^{n+1}-p_{i,j}^{n+1}}{\triangle x}\right)-2\right)\\ &-\frac{1}{3}\left(\frac{h_{i,j}^{n+1}-h_{i-1,j}^{n+1}}{2}\right)^{3}\left(\left(\frac{p_{i,j}^{n+1}-p_{i-1,j}^{n+1}}{\triangle x}\right)-2\right)\Bigg],\end{aligned} \tag{5.13}$$

and the pressure Equation (2.35) becomes the implicit form

$$p_{i,j}^{n+1}+\frac{6}{(\triangle x)^{2}}\left[(h_{i+1,j}^{n+1}+s_{i+1,j})-2(h_{i,j}^{n+1}+s_{i,j})+(h_{i-1,j}^{n+1}+s_{i-1,j})\right]=0. \tag{5.14}$$

This is a nonlinear system involving the unknown values of h and p at each grid point (i,j) on the new time level $(n+1)$.

5.3.2 Discussion

In the previous subsection, we have presented the discrete nonlinear algebraic systems that arise from spatial and temporal discretisation schemes for elliptic (and parabolic) PDEs of Sellier's models in one-dimension.

It is worth mentioning that there are several standard implicit schemes for time-dependent systems in the literature such as BDF2 and θ methods; however, for simplicity, we decided to select the BDF1 method for the rest of our work. For the remainder of this chapter, we will consider Sellier's model in two dimensions using FDM in space (and BDF1 scheme in time when relevant), which produces a fully implicit discrete system of the thin film flow model. We will apply more sophisticated approaches than MATLAB's fsolve in order to solve the resulting nonlinear system of equations. Specifically, we consider the nonlinear multigrid Full Approximation Scheme FAS, Newton-MG and Newton-Krylov with a new preconditioner (based around Algebraic Multigrid), which we will refer to as Newton-Krylov-AMG for convenience.

5.4 Steady-State Thin Film Flow Solving in 2D

In this section, we will consider the thin film flow system given by Sellier's model that was presented in the previous section. We shall consider here three different multilevel solution algorithms: nonlinear multigrid FAS, Newton-MG and a Newton-Krylov-AMG solver with a new preconditioner that we have developed, based on the use of an algebraic multigrid. Moreover, we will discuss the implementation of each nonlinear algorithm for solving the thin film flow system to ensure that the implementation is efficient, with sufficient information provided for the reader to recreate an implementation that achieves similar results. We also demonstrate the performance of the nonlinear solvers for the three different multilevel algorithms applied to the steady-state thin film flow system.

In Subsection 5.4.1, we introduce the discrete system for the thin film flow system in the steady-state problem. Then, we present the Jacobian matrix in the steady-state case in Subsection 5.4.2. In the following section, we demonstrate the three different nonlinear multilevel algorithms together with associated empirical results. The first approach is nonlinear MG FAS in Subsection 5.5.1; the second approach is Newton-MG in Subsection 5.5.2, and the third approach is Newton-Krylov-AMG in Subsection 5.5.3.

5.4.1 The Discrete System for Thin Film Flow Model in 2D

We have presented the finite difference discretisation scheme in Chapter 3 in general and in Subsection 3.1.1. We consider this approach to solving the steady-state thin film flow problem in two-dimensions. This discretisation scheme leads to the following nonlinear system of algebraic

equations:

$$\begin{aligned}[F_h(U)]_{i,j} = \frac{1}{\triangle x}\Bigg[&\frac{1}{3}\left(\frac{h_{i,j}+h_{i+1,j}}{2}\right)^3\left(\left(\frac{p_{i+1,j}-p_{i,j}}{\triangle x}\right)-2\right)\\ &-\frac{1}{3}\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3\left(\left(\frac{p_{i,j}-p_{i-1,j}}{\triangle x}\right)-2\right)\Bigg]\\ +\frac{1}{\triangle y}\Bigg[&\frac{1}{3}\left(\frac{h_{i,j}+h_{i,j+1}}{2}\right)^3\frac{p_{i,j+1}-p_{i,j}}{\triangle y}\\ &-\frac{1}{3}\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^3\frac{p_{i,j}-p_{i,j-1}}{\triangle y}\Bigg]=0,\end{aligned} \tag{5.15}$$

and

$$\begin{aligned}[F_p(U)]_{i,j} = p_{i,j} &+ \frac{6}{(\triangle x)^2}[(h_{i+1,j}+s_{i+1,j})-2(h_{i,j}+s_{i,j})+(h_{i-1,j}+s_{i-1,j})]\\ &+\frac{6}{(\triangle y)^2}[(h_{i,j+1}+s_{i,j+1})-2(h_{i,j}+s_{i,j})+(h_{i,j-1}+s_{i,j-1})]=0.\end{aligned} \tag{5.16}$$

Here U is a vector of all of the unknowns $h_{i,j}$ and $p_{i,j}$, which are the approximate thin film height and pressure of gird point (i,j) respectively, where Equation (5.15) is the discretisation of Equation (2.37) and Equation (5.16) is the discretisation of Equation (2.35). Note that these equations hold at interior grid points only ($2 \leq i \leq N$ and $2 \leq j \leq M$, say): boundary conditions are discussed in Appendix A. The exclusion of boundary conditions from the discussion that follows purely to aid clarity.

5.4.2 The Jacobian Matrix in Steady-State Case

Our goal in this subsection is to define the Jacobian of the nonlinear system given by Equations (5.15) and (5.16). The full analytical Jacobian matrix is a sparse matrix; this is because we use a discretisation scheme that is based upon local approximation, i.e. the FDM. Therefore, we present in this subsection the expression of the sparse analytical Jacobian matrix that will be required for all three nonlinear solution algorithms that we use. Alternatively, this Jacobian matrix can be calculated numerically, using finite differences. However, for the FDM discretisation, it is also straightforward and efficient to compute the Jacobian analytically in sparse matrix form.

Let J_S be the analytical Jacobian matrix for the thin film flow system in the steady-state case which is arranged in the following four blocks:

$$J_S = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}, \tag{5.17}$$

or

$$J_S = \begin{bmatrix} \frac{\partial F_p}{\partial h} & \frac{\partial F_p}{\partial p} \\ \frac{\partial F_h}{\partial h} & \frac{\partial F_h}{\partial p} \end{bmatrix}, \tag{5.18}$$

where, referring to Equations (5.15) and (5.16)

$$F = \begin{pmatrix} F_p \\ F_h \end{pmatrix}, \tag{5.19}$$

and U is ordered with the height unknowns followed by the pressure unknowns:

$$U = \begin{pmatrix} h \\ p \end{pmatrix}. \tag{5.20}$$

Now we generate the entries of the matrix (5.18), J_{11}, J_{12}, J_{21} and J_{22}, from Equations (5.15) and (5.16). As noted in subsection 5.4.1, the terms that we present here are for typical interior points of the domain but may also be modified according to the appropriate boundary conditions (see Appendix A). We consider only the non-zero entries of each block, which can be used to build the analytical Jacobian efficiently in a sparse format.

We differentiate Equation (5.16) with respect to $h_{i+1,j}$, $h_{i-1,j}$, $h_{i,j}$, $h_{i,j+1}$ and $h_{i,j-1}$ to obtain the non-zero entries of J_{11} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial h_{i+1,j}} = \frac{6}{(\Delta x)^2}, \tag{5.21}$$

$$\frac{\partial F_{p_{i,j}}}{\partial h_{i-1,j}} = \frac{6}{(\Delta x)^2}, \tag{5.22}$$

$$\frac{\partial F_{p_{i,j}}}{\partial h_{i,j}} = -2\left(\frac{6}{(\Delta x)^2} + \frac{6}{(\Delta y)^2}\right), \tag{5.23}$$

$$\frac{\partial F_{p_{i,j}}}{\partial h_{i,j+1}} = \frac{6}{(\Delta y)^2}, \tag{5.24}$$

$$\frac{\partial F_{p_{i,j}}}{\partial h_{i,j-1}} = \frac{6}{(\Delta y)^2}, \tag{5.25}$$

where $i = 2, ..., N.$, $j = 2, ..., M$.

Now we differentiate Equation (5.16) with respect to $p_{i,j}$ to obtain the non-zero entries of J_{12},

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j}} = 1, \tag{5.26}$$

where $i = 2, ..., N.$, $j = 2, ..., M.$

Now we differentiate Equation (5.15) with respect to $h_{i+1,j}$, $h_{i-1,j}$, $h_{i,j}$, $h_{i,j+1}$ and $h_{i,j-1}$ to obtain the non-zero entries of J_{21} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i+1,j}} = \frac{1}{2(\Delta x)} \left[\left(\frac{h_{i+1,j} + h_{i,j}}{2} \right)^2 \left(\left(\frac{p_{i+1,j} - p_{i,j}}{\Delta x} \right) - 2 \right) \right], \tag{5.27}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i-1,j}} = \frac{-1}{2(\Delta x)} \left[\left(\frac{h_{i,j} + h_{i-1,j}}{2} \right)^2 \left(\left(\frac{p_{i,j} - p_{i-1,j}}{\Delta x} \right) - 2 \right) \right], \tag{5.28}$$

$$\begin{aligned} \frac{\partial F_{h_{i,j}}}{\partial h_{i,j}} = \frac{1}{2(\Delta x)} & \left[\left(\frac{h_{i+1,j} + h_{i,j}}{2} \right)^2 \left(\left(\frac{p_{i+1,j} - p_{i,j}}{\Delta x} \right) - 2 \right) \right. \\ & \left. - \left(\frac{h_{i,j} + h_{i-1,j}}{2} \right)^2 \left(\left(\frac{p_{i,j} - p_{i-1,j}}{\Delta x} \right) - 2 \right) \right] \\ + \frac{1}{2(\Delta y)} & \left[\left(\frac{h_{i,j+1} + h_{i,j}}{2} \right)^2 \left(\frac{p_{i,j+1} - p_{i,j}}{\Delta y} \right) \right. \\ & \left. - \left(\frac{h_{i,j} + h_{i,j-1}}{2} \right)^2 \left(\frac{p_{i,j} - p_{i,j-1}}{\Delta y} \right) \right], \end{aligned} \tag{5.29}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j+1}} = \frac{1}{2(\Delta y)} \left[\left(\frac{h_{i,j+1} + h_{i,j}}{2} \right)^2 \left(\frac{p_{i,j+1} - p_{i,j}}{\Delta y} \right) \right], \tag{5.30}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j-1}} = \frac{-1}{2(\Delta y)} \left[\left(\frac{h_{i,j} + h_{i,j-1}}{2} \right)^2 \left(\frac{p_{i,j} - p_{i,j-1}}{\Delta y} \right) \right], \tag{5.31}$$

where $i = 2, ..., N.$, $j = 2, ..., M.$

Finally, we differentiate Equation (5.15) with respect to $p_{i+1,j}$, $p_{i-1,j}$, $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{22} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i+1,j}} = \frac{1}{3\,(\Delta x)^2} \left(\frac{h_{i+1,j} + h_{i,j}}{2} \right)^3, \tag{5.32}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i-1,j}} = \frac{1}{3\,(\Delta x)^2} \left(\frac{h_{i,j} + h_{i-1,j}}{2} \right)^3, \tag{5.33}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j}} = \frac{1}{3\,(\Delta x)^2}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^3+\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3\right] + \frac{1}{3\,(\Delta y)^2}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^3+\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^3\right], \tag{5.34}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j+1}} = \frac{1}{3\,(\Delta y)^2}\left(\frac{h_{i,j}+h_{i,j+1}}{2}\right)^3, \tag{5.35}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j-1}} = \frac{1}{3\,(\Delta y)^2}\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^3, \tag{5.36}$$

where $i = 2, ..., N.$, $j = 2, ..., M.$

The sparsity pattern of the matrix is the distribution of the non-zero entries in this matrix. The sparse Jacobian matrix for the thin film flow system has the block sparse structure shown in Figure 6.2.

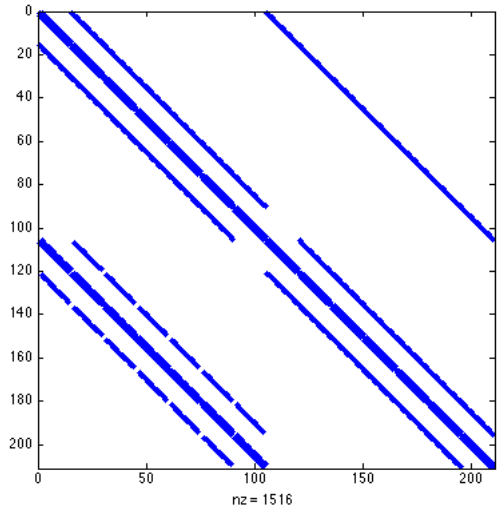


Figure 5.5: The sparse Jacobian matrix for the thin film flow system on a 33×65 grid in the steady-state case in two dimensions.

In the following subsection, we describe the use of the Jacobian matrix in the three different solution methods for the solution of the system Equations (5.15) and (5.16).

5.5 Steady-State Solvers

We introduce the three nonlinear solution techniques: FAS, Newton-MG and the new preconditioned Newton-Krylov-AMG algorithm, in this section. Then, we compare their performance in the steady-state problem in 2D.

5.5.1 FAS

The FAS algorithm is called the full approximation scheme because the coarse grid problem solves for an approximation of the full solution. We have presented this nonlinear multigrid scheme in detail in Algorithm 2 in Chapter 4. This algorithm is one of the primary algorithms that we are interested in applying in this thesis and it is designed to handle the nonlinearity directly, based on a nonlinear smoother and by a coarse grid correction which uses a modified right-hand side. We have selected the most popular transfer operators in the FAS algorithm as mentioned earlier, where we are using a linear interpolation operator and a full-weighting restriction operator. In this subsection, we refer back to the Subsection 4.3.2 in Chapter 4 for the details of this algorithm. While we have described the principle of the nonlinear multigrid FAS algorithm, however, we will explain here how we apply this algorithm to the thin film flow system in two-dimensions with more specific details. In particular, we describe the specific choices of smoother that we have employed.

As we have mentioned before in Chapter 4 the nonlinear multigrid FAS introduces a number of operators including (pre, post) smoother, residual, restriction, right-hand side (RHS) and interpolation (prolongation). In this algorithm, the error is computed on the coarse grid before we interpolate back to the fine grid. There are also different options for each operator and some of them are more expensive than others; for example, for the smoother, among other options, we have a choice between the point-wise nonlinear Jacobi, point-wise nonlinear weighted Jacobi and the point-wise nonlinear Gauss-Seidel Red-Black methods. There are also parameters controlling the number of sweeps that we perform by pre and post-smoothers ν_1 and ν_2, respectively.

In our implementation, we have applied both a point-wise nonlinear Jacobi iteration and a point-wise nonlinear Red-Black-Gauss-Seidel iteration as the smoother. The Red-Black ordering is selected for the Gauss-Seidel smoother because we only need to update the Jacobian evaluation after each red and black sweep, making it much more efficient than other orderings when using the five-point stencil, see [100]. We have used a single Newton step for each iteration of the nonlinear Red-Black-Gauss-Seidel method in our smoother implementation. The FAS algorithm applies a coarse grid correction which is based on solving for the solution on the coarser grid by restricting both the residual r and the approximate solution v to the coarser grids with a modified RHS, and by using an Newton method as the solver on the coarsest grid.

The nonlinear discrete algebraic system of equations of the thin film flow model can be written as follows:

$$F_i(U) = 0,\ i = 1, \ldots, neq, \tag{5.37}$$

where neq is the total number of unknowns in Equations (5.15) and (5.16).

In general, we can define each nonlinear Jacobi iteration as an approximate solve for the i^{th} equation

$$F_i(u_1^k, ..., u_{i-1}^k, u_i^{k+1}, u_{i+1}^k, ..., u_{neq}^k) = 0, \tag{5.38}$$

for $i = 1, \ldots, neq$, where k and $k+1$ denote the current and new approximations. This provides a simple smoother for nonlinear multigrid.

For the nonlinear system considered here, we create a slightly more complex smoother based on a simultaneous update of h and p at each point of the grid. To apply this smoother we use J_{bi} which is the (2×2) block-diagonal system of our analytical Jacobian matrix. At each grid point, we solve the linear system,

$$\underbrace{\begin{pmatrix} \frac{\partial F_{pi}}{\partial h_i} & \frac{\partial F_{pi}}{\partial p_i} \\ \frac{\partial F_{hi}}{\partial h_i} & \frac{\partial F_{hi}}{\partial p_i} \end{pmatrix}}_{J_{bi}} \begin{pmatrix} \delta_{hi} \\ \delta_{pi} \end{pmatrix} = - \begin{pmatrix} F_{pi} \\ F_{hi} \end{pmatrix}, \tag{5.39}$$

where nu is the number of unknown grid points and $i = 1, \ldots, nu$. Then, having solved Equation (5.39) we update h and p at each node $i = 1, \ldots, nu$:

$$\begin{pmatrix} h_i^{k+1} \\ p_i^{k+1} \end{pmatrix} = \begin{pmatrix} h_i^k \\ p_i^k \end{pmatrix} + \begin{pmatrix} \delta_{hi} \\ \delta_{pi} \end{pmatrix}. \tag{5.40}$$

At each step of the smoother we therefore build the (2×2) Jacobian point diagonal block for $i = 1, \ldots, nu$ and solve these (2×2) systems. The updates given by Equations (5.39) and (5.40) may be applied in a Jacobi or a Gauss-Seidel manner.

To summarise, with the nonlinear Newton-Jacobi or the nonlinear Newton-Gauss-Seidel methods as smoothers for the discrete thin film flow model, we update two unknowns h and p associated with one grid point simultaneously with one nonlinear Newton-Jacobi or one nonlinear Newton-Gauss-Seidel iteration. In the nonlinear Newton-Gauss-Seidel method, we used the most up-to-date values of the solution rather than only using the previous solution as in the nonlinear Newton-Jacobi method.

To be more general, we have actually implemented weighted versions of the smoothers: the nonlinear ω-weighted-Jacobi and the nonlinear ω-weighted-Red-Black Gauss-Seidel smoother

with different parameters of ω, see Subsection 4.2.2 in Chapter 4. We will compare these different smoothers with the different parameters ω to select the best smoother with the best parameters for the FAS algorithm on this problem in Subsection 5.7.1.

5.5.2 Newton-MG

In this subsection, we consider the Newton-MG algorithm. Our implementation is a standard application of Newton's method. We applied a global linearisation of the nonlinear problem which means that we linearised the nonlinear problem using the Newton iterations, after which we solve the linear system arising at each Newton iteration by using the linear MG method.

We considered the general form of the discrete nonlinear system arising from application of the FDM to the thin film flow system of equations, (4.37). Again the nonlinear discrete algebraic system of equations is written as Equation (5.37). There are two variables which are h and p, the number of the unknowns is neq in the thin film flow system, and J is the $(neq \times neq)$ sparse analytical Jacobian matrix. After we linearised the nonlinear system using Newton's method, we can write a linear system as follows,

$$\underbrace{\begin{pmatrix} \frac{\partial F_p}{\partial h} & \frac{\partial F_p}{\partial p} \\ \frac{\partial F_h}{\partial h} & \frac{\partial F_h}{\partial p} \end{pmatrix}}_{J} \begin{pmatrix} \delta_h \\ \delta_p \end{pmatrix} = - \begin{pmatrix} F_p \\ F_h \end{pmatrix}, \tag{5.41}$$

$$J\,\delta = -F. \tag{5.42}$$

Then we obtain that

$$\begin{pmatrix} h^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} h^{k} \\ p^{k} \end{pmatrix} + \delta. \tag{5.43}$$

Here J is the sparse Jacobian matrix and the Equation (5.43) gives the update. For the Newton-MG algorithm, a fixed MG V-cycle is used to approximate the solution of Equation (5.42). To implement this we need to calculate the Jacobian matrix on every grid level. At each Newton step we build the sparse $(neq \times neq)$ Jacobian matrix then we solve the Equation (5.42) using linear MG to obtain the correction δ. Once we obtain δ by a linear MG solver (or a fixed V-cycle), we can carry out the update Equation (5.43). We repeat the whole process at each Newton iteration.

In the following subsection, we turn our consideration to the third nonlinear multilevel algorithm which is Newton-Krylov-AMG with a new preconditioner that we have applied to the thin film flow model in the steady-state case.

5.5.3 Newton-Krylov-AMG

This subsection focuses on the nonlinear Newton-Krylov algorithm with a new AMG-based preconditioner. Krylov-subspace methods are commonly used with a preconditioner that accelerates the convergence of the linear iterations; therefore, we consider an implementation of this algorithm to solve the thin film flow model. In this thesis, we develop our algorithm for solving the nonlinear thin film flow system optimally with a new preconditioner. Since Krylov methods converge slowly when employed to this type of a system, we will require a preconditioner in order to reach acceptable convergence rates. As we have mentioned previously the algebraic multigrid AMG iterative methods are the most effective for solving large sparse linear systems that are obtained from the discretisation of single elliptic partial differential equations. Our goal is to incorporate AMG methods as a part of a preconditioner for Krylov subspace methods for our linearised system.

In the Newton-Krylov algorithm, there are two different types of iterations: outer iterations and inner iterations. The outer iterations are the nonlinear Newton's iterations, which we use to linearise the nonlinear system. The inner iterations are the linear Krylov methods for solving linear systems. We have a nonsymmetric matrix J, therefore, we will use GMRES as a Krylov subspace method in our implementation. It is the combination of the two types of iterations that make the Newton-Krylov method, one of the most powerful methods for solving nonlinear algebraic systems, provided a suitable preconditioner is available, as we will confirm in our numerical experiments. In our case, we will use a new preconditioner that incorporates AMG, for which we apply a software implementation that is available in the Harwell Subroutine Library (HSL) [15,70]. This software involves routines called HSL-MI20 for the AMG scheme and HSL-MI24 for the GMRES scheme. Building on this, we will provide a new optimal (or near optimal) preconditioner for Newton-Krylov iterations as a part of the solution of the discrete nonlinear system of PDEs.

We discussed the discrete nonlinear thin film flow system in section 5.4.1; the linearisation of this system leads to a large sparse linear system which must be solved at each step. Let us rewrite the linear system (5.41) in the following matrix form:

$$\begin{pmatrix} \mathbf{K} & \mathbf{I} \\ \mathbf{B}_\alpha & \mathbf{K}_\alpha \end{pmatrix} \begin{pmatrix} \delta_h \\ \delta_p \end{pmatrix} = \begin{pmatrix} F_p \\ F_h \end{pmatrix} \tag{5.44}$$

where $\mathbf{K} = \frac{\partial F_p}{\partial h}$, $\mathbf{I} = \frac{\partial F_p}{\partial p}$, $\mathbf{B}_\alpha = \frac{\partial F_h}{\partial h}$ and $\mathbf{K}_\alpha = \frac{\partial F_h}{\partial p}$ are block matrices.

We require a preconditioner, P say, to approximate the action of the full Jacobian. We examine

the numerical solution of Equation (5.44) with right preconditioning, which requires solution of

$$J\,P^{-1}u = b. \tag{5.45}$$

We do not require the matrix $J\,P^{-1}$ to be formed explicitly, however we do require the preconditioner equation to be solved whenever required, i.e.

$$P\,z = r. \tag{5.46}$$

Let $z = (z_h \;\; z_p)^\top$ and $r = (r_p \;\; r_h)^\top$ be column-vectors so that, for a block upper triangular preconditioner, Equation (5.46) has the following matrix form:

$$P_1\,z = \begin{pmatrix} \mathbf{K} & \mathbf{I} \\ \mathbf{0} & \mathbf{S}_1 \end{pmatrix} \begin{pmatrix} z_h \\ z_p \end{pmatrix} = \begin{pmatrix} r_p \\ r_h \end{pmatrix}, \tag{5.47}$$

where P_1 is our first proposed preconditioner and $S_1 = K_\alpha - B_\alpha\,K^{-1}I$. Here we address the question on how to choose a good preconditioner for solving the linearized thin film flow model at each Newton iteration.

For the first preconditioner P_1 we use the exact Schur Complement, S_1 in the $(2,2)$ block. The benefit of using the exact Schur Complement is that this preconditioned system needs just two iterations to converge [47,97,135]. However, obviously, this is not efficient since we use a lot of memory and great expense to compute S_1. Therefore, we will investigate how to improve this preconditioner by replacing the Schur Complement S_1 by an approximation.

We have made approximations of the preconditioners firstly, via approximate the Schur Complement by replacing it with the first term in the Schur Complement and then we examine this preconditioner via eigenvalues. However, these preconditioners are not practical and efficient. Therefore, we decided to make a further approximation, which is a more efficient approach by using AMG, which is practical and efficient implementation as we will discuss below.

Our approach will be to develop a sequence of potential preconditioners for the matrix J in Equation (5.44). As also described, the first choice of a preconditioner is P_1, the upper triangular block preconditioner written as follows:

$$P_1 = \begin{pmatrix} \mathbf{K} & \mathbf{I} \\ \mathbf{0} & \mathbf{S}_1 \end{pmatrix}, \tag{5.48}$$

where S_1 is the Schur Complement $S_1 = K_\alpha - B_\alpha K^{-1} I$.

The second preconditioner that we consider, P_{1a}, is the upper triangular block preconditioner

formulated as follows:

$$P_{1a} = \begin{pmatrix} \mathbf{K} & \mathbf{I} \\ \mathbf{0} & \mathbf{K}_\alpha \end{pmatrix}, \tag{5.49}$$

where we have replaced S_1 with K_α which we hope will provide an approximation to the Schur Complement.

A third preconditioner, P_{1b}, uses one AMG iteration with this approximate Schur Complement, $AMG(K_\alpha)$, for the $(2,2)$ block and, similarly replace the $(1,1)$ block in P_{1a} with an AMG approximation to K. The advantage of using AMG is in the fact that it is much faster than an exact solve but is spectrally equivalent. Hence,

$$P_{1b} = \begin{pmatrix} \mathbf{AMG(K)} & \mathbf{I} \\ \mathbf{0} & \mathbf{AMG(K}_\alpha\mathbf{)} \end{pmatrix}, \tag{5.50}$$

where $\mathbf{AMG(X)}$ is one AMG iteration for the matrix X.

As with the preconditioners P_1, P_{1a} and P_{1b} we can also introduce a second sequence of potential preconditioners by using forward elimination instead of using the backward elimination (to create block lower triangular preconditioners). These preconditioners are denoted P_2, P_{2a} and P_{2b}. Here P_2 is the lower block triangular preconditioner written as follows:

$$P_2 = \begin{pmatrix} \mathbf{S}_2 & \mathbf{0} \\ \mathbf{B}_\alpha & \mathbf{K}_\alpha \end{pmatrix}, \tag{5.51}$$

where S_2 is the Schur Complement $S_2 = K - IK_\alpha^{-1}B_\alpha$.

The preconditioner P_{2a} is given by the lower block triangular matrix as follows:

$$P_{2a} = \begin{pmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{B}_\alpha & \mathbf{K}_\alpha \end{pmatrix}, \tag{5.52}$$

and the preconditioner P_{2b} is given by the lower block triangular matrix as follows:

$$P_{2b} = \begin{pmatrix} \mathbf{AMG(K)} & \mathbf{0} \\ \mathbf{B}_\alpha & \mathbf{AMG(K}_\alpha\mathbf{)} \end{pmatrix}. \tag{5.53}$$

Note that the preconditioners P_{1a} and P_{2a} approximate the Schur Complement but still solve the backward (or forward) elimination exactly (using backslash in MATLAB in our implementations). The preconditioners P_{1b} and P_{2b} approximate the Schur Complement and the other diagonal block by using one AMG V-cycle. In other words, we applied AMG separately for each diagonal block in the P_{1b} and P_{2b} preconditioners in Equations (5.50) and (5.53) respectively.

We are using HSL-MI20 for the AMG preconditioner, available in Harwell Subroutine Library (HSL) [70]. For comprehensive details of the implementation of the Algebraic Multigrid preconditioning from HSL-MI20, see [1, 15, 112].

The numerical results of the Newton-Krylov-AMG algorithm preconditioned with P_1, P_{1a}, P_{1b}, P_2, P_{2a} and P_{2b} are presented in Subsection 5.7.3.

5.6 Behavior of Eigenvalues

In this subsection, we considered the eigenvalues of each of the preconditioned systems proposed in the previous subsection. Our motivation here is to study and understand the quality of the preconditioners for the GMRES solution. This is because if the eigenvalues of matrix J are widespread, then the GMRES scheme can have slow convergence [136]. Furthermore, if we have a good preconditioner, we should have a good approximation to the original matrix J that clusters the eigenvalues in small groups and, also, these eigenvalues should be bounded away from zero and infinity as the FDM grid is refined. As noted by Greenbaum et al. [57] for any convergence curve we can obtain with GMRES used to a matrix having for any wanted eigenvalues. However, these eigenvalues do not give any guarantee about the convergence of performance but often give a good indicator.

In Table 5.6 and Table 5.7, we present the maximum eigenvalues and the minimum eigenvalues of the coefficient matrix J and the preconditioned matrix JP_{1a}^{-1} and the preconditioned matrix JP_{2a}^{-1} respectively for the first Newton iteration as a sequence of three grids which are the grid level 3, 4, 5 on the grid size 9×17, 17×33 and 33×65 respectively. We observed that the eigenvalues of J are not all real, however, the maximum and the minimum values are only the real part in the complex case. As we can clearly see in these tables, the spread area of the eigenvalues of the original matrix grows with the grid size, whereas the area of the eigenvalues of the preconditioned systems is relatively fixed for all grids sizes, and as we can also see, these preconditioned systems improve the results and make the majority of the eigenvalues cluster in a small area.

From Figure 5.6 to Figure 5.11 we show the eigenvalues of the matrix J in blue and the preconditioned matrix JP_{1a}^{-1} in red, on the grid sizes 9×17, 17×33 and 33×65 respectively. In Figure 5.7, Figure 5.9 and Figure 5.11 we display the eigenvalues of our preconditioned matrix P_{1a} the majority of these eigenvalues are shown to cluster in a small range around one as the grid is refined on the grid size 9×17, 17×33 and 33×65.

From Figure 5.12 until Figure 5.17 we display the eigenvalues of the matrix J in blue and the preconditioned matrix JP_{2a}^{-1} in red, on the grid sizes 9×17, 17×33 and 33×65 respectively. In

Figure 5.13, Figure 5.15 and Figure 5.17, we show the eigenvalues of our preconditioned matrix P_{2a} again, the majority of these eigenvalues cluster in a small range around one on the grid size 9×17, 17×33 and 33×65.

From these figures, we observed that when we used our preconditioning procedures P_{1a} and P_{2a} the majority of the eigenvalues cluster in a small range around one, with just a small number of separated eigenvalues elsewhere in the complex plane. These results imply that our preconditioners are good preconditioners and that K_α and K do indeed provide good approximations to S_1 and S_2 respectively.

Table 5.6: The maximum and minimum eigenvalues (or real part in the complex case) of the coefficient matrix J and the preconditioned matrix JP_{1a}^{-1} where the imaginary parts are unchanged by the preconditioning, as can be seen from the figures 5.6-5.11.

Grid level	Min ($Re\,\lambda(J)$)	Max ($Re\,\lambda(J)$)	Min ($Re\,\lambda(JP_{1a}^{-1})$)	Max ($Re\,\lambda(JP_{1a}^{-1})$)
3	-29.9876	-0.1961	1.0000	1.0000
4	-122.1418	-0.2094	1.0000	1.0000
5	-490.7803	-0.2128	1.0000	1.0000

Figure 5.6: The eigenvalues of the original matrix J on grid size 9×17.

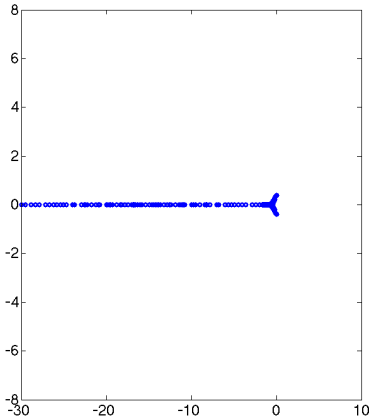


Figure 5.7: The eigenvalues of the upper triangular block preconditioned matrix JP_{1a}^{-1} on grid size 9×17.

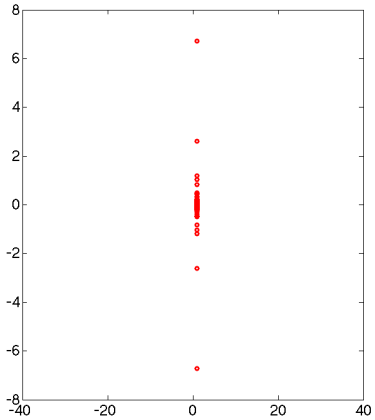


Figure 5.8: The eigenvalues of the original matrix J on grid size 17×33.

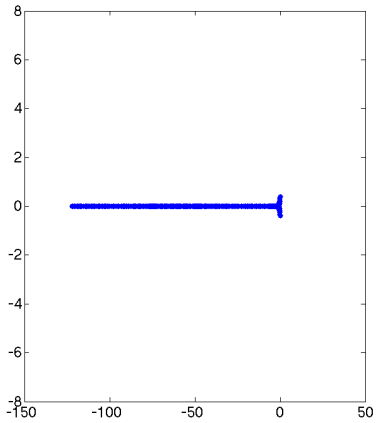


Figure 5.9: The eigenvalues of the upper triangular block preconditioned matrix JP_{1a}^{-1} on grid size 17×33.

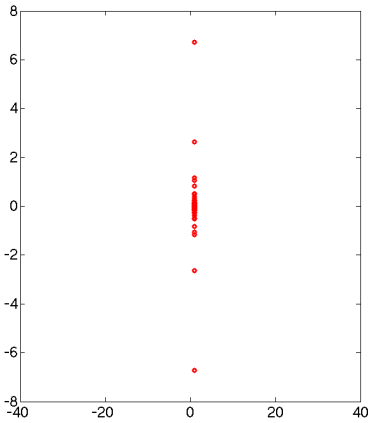


Figure 5.10: The eigenvalues of the original matrix J on grid size 33×65.

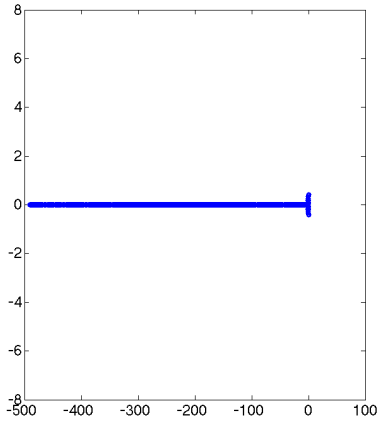


Figure 5.11: The eigenvalues of the upper triangular block preconditioned matrix JP_{1a}^{-1} on grid size 33×65.

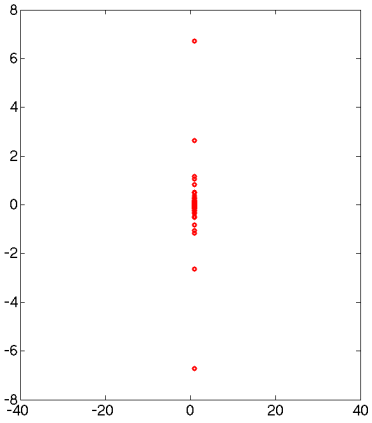


Table 5.7: The maximum and minimum eigenvalues (or real part in the complex case) of the coefficient matrix J and the preconditioned matrix JP_{2a}^{-1} where the imaginary parts are unchanged by the preconditioning, as can be seen from the figures 5.12- 5.17.

Grid level	Min ($Re\,\lambda(J)$)	Max ($Re\,\lambda(J)$)	Min ($Re\,\lambda(JP_{2a}^{-1})$)	Max ($Re\,\lambda(JP_{2a}^{-1})$)
3	-29.9876	-0.1961	1.0000	1.0000
4	-122.1418	-0.2094	1.0000	1.0000
5	-490.7803	-0.2128	1.0000	1.0000

Figure 5.12: The eigenvalues of the original matrix J on grid size 9×17.

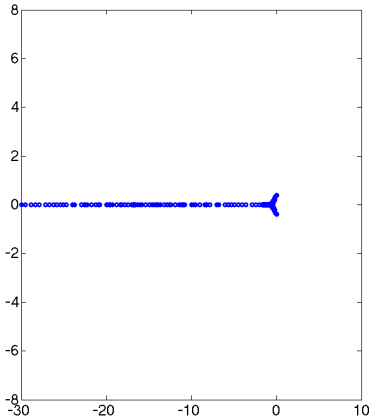


Figure 5.13: The eigenvalues of the lower triangular block preconditioned matrix JP_{2a}^{-1} on grid size 9×17.

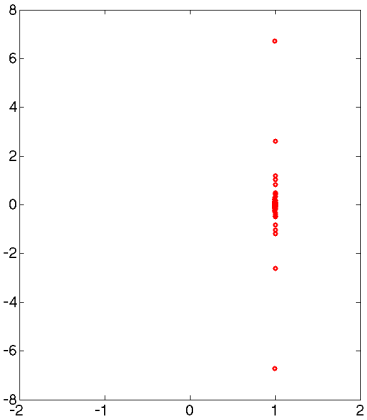


Figure 5.14: The eigenvalues of the original matrix J on grid size 17×33.

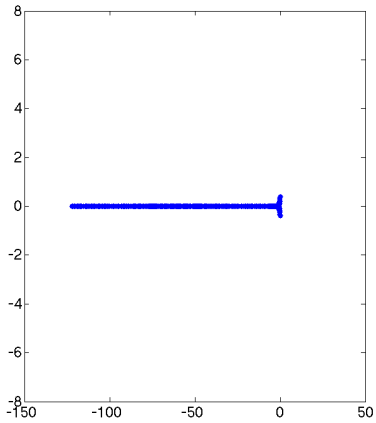


Figure 5.15: The eigenvalues of the lower triangular block preconditioned matrix JP_{2a}^{-1} on grid size 17×33.

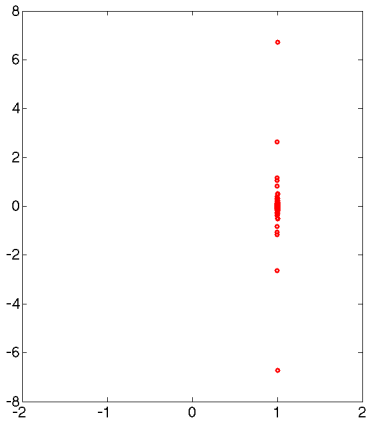


Figure 5.16: The eigenvalues of the original matrix J on grid size 33×65.

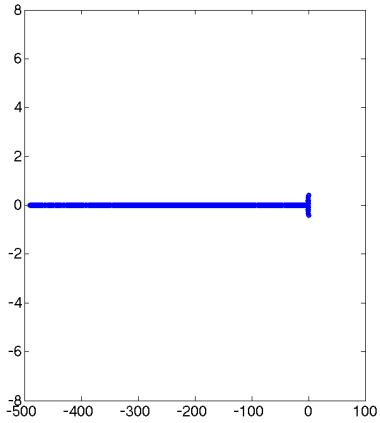
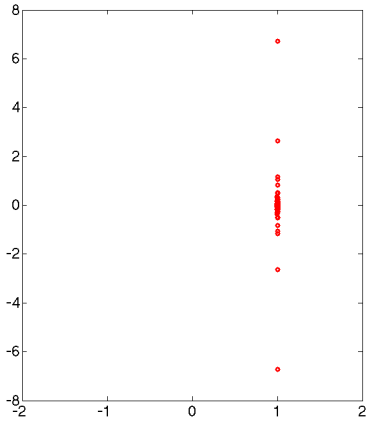


Figure 5.17: The eigenvalues of the lower triangular block preconditioned matrix JP_{2a}^{-1} on grid size 33×65.



In the next subsections, we will present the numerical results of the three nonlinear multilevel algorithms and, also, we will discuss and compare all these algorithms to assess which algorithm is the best.

5.7 Numerical Results in the Steady-State Case

All computational results obtained in this section have been generated using the MATLAB software. We have performed extensive numerical experiments in order to optimize the parameters selected for the FAS, Newton-MG and Newton-Krylov-AMG nonlinear multilevel schemes. Furthermore, in subsection 5.7.4 we will compare our best choice of parameters for these three nonlinear multilevel schemes to determine the best between them.

In our experiments the domain is $X_1 = -10$, $X_2 = 10$, $Y_1 = -5$, $Y_2 = 5$, $dx = (X_2 - X_1)/(mx - 1)$, and $dy = (Y_2 - Y_1)/(my - 1)$. The grid size is $my = 2^{L_{MAX}} + 1$ and $mx = 2 * (my - 1) + 1$ in directions X and Y, where L_{MAX} denotes the maximum grid level. In all these tests we used pure Dirichlet Boundary conditions. The number of unknown grid points is defined as $nu = nx \cdot ny$, where $nx = mx - 2$ and $ny = my - 2$. The number of equations is $neq = 2 \cdot nu$. We defined the Dirichlet boundary condition on the domain $(x, y) \in [X_1, X_2] \times [Y_1, Y_2]$ here as the following; $h = 1$; $p = 0$.
For the topography shape function, we choose a flat bed with an oblong shaped hole, (see Figure 5.18), which is given by:

$$S(x, y) = min(max(d, -1), 0), \tag{5.54}$$

where

$$\begin{aligned} d &= max(dx, dy) \\ dx &= max(x, -x - 4), \\ dy &= max(y - 2, -y - 2). \end{aligned}$$

The numerical results of our experiments in the steady-state case are presented in Subsection 5.7.1 for the FAS algorithm, in Subsection 5.7.2 for the Newton-MG algorithm and in Subsection 5.7.3 for the Newton-Krylov-AMG algorithm. Also, all these three multilevel algorithms optimality are discussed and compared in Subsection 5.7.4.

In order to choose the best parameters for the first two algorithms we employ two different types of a smoother: the weighted Jacobi and the weighted Red-Black Gauss-Seidel smoothers; these smoothers are applied with varying values of the parameter ω. Moreover, in order to compare the convergence of Newton iterations (outer iterations), we use two different values for the nonlinear residual tolerance, Tol $= 1e - 04$ and Tol $= 1e - 08$. For the third solver, inner tolerance values are varied in order to control the convergence of GMRES iterations (inner iterations) and we consider different choices for the restart parameter (Restart) of GMRES iterations and the maximum number of GMRES iterations (Maxit) in each case. We define here the (Restart) to restart the method every restart inner iterations, (Maxit) to be the maximum number of outer iterations which means that the total number of iterations which does not exceed restart. Table 5.8 shows the different sizes for the grid levels that we have used to solve the thin film flow system in the following subsections.

Table 5.8: The grid level, grid size, the number of unknown grid points nu, and the number of equations $neq = 2 \cdot nu$ used in the thin film flow system of equations.

Grid level	Grid size	No. of unknown grid points	No. of equations
3	9×17	7×15	210
4	17×33	15×31	930
5	33×65	31×63	3906
6	65×129	63×127	16002
7	129×257	127×257	64770
8	257×513	257×511	260610
9	513×1025	511×1023	1045506

5.7.1 Numerical Results using FAS

In this subsection, we consider the numerical solutions provided by the nonlinear FAS multigrid solver. We solve a thin film flow test problem depicted in Figure 5.18. We have performed extensive numerical tests to optimize the parameter selection for the FAS method in the steady-state case.

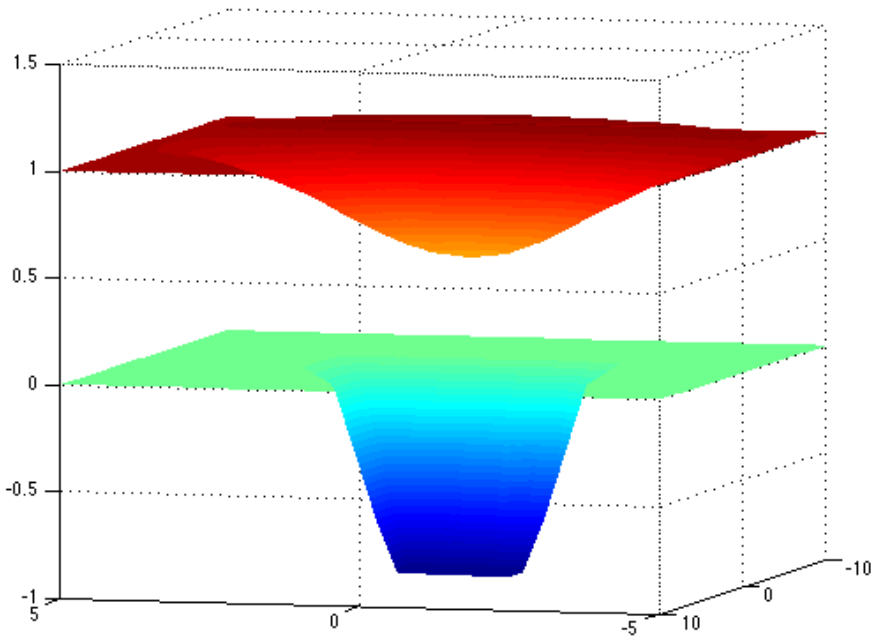


Figure 5.18: Oblong hole bed shape s in blue (bottom) and the free surface (numerical solution) $h + s$ in red (top), where $w = 2, D = 0.5$ and $\delta = 0.05$.

FAS with Jacobi smoother and varying damping factor ω

The first set of results presented illustrate the effect of varying damping factor ω when applying the weighted Jacobi smoother with 1 pre- and 1 post-smooth, a coarse grid of level 3 and a nonlinear tolerance of Tol $= 1e-8$.

In Tables 5.9 to 5.14 we use varying damping factor ω in the range of $\omega = 1/3$, $1/2$, $2/3$, 0.8, 0.9 and $\omega = 1$, with the aim of finding the optimal value for the nonlinear Jacobi smoother with a $(1,1)$ pre- and post-smoother and the coarsest grid level $= 3$. The best choice for the free parameter damping factor ω for the Jacobi smoother is $\omega = 0.8$ since for ω less or greater than 0.8 the performance of FAS is slower and requires more V-cycles to fully converge. Therefore, we will use $\omega = 0.8$ in the FAS algorithm with the Jacobi smoother. Note however that the method becomes optimal (i.e. bounded V-cycles and cost of $O(N)$) in all cases.

Table 5.9: FAS performance for varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 1/3$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	3	1	1	47	2.0224
6	3	1	1	49	5.1757
7	3	1	1	49	18.8803
8	3	1	1	49	74.4457
9	3	1	1	50	307.5922

Table 5.10: FAS performance for varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 1/2$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	3	1	1	30	1.7213
6	3	1	1	31	3.2970
7	3	1	1	32	12.3587
8	3	1	1	32	48.8317
9	3	1	1	32	198.3339

Table 5.11: FAS performance for varying grid size for the thin film flow the in steady-state case with Jacobi $\omega = 2/3$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	3	1	1	22	1.2078
6	3	1	1	22	2.4214
7	3	1	1	23	9.2648
8	3	1	1	23	36.6635
9	3	1	1	23	148.9333

Table 5.12: FAS performance for varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	3	1	1	18	0.6510
6	3	1	1	18	1.8613
7	3	1	1	18	6.7937
8	3	1	1	18	26.9172
9	3	1	1	18	110.7459

Table 5.13: FAS performance for varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.9$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	3	1	1	33	1.5486
6	3	1	1	35	3.6341
7	3	1	1	33	12.4522
8	3	1	1	36	53.3293
9	3	1	1	34	218.8187

Table 5.14: FAS performance for varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 1$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	3	1	1	50	2.1967
6	3	1	1	50	5.4753
7	3	1	1	50	20.1664
8	3	1	1	50	80.1653
9	3	1	1	50	323.8340

FAS with Jacobi smoother and varying *Coarse grid level*

The next set of results in Tables 5.15 to 5.17 show the effect of varying the coarsest grid level when $\omega = 0.8$, Tol $= 1e-8$ and 1 pre- and 1 post-smooth are used. Each of these results is compared to those in Tables 5.12, suggesting that level = 3 is the best choice for the coarsest grid; therefore, we will use this grid level when comparing the number of pre- and post-smooth V-cycles.

Table 5.15: FAS performance for varying grid size for the thin film flow in the steady-state case with nonlinear Jacobi smoother and $\omega = 0.8$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	4	1	1	17	1.0801
6	4	1	1	17	2.2852
7	4	1	1	18	7.3415
8	4	1	1	18	27.6254
9	4	1	1	18	114.5418

Table 5.16: FAS performance for varying grid size for the thin film flow in the steady-state case with nonlinear Jacobi smoother and $\omega = 0.8$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
6	5	1	1	17	4.6934
7	5	1	1	18	9.6021
8	5	1	1	18	29.8901
9	5	1	1	18	113.8811

Table 5.17: FAS performance for varying grid size for the thin film flow in the steady-state case with nonlinear Jacobi smoother and $\omega = 0.8$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
7	6	1	1	17	21.0544
8	6	1	1	17	39.8623
9	6	1	1	18	127.0208

FAS with Jacobi smoother and varying $(pre, post)_{smooth}$

We now investigate the impact of increasing the number of pre- and post- smooths. In Tables 5.18 to 5.22 we use FAS with $Coarse\ Grid = 3$ and $\omega = 0.8$ for the nonlinear Jacobi parameter. The data obtained in these tables show how the results of the implementation of FAS with values for the $(pre, post)_{smooth}$ V-cycle varied as $(2,1), (2,2), (3,1), (3,2)$, and $(3,3)$. We can observe that for a higher pre- and post-smooth value the number of V-cycles goes down. However, despite this, the computation timing increases relative to Table 5.12 due to the extra work of pre and post smoothing. Consequently, we conclude that the best approach to use the pre- and post-smooth $(1,1)$ in this case.

Table 5.18: FAS performance for varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	3	2	1	13	0.9794
6	3	2	1	14	2.0484
7	3	2	1	14	7.5969
8	3	2	1	14	30.3407
9	3	2	1	14	124.6346

Table 5.19: FAS performance for varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	3	2	2	11	0.9805
6	3	2	2	12	2.2287
7	3	2	2	12	8.4265
8	3	2	2	12	33.7618
9	3	2	2	11	127.5553

Table 5.20: FAS performance for varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	3	3	1	11	1.0454
6	3	3	1	12	2.2434
7	3	3	1	12	8.4433
8	3	3	1	12	33.8368
9	3	3	1	12	139.1426

Table 5.21: FAS performance for varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	3	3	2	10	1.0412
6	3	3	2	11	2.4819
7	3	3	2	11	9.5792
8	3	3	2	11	38.0472
9	3	3	2	10	142.6714

Table 5.22: FAS performance for varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No.V-Cycle	-
5	3	3	3	9.0	1.0629
6	3	3	3	10	2.6604
7	3	3	3	10	10.2694
8	3	3	3	10	41.2773
9	3	3	3	10	169.5089

FAS with Red Black Gauss-Seidel with varying damping factor ω

We now turn to FAS with the Red-Black Gauss-Seidel smoother; we again begin by varying the values ω with $(pre, post)_{smooth}$ set to $(1, 1)$ and relative tolerance Tol $= 1e-8$. In Tables 5.23 to 5.28 we display the results of the numerical solution of the thin film flow using the FAS algorithm with $\omega = 0.8$, 0.9, 1, 1.1, 1.2, 1.5, $(pre, post)_{smooth} = (1, 1)$ and the $Coarse\ Grid = 3$. We say from Tables 5.26 and 5.27 the best choice is in the range, $\omega = 1.1$ and $\omega = 1.2$. The results presented in Table 5.26 and Table 5.27 are almost the same, hence we will continue with $\omega = 1.2$. Note that those iterations counts and solution times are better than those obtained using Jacobi smoothing.

Table 5.23: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 0.8$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	3	1	1	13	0.9020
6	3	1	1	14	1.7741
7	3	1	1	14	6.4454
8	3	1	1	14	25.5605
9	3	1	1	14	104.8452

Table 5.24: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 0.9$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	3	1	1	11	0.8180
6	3	1	1	11	1.4028
7	3	1	1	12	5.5443
8	3	1	1	12	22.0452
9	3	1	1	11	83.5470

Table 5.25: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	3	1	1	10	0.9964
6	3	1	1	10	1.3196
7	3	1	1	10	4.7933
8	3	1	1	10	19.1804
9	3	1	1	10	78.5549

Table 5.26: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.1$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	3	1	1	9	0.7306
6	3	1	1	9	1.1573
7	3	1	1	9	4.1897
8	3	1	1	9	16.6104
9	3	1	1	9	68.7326

Table 5.27: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	3	1	1	9	0.7431
6	3	1	1	9	1.1468
7	3	1	1	9	4.1946
8	3	1	1	9	16.6510
9	3	1	1	9	68.9505

Table 5.28: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.5$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	3	1	1	15	0.9863
6	3	1	1	16	2.0189
7	3	1	1	16	7.3688
8	3	1	1	16	29.1187
9	3	1	1	16	119.2528

FAS with Red Black Gauss-Seidel smoother and varying *Coarse grid level*

Tables 5.29 to 5.31 show results obtained with Red Black Gauss-Seidel varying coarse grid level G, whilst fixing $\omega = 1.2$, 1 pre- and 1 post- smooth and Tolerance $= 1e-8$. In this case, a coarsest level of *Coarse Grid* $= 5$ (i.e. 33×65) appears to be the optimal choice.

Table 5.29: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	4	1	1	7	0.8659
6	4	1	1	8	1.2688
7	4	1	1	8	4.0395
8	4	1	1	8	15.5780
9	4	1	1	8	64.1942

Table 5.30: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
6	5	1	1	7	2.2061
7	5	1	1	7	4.3632
8	5	1	1	7	14.3745
9	5	1	1	7	56.7912

Table 5.31: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
7	6	1	1	7	8.8157
8	6	1	1	7	18.9920
9	6	1	1	7	62.389

FAS with Red Black Gauss-Seidel smoother and varying $(pre, post)_{smooth}$

In Tables 5.32 to 5.36 we vary the number of pre- and post- smooth using Red-Black Gauss-Seidel with $\omega = 1.2$, a coarsest level of grid $Coarse\ Grid = 5$ and Tol=$1e-8$. As with Jacobi smoothing, we see that more applications of the smoother always increases the execution time (compared to Table 5.30), despite reducing the number of cycles.

Table 5.32: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
6	5	2	1	6	2.4477
7	5	2	1	6	4.8636
8	5	2	1	6	17.0141
9	5	2	1	6	67.7573

Table 5.33: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
6	5	2	2	5	2.3402
7	5	2	2	6	6.0235
8	5	2	2	6	21.8361
9	5	2	2	6	87.5491

Table 5.34: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
6	5	3	1	6	2.6502
7	5	3	1	6	5.9880
8	5	3	1	6	21.6831
9	5	3	1	6	86.9037

Table 5.35: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
6	5	3	2	5	2.4898
7	5	3	2	6	7.1425
8	5	3	2	6	26.5750
9	5	3	2	6	107.0374

Table 5.36: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$ and Tol $= 1e-8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
6	5	3	3	5	2.6804
7	5	3	3	5	6.8983
8	5	3	3	5	26.1816
9	5	3	3	5	105.7665

FAS with Red Black Gauss-Seidel smoother and varying damping factor ω and Tol=$1e-4$

Tables 5.37, 5.38, 5.39, 5.40 and Table 5.41 detail a series of experiments where we used FAS with the Red-Black Gauss-Seidel smoother, $(pre, post)_{smooth} = (1, 1)$ and relative tolerance Tol $= 1e-4$ with varying values for the parameter $\omega = 0.8$, 0.9, 1, 1.1, 1.5 and the $Coarse\ Grid = 3$. It is apparent from these tables that if we double the exponential of the relative tolerance for FAS from Tol $= 1e-4$ to Tol $= 1e-8$, as a result the number of V-cycles approximately doubles (compared with Table 5.23). Furthermore, we see that the choice of Tol has no impact on the optimal parameter choices for FAS.

Table 5.37: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 0.8$ and Tol $= 1e-4$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	3	1	1	7	1.7562
6	3	1	1	7	1.3118
7	3	1	1	7	4.9286
8	3	1	1	7	19.7533
9	3	1	1	7	81.5252

Table 5.38: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 0.9$ and Tol $= 1e-4$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	3	1	1	5	0.3091
6	3	1	1	6	1.1042
7	3	1	1	6	4.2145
8	3	1	1	6	16.9673
9	3	1	1	6	70.1030

Table 5.39: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1$ and Tol $= 1e-4$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	3	1	1	5	0.3251
6	3	1	1	5	0.9213
7	3	1	1	5	3.5355
8	3	1	1	5	14.2875
9	3	1	1	5	59.3023

Table 5.40: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.1$ and Tol $= 1e-4$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	3	1	1	4	0.2518
6	3	1	1	4	0.7476
7	3	1	1	4	2.8526
8	3	1	1	4	11.5240
9	3	1	1	4	48.1700

Table 5.41: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.5$ and Tol $= 1e-4$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	3	1	1	8	0.5135
6	3	1	1	8	1.4734
7	3	1	1	8	5.6196
8	3	1	1	8	22.4259
9	3	1	1	8	91.4122

FAS with Red Black Gauss-Seidel smoother, *Coarse grid level*$= 4$ and Tol=$1e-4$

We undertake experiment for the Red-Black Gauss-Seidel smoother with the $Coarse\ Grid = 4$ and the tolerance is $Tol = 1e-4$. Table 5.42 shows that we again take approximately half the number of V-cycles compared to $Tol = 1e-8$ (see Table 5.29).

Table 5.42: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$ and Tol $= 1e-4$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
5	4	1	1	4	0.6404
6	4	1	1	4	0.6662
7	4	1	1	4	2.0394
8	4	1	1	4	7.9207
9	4	1	1	4	33.5427

Summary

We have now investigated how a choice between the Jacobi and the Red-Black Gauss-Seidel smoother with varying values of free parameters affects the performance of the FAS algorithm applied to the thin film flow in the steady-state problem. The best results in terms of the overall execution time that we have obtained are presented in Table 5.30. The experiment in Table 5.30 demonstrates application of FAS with the nonlinear Red-Black Gauss-Seidel smoother with parameter $\omega = 1.2$, $(pre, post)_{smooth} = (1, 1)$, $Coarse\ Grid = 5$ and the relative tolerance $\text{Tol} = 1e-8$. All of these empirical results strongly suggest that our implementation is optimal for the thin film flow in the steady-state problem. To sum up, all numerical results of our FAS algorithm that are given here have shown results we require a bounded number of iterations, independent of grid size.

In the following subsection, we will present the numerical results obtained with the Newton-MG algorithm for the thin film flow system in the steady-state problem.

5.7.2 Numerical Results using the Newton-MG

We have performed extensive numerical experiments in order to optimize the parameter selection for the Newton-MG algorithm for the thin film flow system in the steady-state case. We present these experiments in this section.

Newton-MG with Jacobi smoother and varying damping factor ω

In Tables 5.43 to 5.47 ω is varied in the range of $\omega = 1/3$, $1/2$, $2/3$, 0.8 and $\omega = 0.9$, with the goal of determining the optimal value for the linear Jacobi smoother with a $(1, 1)$ pre- and post-smoother and the coarsest grid level $= 3$. In each test, we are using a fixed number of V-cycles (3) to undertake each approximate linear slove. We found that the best choice for the free parameter ω for the Jacobi smoother is $\omega = 0.8$ since for ω less or greater than 0.8 the overall performance of Newton-MG is slower. Hence, we will use $\omega = 0.8$ in the Newton-MG algorithm with the Jacobi smoother in order to determine the best values for other free parameters.

Table 5.43: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi with $\omega = 1/3$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	3	1	1	3	21	1.7336
6	3	1	1	3	22	5.7000
7	3	1	1	3	23	24.0488
8	3	1	1	3	24	103.1880
9	3	1	1	3	25	433.3207

Table 5.44: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi with $\omega = 1/2$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	3	1	1	3	13	1.2531
6	3	1	1	3	14	3.6475
7	3	1	1	3	15	15.6734
8	3	1	1	3	16	68.1310
9	3	1	1	3	16	277.6096

Table 5.45: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi with $\omega = 2/3$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	3	1	1	3	10	1.0320
6	3	1	1	3	11	2.8734
7	3	1	1	3	11	11.4921
8	3	1	1	3	12	51.1680
9	3	1	1	3	12	209.9194

Table 5.46: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi with $\omega = 0.8$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol = $1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	3	1	1	3	8	0.8860
6	3	1	1	3	9	2.3694
7	3	1	1	3	10	10.4693
8	3	1	1	3	10	42.6646
9	3	1	1	3	10	174.9716

Table 5.47: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi with $\omega = 0.9$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol = $1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	3	1	1	3	15	1.3442
6	3	1	1	3	17	4.4525
7	3	1	1	3	17	17.7399
8	3	1	1	3	18	76.6504
9	3	1	1	3	18	311.5757

Newton-MG with Jacobi smoother and varying *Coarse grid level*

In Tables 5.48 and 5.49 we contrast the effect of varying the coarsest grid, relative to Table 5.46 ($Coarse\ Grid = 3$). We see that both results are superior to those in Table 5.46. Suggestion that $Coarse\ Grid = 4$ or $Coarse\ Grid = 5$ are better choices.

Table 5.48: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	4	1	1	3	8	2.2833
6	4	1	1	3	9	2.2841
7	4	1	1	3	9	9.0643
8	4	1	1	3	10	41.6204
9	4	1	1	3	10	168.3731

Table 5.49: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
6	5	1	1	3	9	2.6596
7	5	1	1	3	9	9.0757
8	5	1	1	3	10	41.1382
9	5	1	1	3	10	168.0172

Newton-MG with Jacobi smoother and varying $(pre, post)_{smooth}$

In tables 5.50 to 5.54 we use Newton-MG with *Coarse Grid* = 5 and $\omega = 0.8$ for the linear Jacobi smoother. The data obtained in these tables present the results of the implementation of Newton-MG with values for the $(pre, post)_{smooth}$ V-cycle varying between $(2,1), (2,2), (3,1), (3,2)$, and $(3,3)$, again based upon a fixed number of V-cycles per linear solve. We can see that even if we have a fewer number of Newton iterations, the total timing becomes worse. Therefore, the pre- and post-smooth $(1,1)$ is the best choice in this problem, with fixed V-cycles per inner solve, as shown in Table 5.48.

Table 5.50: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
6	3	2	1	3	7	2.6626
7	3	2	1	3	8	11.1954
8	3	2	1	3	8	46.7468
9	3	2	1	3	8	189.9418

Table 5.51: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
6	3	2	2	3	6	2.5352
7	3	2	2	3	6	12.3482
8	3	2	2	3	7	51.6027
9	3	2	2	3	7	211.5170

Table 5.52: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
6	3	3	1	3	6	2.8363
7	3	3	1	3	7	12.7122
8	3	3	1	3	7	53.0417
9	3	3	1	3	8	247.6334

Table 5.53: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
6	3	3	2	3	6	2.9703
7	3	3	2	3	7	13.8123
8	3	3	2	3	7	60.3435
9	3	3	2	3	7	258.5162

Table 5.54: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Jacobi $\omega = 0.8$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
6	3	3	3	3	6	3.0349
7	3	3	3	3	6	13.6574
8	3	3	3	3	7	67.5521
9	3	3	3	3	7	277.4261

Newton-MG with Red Black Gauss-Seidel smoother and varying damping factor ω

In Tables 5.55 to 5.60, we present the results of the numerical solution of the thin film flow using the Newton-MG algorithm with the Red Black Gauss-Seidel smoother and damping factors $\omega = 0.8, 0.9, 1, 1.1, 1.2, 1.5$, $(pre, post)_{smooth} = (1, 1)$, three V-cycles per linear solve and the $Coarse\ Grid = 3$. Tables 5.57 to 5.59 show the best choice of ω to be between $\omega = 1.0$ and $\omega = 1.2$. Hence subsequent tests will use values in this range.

Table 5.55: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 0.8$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	3	1	1	3	7	0.8453
6	3	1	1	3	8	2.0750
7	3	1	1	3	8	8.2375
8	3	1	1	3	8	33.7755
9	3	1	1	3	9	155.0077

Table 5.56: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 0.9$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	3	1	1	3	6	0.7643
6	3	1	1	3	7	1.8013
7	3	1	1	3	8	7.2468
8	3	1	1	3	8	33.8062
9	3	1	1	3	8	138.9229

Table 5.57: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	3	1	1	3	6	0.7504
6	3	1	1	3	6	1.4803
7	3	1	1	3	7	6.9759
8	3	1	1	3	7	28.5065
9	3	1	1	3	7	117.4988

Table 5.58: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.1$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	3	1	1	3	6	0.7642
6	3	1	1	3	6	1.4771
7	3	1	1	3	6	5.9768
8	3	1	1	3	6	24.5663
9	3	1	1	3	7	117.8308

Table 5.59: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	3	1	1	3	6	0.7516
6	3	1	1	3	6	1.4798
7	3	1	1	3	6	5.9709
8	3	1	1	3	6	24.3250
9	3	1	1	3	7	117.3044

Table 5.60: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.5$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	3	1	1	3	8	0.8896
6	3	1	1	3	9	2.3411
7	3	1	1	3	10	10.3174
8	3	1	1	3	10	42.2273
9	3	1	1	3	10	173.9966

Newton-MG with Red Black Gauss-Seidel smoother and varying V-cycle Tol

Tables 5.61 to 5.63 present the numerical results for Newton-MG performance for the thin film flow in the steady-state case with Red Black Gauss-Seidel smoother with $\omega = 1.1$, $(pre, post)_{smooth} = (1, 1)$, $Coarse\ Grid = 4$. We have used a varying number of V-cycles (with a maximum number of 20), based upon different values for the relative stepping tolerance based on the linear residual. The values considered are Tol $= 1e-2$ Tol $= 1e-4$ and Tol $= 1e-6$, respectively: with the former always being superior.

Similarly, Tables 5.64, 5.65 and 5.66, present results computed with different fixed choices of V-cycle number for each Newton step. We have again used the Red Black G-S smoother with $\omega = 1.1$, $(pre, post)_{smooth} = (1, 1)$, $Coarse\ Grid = 4$. Tables 5.61 to 5.63 show that when we vary the tolerance the number of V-cycles is almost always fixed. Therefore, considering tables 5.61 to 5.66 together, from now on we are going to fix the number of V-cycles (at 3) rather than fix the tolerance.

Table 5.61: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.1$, maximum number of V-cycles is 20 and Tol $= 1e-2$.

Input				Output		
Grid Size		Smoother		Newton Solver	MG Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. Newton Iteration	No. MG V-Cycle	-
5	4	1	1	6	2	0.6818
6	4	1	1	6	3	1.5447
7	4	1	1	6	3	6.2639
8	4	1	1	6	3	25.6072
9	4	1	1	7	3	123.2017

Table 5.62: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.1$, maximum number of V-cycles is 20 and Tol $= 1e-4$.

Input				Output		
Grid Size		Smoother		Newton Solver	MG Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. Newton Iteration	No. MG V-Cycle	-
5	4	1	1	5	4	2.1790
6	4	1	1	5	5	1.8114
7	4	1	1	5	5	7.7283
8	4	1	1	6	5	37.8208
9	4	1	1	6	5	154.9409

Table 5.63: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.1$, maximum number of V-cycles is 20 and Tol $= 1e-6$.

Input				Output		
Grid Size		Smoother		Newton Solver	MG Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. Newton Iteration	No. MG V-Cycle	-
5	4	1	1	5	6	0.9217
6	4	1	1	5	7	2.6403
7	4	1	1	5	7	11.1125
8	4	1	1	5	7	46.8816
9	4	1	1	5	7	195.2469

Table 5.64: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.1$, fixed number of V-cycles = 3.

Input				Output		
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	4	1	1	3	6	0.4078
6	4	1	1	3	6	1.5277
7	4	1	1	3	6	6.2500
8	4	1	1	3	6	25.5887
9	4	1	1	3	7	123.2528

Table 5.65: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.1$, fixed V-cycle, the number of V-cycles = 5 and the relative tolerance for Newton is Tol $= 1e-8$.

Input				Output		
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	4	1	1	5	5	2.3588
6	4	1	1	5	5	1.8444
7	4	1	1	5	5	7.7822
8	4	1	1	5	6	38.4403
9	4	1	1	5	6	156.5030

Table 5.66: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.1$, fixed V-cycle, the number of V-cycles = 7 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	4	1	1	7	5	0.6540
6	4	1	1	7	5	2.6332
7	4	1	1	7	5	11.5455
8	4	1	1	7	5	47.2581
9	4	1	1	7	5	193.0136

Newton-MG with Red Black Gauss-Seidel smoother and varying *Coarse grid level*

We now experiment the coarse grid level to find the optimal choice for the Newton-MG algorithm. Comparing Table 5.59, Table 5.67 and Table 5.68 we observed that the best coarse grid level for this problem is $Coarse\ Grid = 4$ as we can see in Table 5.67.

Table 5.67: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	4	1	1	3	6	0.7679
6	4	1	1	3	6	1.3197
7	4	1	1	3	6	5.6554
8	4	1	1	3	7	27.3475
9	4	1	1	3	6	100.5532

Table 5.68: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
6	5	1	1	3	6	1.2389
7	5	1	1	3	6	5.7582
8	5	1	1	3	6	23.9345
9	5	1	1	3	7	111.8401

Newton-MG with Red Black Gauss-Seidel smoother and varying $(pre, post)_{smooth}$

In Tables 5.69 to 5.73 we present results of experiments with the following $(pre, post)_{smooth}$ values: $(2,1)$, $(2,2)$, $(3,1)$, $(3,2)$, and $(3,3)$, in order to determine the best value of pre- and post- smooth for this algorithm. We can compare to Table 5.67 to see that the choice $(pre, post)_{smooth} = (1,1)$ is most efficient here.

Table 5.69: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	4	2	1	3	6	0.7497
6	4	2	1	3	6	1.8728
7	4	2	1	3	6	7.7227
8	4	2	1	3	7	35.9747
9	4	2	1	3	6	137.7003

Table 5.70: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	4	2	2	3	6	0.8589
6	4	2	2	3	6	1.9488
7	4	2	2	3	6	9.8611
8	4	2	2	3	7	46.1188
9	4	2	2	3	7	188.2890

Table 5.71: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	4	3	1	3	6	0.8109
6	4	3	1	3	6	1.9556
7	4	3	1	3	6	9.4684
8	4	3	1	3	7	38.9025
9	4	3	1	3	6	159.4066

Table 5.72: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	4	3	2	3	6	0.9077
6	4	3	2	3	6	2.3472
7	4	3	2	3	6	9.8690
8	4	3	2	3	7	47.2647
9	4	3	2	3	7	193.1786

Table 5.73: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	No. Newton Iteration	-
5	4	3	3	3	6	0.9702
6	4	3	3	3	6	2.5451
7	4	3	3	3	6	10.7489
8	4	3	3	3	6	44.1336
9	4	3	3	3	7	212.4052

Summary

When we compare the best results of the Newton-MG algorithm with Jacobi smoother with parameter $\omega = 0.8$, $(pre, post)_{smooth} = (1,1)$, $Coarse\ Grid = 5$ and the relative tolerance Tol $= 1e-8$ in Table 5.49 with the best results of the Newton-MG algorithm with Red-Black G-S smoother with $\omega = 1.2$, $(pre, post)_{smooth} = (1,1)$, $Coarse\ Grid = 4$ and the relative tolerance Tol $= 1e-8$ in Table 5.67, we conclude that the Newton-MG algorithm with Red-Black G-S smoother is most efficient in this case. Furthermore, only a very approximate linear multigrid solve is required: based upon 3 V-cycles per Newton iteration. Moreover, in almost all of these examples, when we double the number of grid points we find that the number of V-cycles stays constant. In addition, when the problem size is increased by a factor of 4 the total time is also increased by a factor of approximately 4 which implies a linear complexity of $O(N)$ of the overall algorithm for each Newton iteration.

In the following subsection, we illustrated the numerical results of the thin film flow using the Newton-Krylov-AMG with our proposed new multilevel preconditioners for the steady-state problem.

5.7.3 Numerical Results using Newton-Krylov-AMG

In this section, we perform a wide set of numerical experiments to optimize the parameter selection for the Newton-Krylov-AMG. In order to achieve a suitable preconditioner with robust convergence rates, we will solve the discrete nonsymmetric Jacobian system with the GMRES method and various preconditioners. The results that follow involve various choices for the values of the parameters for the different preconditioners. Within our preconditioners, when we apply $AMG(K)$ and $AMG(K_\alpha)$, we can choose what the $(pre, post)_{smooth}$ values are, compared to the default of $(pre, post)_{smooth} = (2,2)$ in the Harwell Subroutine Library (HSL) [70].

Newton-Krylov-AMG with the Preconditioner P_1

Table 5.74 demonstrates that, as explained in the previous sections and in [97, 135, 136], P_1 in Equation (5.48) is an optimal preconditioner in terms of iterations taken. It is prohibitively expensive, however, due to the computation of the Schur Complement matrix and direct linear algebra.

Table 5.74: Newton-Krylov-AMG solver performance in steady-state case, using P_1 with GMRES maximum iteration $Maxit = 20$, (the relative tolerance for GMRES is $Tol = 1e-4$) and $Restart = 20$ where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
3	4	1	2	1.20	0.2686
4	5	1	2	1.20	0.2060
5	5	1	2	1.20	3.1470
6	5	1	2	1.20	119.2166

Newton-Krylov-AMG with the Preconditioner P_{1a} and varying GMRES Tol

In order to improve the efficiency of the preconditioner, we took the upper triangular blocks of the matrix J and replaced the exact Schur complement with an approximation, i.e. the block matrix $S1 \simeq K_\alpha$ in the preconditioner P_{1a} in Equation (5.49). The numerical results of Tables 5.75, 5.76 and 5.77 show the performance of Newton-Krylov with P_{1a} preconditioner with different tolerances for the GMRES iterations. The advantage of this preconditioner is that it has a constant number of iterations as reported in these tables. It is still relatively slow, however, as we solve the diagonal blocks with direct linear algebra.

Table 5.75: Newton-Krylov-AMG solver performance in steady-state case, using P_{1a} with GMRES maximum iteration $Maxit = 20$, (the relative tolerance for GMRES is $Tol = 1e-2$), $Restart = 20$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	6	8	11	9.16	0.4606
5	6	8	11	9.16	0.2976
6	6	8	11	9.16	1.5546
7	6	8	11	9.16	6.2882
8	7	8	11	9.14	34.1152
9	7	8	11	9.14	160.6913

Table 5.76: Newton-Krylov-AMG solver performance in steady-state case, using P_{1a} with GMRES maximum iteration $Maxit = 20$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	10	12	10.60	0.3643
5	5	10	12	10.60	0.2790
6	6	10	12	10.50	1.8098
7	6	10	12	10.50	6.9315
8	6	10	12	10.50	33.0380
9	6	10	12	10.50	156.0248

Table 5.77: Newton-Krylov-AMG solver performance in steady-state case, using P_{1a} with GMRES maximum iteration $Maxit = 20$, (the relative tolerance for GMRES is $Tol = 1e-4$), $Restart = 20$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	11	14	12.20	0.3671
5	5	11	14	12.20	0.3107
6	5	11	14	12.20	1.6498
7	5	11	14	12.20	6.5998
8	6	11	14	12.16	36.7750
9	6	11	14	12.16	174.6869

Newton-Krylov-AMG with the Preconditioner P_{1b}, $(pre, post)_{smooth} = (1, 1)$ and varying GMRES Tol

As we can see from Tables 5.75, 5.76, 5.77, the results presented demonstrate a good performance of the P_{1a} preconditioner with respect to the GMRES iteration count, which is constant for all cases. On the other hand, the P_{1a} preconditioner is not optimal with respect to running time. Therefore, we need to develop this preconditioner to reduce the time required to solve this system. In order to improve the efficiency of the algorithm, we performed our new preconditioner by replacing the exact Schur Complement with an approximation that uses one AMG V-cycle with Gauss-Seidel smoothing and varying pre- and post-smoothing iterations to solve for the diagonal blocks in the new preconditioner P_{1b} in Equation (5.50). Tables 5.78 to 5.80 present a series of experiments using P_{1b} where we varied tolerance for GMRES between Tol $= 1e-02$, Tol $= 1e-03$ and Tol $= 1e-04$ with Tol $= 1e-08$ as the tolerance for Newton iterations with $(pre, post)_{smooth} = (1, 1)$. We see that all three choices yield close to optimal,

$O(N)$ algorithms, with Tol $= 1e-03$ being marginally best. Moreover, the overall solution times are extremely competitive when compared with the other methods we have considered so far.

Table 5.78: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-2$), $Restart = 20$, $(pre, post)_{smooth} = (1,1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	6	8	11	9.50	1.5636
5	6	8	11	9.50	0.1894
6	7	8	11	9.42	0.8940
7	7	8	11	9.57	2.9539
8	7	8	11	9.57	11.7392
9	7	8	11	9.57	49.7900

Table 5.79: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$, $(pre, post)_{smooth} = (1,1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	11	14	11.80	0.0762
5	6	10	13	11.50	0.1962
6	6	10	14	11.66	0.7764
7	6	10	14	11.83	2.7540
8	6	10	14	11.66	11.1152
9	6	10	14	11.83	47.9053

Table 5.80: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is Tol $= 1e-4$), $Restart = 20$, $(pre, post)_{smooth} = (1,1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	12	16	13.80	0.1892
5	5	12	17	13.80	0.1846
6	5	13	16	14.00	0.7169
7	6	13	16	14.16	3.0731
8	6	13	16	14.00	12.6077
9	6	13	16	14.16	54.1662

Newton-Krylov-AMG with the Preconditioner P_{1b}, $(pre, post)_{smooth} = (2,1)$ and varying GMRES Tol

Tables 5.81, 5.82 and 5.83 show the performance of the Newton-Krylov-AMG with preconditioner P_{1b} with $(pre, post)_{smooth} = (2,1)$ for the AMG parts of the P_{1b} preconditioner. Whilst this is still optimal it is slower than the $(pre, post)_{smooth} = (1,1)$ case.

Table 5.81: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-2$), $Restart = 20$, $(pre, post)_{smooth} = (2,1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	6	8	11	9.50	0.1890
5	6	8	11	9.33	0.1700
6	7	8	11	9.42	0.8102
7	7	8	11	9.57	2.9748
8	7	8	11	9.57	12.1842
9	7	8	11	9.57	51.4669

Table 5.82: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$, $(pre, post)_{smooth} = (2,1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
$Fine$	NNI	Min	Max	$Average$	-
4	5	10	14	11.60	0.1329
5	6	10	13	11.33	0.1956
6	6	10	13	11.33	0.7997
7	6	10	14	11.50	2.8242
8	6	10	14	11.50	11.5426
9	6	10	14	11.66	49.4254

Table 5.83: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is Tol $= 1e-4$), $Restart = 20$, $(pre, post)_{smooth} = (2,1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
$Fine$	NNI	Min	Max	$Average$	-
4	5	12	16	13.80	0.1226
5	5	12	16	13.40	0.1761
6	5	12	16	13.60	0.7294
7	6	13	16	14.00	3.1993
8	6	13	16	14.00	13.0380
9	6	13	16	14.00	56.0957

Newton-Krylov-AMG with the Preconditioner P_{1b}, $(pre, post)_{smooth} = (2,2)$ and varying GMRES Tol

Tables 5.84 to 5.86 present a series of experiments with $(pre, post)_{smooth} = (2,2)$ for the P_{1b} preconditioner. Again the results are close to optimal but slower than the $(pre, post)_{smooth} = (1,1)$ case.

Table 5.84: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-2$), $Restart = 20$, $(pre, post)_{smooth} = (2, 2)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	6	8	11	9.33	0.1453
5	6	8	11	9.33	0.1735
6	6	8	11	9.33	0.7088
7	7	8	11	9.42	2.9970
8	7	8	11	9.42	12.3526
9	7	8	11	9.57	53.4829

Table 5.85: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$, $(pre, post)_{smooth} = (2, 2)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	10	13	11.20	0.1422
5	6	10	13	11.00	0.1651
6	6	10	13	11.16	0.8109
7	6	10	13	11.16	2.8836
8	6	10	12	11.00	11.5203
9	6	10	14	11.66	50.5599

Table 5.86: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is Tol $= 1e-4$), $Restart = 20$, $(pre, post)_{smooth} = (2,2)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	11	15	13.00	0.1544
5	5	12	16	13.20	0.1808
6	5	12	15	13.40	0.8016
7	5	12	15	13.40	2.6800
8	6	12	15	13.33	12.8699
9	6	12	16	13.66	57.2634

Newton-Krylov-AMG with the Preconditioner P_{1b}, $(pre, post)_{smooth} = (3,3)$ and varying GMRES Tol

Tables 5.87, 5.88 and 5.89 show the performance of the Newton-Krylov-AMG with $(pre, post)_{smooth} = (3,3)$ for the P_{1b} preconditioner. Once more these results are inefficient compared to the $(pre, post)_{smooth} = (1,1)$ case.

Table 5.87: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-2$), $Restart = 20$, $(pre, post)_{smooth} = (3,3)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	6	8	11	9.16	1.6449
5	6	8	11	9.33	0.2409
6	6	8	11	9.16	0.8569
7	6	8	11	9.33	2.8122
8	7	8	11	9.42	13.1644
9	7	8	11	9.42	54.8534

Table 5.88: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$, $(pre, post)_{smooth} = (3,3)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	10	12	11.00	0.1456
5	5	10	13	11.00	0.1736
6	6	10	12	11.00	0.8373
7	6	10	12	11.00	3.0425
8	6	10	13	11.16	12.4137
9	6	10	13	11.16	52.5767

Table 5.89: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is Tol $= 1e-4$), $Restart = 20$, $(pre, post)_{smooth} = (3,3)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	11	15	13.00	0.3912
5	5	11	15	12.80	0.1936
6	5	11	15	12.80	0.7843
7	5	12	15	13.00	2.7414
8	6	12	15	13.33	13.8272
9	6	12	15	13.33	58.3836

Newton-Krylov-AMG with the Preconditioner P_2 and varying GMRES Tol

Now we repeat all experiments for the preconditioner P_1 in Equation (5.48) with the preconditioner P_2 in Equation (5.51). In Tables 5.90 to 5.92 we show that P_2 with different GMRES tolerances is an optimal preconditioner in terms of bounding Newton and GMRES iterations. However, it is prohibitively expensive, since we are required to solve the diagonal blocks with direct linear algebra.

Table 5.90: Newton-Krylov-AMG solver performance in steady-state case, using P_2 with the exact Schur Complement $S_2 = K - IK_\alpha^{-1}B_\alpha$, with GMRES maximum iteration $Maxit = 20$, (the relative tolerance for GMRES is $Tol = 1e-2$), $Restart = 20$, and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
3	4	1	2	1.75	0.2768
4	5	1	2	1.80	0.2171
5	5	1	2	1.80	3.4141
6	5	1	2	1.80	143.9946

Table 5.91: Newton-Krylov-AMG solver performance in steady-state case, using P_2 with the exact Schur Complement $S_2 = K - IK_\alpha^{-1}B_\alpha$, with GMRES maximum iteration $Maxit = 20$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$, and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
3	4	1	2	1.75	0.2718
4	5	1	2	1.80	0.2527
5	5	1	2	1.80	3.4280
6	5	1	2	1.80	140.4432

Table 5.92: Newton-Krylov-AMG solver performance in steady-state case, using P_2 with the exact Schur Complement $S_2 = K - IK_\alpha^{-1}B_\alpha$, with GMRES maximum iteration $Maxit = 20$, (the relative tolerance for GMRES is $Tol = 1e-4$), $Restart = 20$, and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
3	4	1	2	1.75	0.2589
4	5	1	2	1.80	0.2519
5	5	1	2	1.80	3.5432
6	5	1	2	1.80	141.8519

Newton-Krylov-AMG with the Preconditioner P_{2a} and varying GMRES Tol

In order to develop the efficiency of the preconditioner, we took the lower triangular blocks of the matrix J and replaced the exact Schur complement with an approximation, i.e. the block matrix $S2 \simeq K$ in the preconditioner P_{2a} in Equation (5.52). The numerical results of Tables 5.93, 5.94 and 5.95 show the performance of Newton-Krylov with P_{2a} preconditioner with different tolerances for the GMRES iterations. It is still far too slow due to the direct solution of the blocks. However, the benefit of this preconditioner is that it has a bounded number of iterations despite the approximation of the Schur complement as recorded in the previous tables.

Table 5.93: Newton-Krylov-AMG solver performance in steady-state case, using P_{2a} with GMRES maximum iteration $Maxit = 20$, (the relative tolerance for GMRES is $Tol = 1e-2$), $Restart = 20$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	6	8	9	8.66	0.3654
5	6	8	9	8.66	0.2951
6	6	8	9	8.66	1.5369
7	7	8	9	8.57	7.1852
8	7	8	9	8.57	33.3193
9	7	8	9	8.57	157.1997

Table 5.94: Newton-Krylov-AMG solver performance in steady-state case, using P_{2a} with GMRES maximum iteration $Maxit = 20$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	10	11	10.60	0.3840
5	6	10	11	10.33	0.3854
6	6	10	11	10.16	1.8421
7	6	10	11	10.16	7.2628
8	6	10	11	10.16	33.8123
9	6	10	11	10.16	157.4489

Table 5.95: Newton-Krylov-AMG solver performance in steady-state case, using P_{2a} with GMRES maximum iteration $Maxit = 20$, (the relative tolerance for GMRES is $Tol = 1e-4$), $Restart = 20$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	11	12	11.80	0.3914
5	5	11	12	11.80	0.3233
6	5	11	12	11.80	1.6995
7	6	12	12	12.00	8.1155
8	6	12	12	12.00	38.2169
9	6	12	12	12.00	179.7662

Newton-Krylov-AMG with the Preconditioner P_{2b}, $(pre, post)_{smooth} = (1, 1)$ and varying GMRES Tol

As we can see from Tables 5.93, 5.94 and 5.95 the results presented demonstrate a good performance of the P_{2a} preconditioner with respect to the Newton and GMRES iteration count, which is constant for all cases. On the other hand, the P_{2a} preconditioner is not optimal with respect to running time. Therefore, we need to improve this preconditioner to decrease the time needed to solve this system. In order to improve the efficiency of the algorithm, we created our new preconditioner by replacing the exact Schur Complement with the approximation that applies one AMG V-cycle with Gauss-Seidel smoothing and varying pre- and post-smoothing iterations to solve the lower triangular blocks in the new preconditioner P_{2b} in Equation (5.53).

In the same way we analyse the performance of the lower triangular block preconditioner P_{2b}. Tables 5.96 to 5.98 display a series of experiments where we varied the tolerance of GMRES between Tol $= 1e-02$, Tol $= 1e-03$ and Tol $= 1e-04$, where the tolerance of Newton iterations is Tol $= 1e-08$ with $(pre, post)_{smooth} = (1, 1)$ for the P_{2b} preconditioner.

Table 5.97 shows the best performance of the Newton-Krylov-AMG with our new P_{2b} preconditioner with Tol $= 1e-03$ of GMRES iterations with $(pre, post)_{smooth} = (1, 1)$. We see that all three choices yield optimal, $O(N)$ algorithms, however, with Tol $= 1e-03$ being marginally best.

Table 5.96: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-2$), $Restart = 20$, $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	6	8	10	9.00	0.1372
5	7	8	10	9.28	0.2059
6	7	8	10	9.28	0.7881
7	7	8	10	9.28	2.8808
8	7	8	10	9.28	11.6850
9	7	8	10	9.28	49.2499

Table 5.97: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$, $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	10	12	11.20	0.0887
5	6	10	13	11.16	0.1959
6	6	10	13	11.16	0.7592
7	6	10	13	11.33	2.7394
8	6	10	13	11.16	11.0972
9	6	10	13	11.50	47.7340

Table 5.98: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is Tol $= 1e-4$), $Restart = 20$, $(pre, post)_{smooth} = (1,1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	12	14	13.20	0.0625
5	5	13	14	13.20	0.1784
6	6	13	15	13.66	0.8596
7	6	13	15	13.66	3.0550
8	6	13	15	13.66	12.5570
9	6	13	15	14.00	54.3331

Newton-Krylov-AMG with the Preconditioner P_{2b}, $(pre, post)_{smooth} = (2,1)$ and varying GMRES Tol

Tables 5.99, 5.100 and 5.101 show the performance of the Newton-Krylov-AMG with our new preconditioner for the P_{2b} preconditioner with tolerance Tol $= 1e-08$ for Newton iterations and a varying number of GMRES iterations with the AMG solver undertaken using $(pre, post)_{smooth} = (2,1)$. Whilst this is still optimal it is slower than the $(pre, post)_{smooth} = (1,1)$ case.

Table 5.99: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-2$), $Restart = 20$, $(pre, post)_{smooth} = (2,1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	6	8	10	9.00	0.1672
5	6	8	10	9.16	0.1842
6	7	8	10	9.00	0.7947
7	7	8	10	9.14	2.8884
8	7	8	10	9.28	12.0089
9	7	8	10	9.28	50.6377

Table 5.100: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$, $(pre, post)_{smooth} = (2,1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	10	12	11.00	0.1318
5	6	10	13	11.00	0.1891
6	6	10	13	11.00	0.7765
7	6	10	13	11.16	2.8153
8	6	10	13	11.16	11.4794
9	6	10	13	11.33	48.8011

Table 5.101: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is Tol $= 1e-4$), $Restart = 20$, $(pre, post)_{smooth} = (2,1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	13	14	13.40	0.1572
5	5	13	14	13.20	0.1840
6	5	13	14	13.40	0.7453
7	6	13	15	13.66	3.1882
8	6	12	15	13.33	12.8373
9	6	13	15	13.66	56.2028

Newton-Krylov-AMG with the Preconditioner P_{2b}, $(pre, post)_{smooth} = (2,2)$ and varying GMRES Tol

Tables from 5.102 to 5.104 present a series of experiments where we use various tolerances for GMRES which are Tol $= 1e-02$, Tol $= 1e-03$ and Tol $= 1e-04$ with tolerance $1e-08$ for Newton iterations with the AMG solver undertaken using $(pre, post)_{smooth} = (2,2)$ for the P_{2b} preconditioner. Again the results are optimal but slower than the $(pre, post)_{smooth} = (1,1)$ case.

Table 5.102: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-2$), $Restart = 20$, $(pre, post)_{smooth} = (2, 2)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	6	8	10	9.00	0.1433
5	6	8	10	8.83	0.1745
6	7	8	10	9.14	0.8815
7	7	8	10	8.85	3.0020
8	7	8	10	9.14	12.1891
9	7	8	10	9.28	51.5405

Table 5.103: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$, $(pre, post)_{smooth} = (2, 2)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	10	12	10.80	0.2042
5	6	10	12	10.66	0.2211
6	6	10	12	10.66	0.7892
7	6	10	12	10.83	2.8675
8	6	10	12	10.83	11.7889
9	6	10	13	11.33	50.9960

Table 5.104: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is Tol $= 1e-4$), $Restart = 20$, $(pre, post)_{smooth} = (2, 2)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	12	14	13.00	0.1444
5	5	12	14	12.50	0.1825
6	5	12	14	12.80	0.7453
7	6	13	14	13.33	3.2500
8	6	12	15	13.00	13.1054
9	6	13	14	13.33	56.9457

Newton-Krylov-AMG with the Preconditioner P_{2b}, $(pre, post)_{smooth} = (3, 3)$ and varying GMRES Tol

Tables 5.105, 5.106 and 5.107 show that the performance of the Newton-Krylov-AMG with the tolerance Tol $= 1e-08$ for Newton iterations and a varying number of GMRES iterations with the AMG solver undertaken using $(pre, post)_{smooth} = (3, 3)$ for the P_{2b} preconditioner. These numerical results are still optimal, however, it is still slower than the $(pre, post)_{smooth} = (1, 1)$ case.

Table 5.105: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-2$), $Restart = 20$, $(pre, post)_{smooth} = (3, 3)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	6	8	10	9.00	0.1581
5	6	8	10	8.83	0.1825
6	6	8	10	8.83	0.7361
7	7	8	10	9.00	3.1785
8	7	8	10	8.85	13.0604
9	7	8	10	9.00	55.7512

Table 5.106: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$, $(pre, post)_{smooth} = (3,3)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
$Fine$	NNI	Min	Max	$Average$	-
4	5	10	12	10.80	0.0665
5	6	10	12	10.66	0.2100
6	6	10	12	10.66	0.8264
7	6	10	12	10.66	3.0199
8	6	10	13	10.83	12.4680
9	6	10	12	10.83	53.4020

Table 5.107: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is Tol $= 1e-4$), $Restart = 20$, $(pre, post)_{smooth} = (3,3)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
$Fine$	NNI	Min	Max	$Average$	-
4	5	12	14	12.80	0.1579
5	5	12	14	12.60	0.1919
6	5	12	14	12.80	0.7945
7	6	12	15	12.83	3.4287
8	6	12	15	12.83	14.1624
9	6	12	15	13.00	60.5789

Summary

From all the previous numerical results for the P_{1b} and P_{2b} preconditioners we found that the optimal performance is given by Table 5.79 and Table 5.97 where GMRES tolerance is Tol $= 1e-03$ and $(pre, post)_{smooth} = (1,1)$. It is apparent from Tables 5.79 and 5.97 that these preconditioners P_{1b} and P_{2b} each require a small and bounded number of Newton iterations and both of them have short running time which confirms that they are superior to other preconditioners.

To conclude, from all these results, we can remark that the best new preconditioners for solving the thin film flow in the steady-state case system are P_{1b} and P_{2b} which significantly decrease the number of Newton and GMRES iterations as well as the execution time compared to no

preconditioning. Moreover, when we compared the execution time of P_{1b} and P_{2b} we observed that they are much cheaper overall and also, their execution times scale linearly with the size of the problem. In the following subsection, we discuss and compare the results of the numerical solution of the thin film flow system using the three separate nonlinear multilevel algorithms in order to decide on the best numerical solution for the steady-state problem in two dimensions.

5.7.4 Discussion and Comparison

In this subsection, we compare the FAS, Newton-MG and Newton-Krylov algorithms for the numerical solution of the thin film flow in the steady-state problems. The analysis of the time-dependent problems will develop in section 5.8.

We have demonstrated that all three nonlinear multilevel algorithms are successful with regards to the thin film flow system. Optimal FAS, Newton-MG and Newton-Krylov-AMG performance show that all three multilevel algorithms have $O(N)$ cost. We have used the analytic Jacobian for all three nonlinear multilevel algorithms to improve their efficiency. Nevertheless, the best parameters in each case were obtained through extensive numerical tests; we found that these algorithms are sensitive to the choice of parameters and that the best selection of parameters for one algorithm is not necessarily the best choice for the other two algorithms.

For the FAS algorithm we have found that the best value of the parameter ω is $\omega = 1.2$ and the best smoother is Red Black G-S, with the best value of the parameter pre- and post-smooth $(pre, post) = (1, 1)$ and the best coarse grid size is $G = 5$, as shown in Table 5.30. For clarity, this table is presented here again as the following:

Table 5.108: FAS performance for varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$ and Tol $= 1e - 8$.

Input				Output	
Grid Size		Smoother		FAS Solver	Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. V-Cycle	-
6	5	1	1	7	2.2061
7	5	1	1	7	4.3632
8	5	1	1	7	14.3745
9	5	1	1	7	56.7912

In this table, the whole times perform optimally and the number of V-cycle iterations is fixed. The experiments presented in subsection 5.7.1 show that when we increase grid levels, the number of V-cycles remains constant which implies that it is independent of the problem size. Moreover, the execution time of this algorithm increases by approximately a factor of 4 from one run to the next. As the problem size also grows by a factor of 4, this suggests that our FAS

nonlinear multigrid algorithm converges with a linear complexity of $O(N)$.

For the Newton-MG algorithm we have found that the best value of the parameter ω is 1.2, with the best smoother being Red Black G-S smoother, with the best value of the parameter pre- and post-smooth is $(pre, post) = (1, 1)$ and the best coarse grid size is $G = 4$, as shown in Table 5.67. For simplicity, we present this table here again as the following:

Table 5.109: Newton-MG performance for a varying grid size for the thin film flow in the steady-state case with Red Black G-S with $\omega = 1.2$, fixed V-cycle, the number of V-cycles = 3 and the relative tolerance for Newton is Tol $= 1e - 8$.

Input					Output	
Grid Size		Smoother		MG Solver	Newton Solver	Time (sec)
$Fine$	$Coarse$	Pre	$Post$	No. MG V-Cycle	No. Newton Iteration	-
5	4	1	1	3	6	0.7679
6	4	1	1	3	6	1.3197
7	4	1	1	3	6	5.6554
8	4	1	1	3	7	27.3475
9	4	1	1	3	6	100.5532

As we can see from this Table the total times behave optimally and the number of Newton iterations remains almost constant. Furthermore, we have observed that the execution time is increased by approximately a factor of 4 from one run to the next; as the problem size is increased by a factor of 4, this implies that our Newton-MG algorithm converges with a linear complexity of $O(N)$.

For our newly proposed preconditioner applied with Newton-Krylov and with the AMG block solved using $(pre, post) = (1, 1)$ (whereas the default for the AMG preconditioner that is used by applying HSL-MI20 software is $(2, 2)$), we found little to choose between P_{1b} and P_{2b}, where the best results are presented in Table 5.110 and Table 5.111. Here we display these tables again for simplicity as the following:

Table 5.110: Newton-Krylov-AMG solver performance in steady-state case, using P_{1b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$, $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	11	14	11.80	0.0762
5	6	10	13	11.50	0.1962
6	6	10	14	11.66	0.7764
7	6	10	14	11.83	2.7540
8	6	10	14	11.66	11.1152
9	6	10	14	11.83	47.9053

Table 5.111: Newton-Krylov-AMG solver performance in steady-state case, using P_{2b} AMG preconditioned with GMRES maximum iteration $Maxit = 300$, (the relative tolerance for GMRES is $Tol = 1e-3$), $Restart = 20$, $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$, where the number of Newton iterations is NNI.

Input	Output				
Grid size	Newton Solver	GMRES Solver			Time (sec)
Fine	*NNI*	*Min*	*Max*	*Average*	-
4	5	10	12	11.20	0.0887
5	6	10	13	11.16	0.1959
6	6	10	13	11.16	0.7592
7	6	10	13	11.33	2.7394
8	6	10	13	11.16	11.0972
9	6	10	13	11.50	47.7340

As seen in both tables, the computational time is optimal and the needed number of Newton and GMRES iterations is fixed as the grid is refined.

While all three nonlinear multilevel algorithms return an optimal solution for the thin film flow in the steady-state problem, the best performance overall is obtained by the Newton-Krylov-AMG with our new preconditioner.

For the study at hand, it assists to show that the comparison between the very best numerical results of FAS algorithm performance in Table 5.108 with the very best numerical results of Newton-MG algorithm performance in Table 5.109 with the very best numerical results of Newton-Krylov-AMG preconditioner algorithm performance in Table 5.111. From this comparison, this research has shown that the best nonlinear multilevel approach to solve the thin

film flow in the steady-state case is the Newton-Krylov-AMG with our new preconditioner algorithm. The present work has been the first attempt to thoroughly examine the nonlinear system of equations by using these three nonlinear multilevel algorithms. We can highlight that from all these extensive experiments in this section, the Newton-Krylov-AMG with the new preconditioner is our preferred algorithm which gives optimal performance and is superior to both Newton-MG and FAS algorithms in terms of time running.

To conclude, we have obtained an optimal solution through FAS and Newton-MG, however, FAS is superior to Newton-MG for the thin film flow in the steady-state case. Nevertheless, our preconditioned Newton-Krylov-AMG is superior to both. In the following section, we will compare FAS, Newton-MG and Newton-Krylov-AMG applied to the thin film flow nonlinear system in the time-dependent case.

5.8 Time-Dependent Thin Film Flow Solving in 2D

So far we have considered the steady-state numerical solution of the nonlinear thin film flow system of equations. In this section, we consider the fully discrete thin film flow model in the time-dependent case and we solve it using implicit time stepping combined with the three different nonlinear multilevel solvers under consideration.

We have applied the BDF1 method as the temporal discretisation scheme and the FDM as the spatial discretisation scheme for the model in this work. Subsequently, we applied these three nonlinear algorithms to solve the discrete nonlinear system that results from discretising the nonlinear thin film flow system at each time step.

The system of nonlinear equations for the time-dependent problem, in general, will require a complete set of initial and boundary conditions to determine the numerical solution. Therefore, we apply Dirichlet boundary conditions and initial conditions for this system, as we described earlier in Chapter 2 in Section 2.2.3.

We will describe the three nonlinear multilevel algorithms in the time-dependent problem, in the few next subsections, after that we will compare these algorithms in order to evaluate which multilevel algorithm is the best for solving the time-dependent thin film flow system.

5.8.1 The Fully Discrete System for Thin Film Flow Model in 2D

To obtain the nonlinear algebraic equations that we consider in this subsection we discretise Sellier's model (Equations (2.38) and (2.35)) in time with a fixed step $\triangle t$ to replace the derivative $\frac{\partial h}{\partial t}$. The pressure Equation (2.35) does not change because it contains no time derivative.

The thin film Equation (2.38) reduces to the equations $[F_h(U)] = 0$, where

$$
\begin{aligned}
[F_h(U)]_{i,j} \quad &= \tfrac{h_{i,j}^{n+1} - h_{i,j}^{n}}{\triangle t} \\
-(\frac{1}{\triangle x}) &\left[(\frac{(\frac{h_{i,j}^{n+1} + h_{i+1,j}^{n+1}}{2})^3}{3})((\frac{p_{i+1,j}^{n+1} - p_{i,j}^{n+1}}{\triangle x}) - 2) - (\frac{(\frac{h_{i,j}^{n+1} + h_{i-1,j}^{n+1}}{2})^3}{3})((\frac{p_{i,j}^{n+1} - p_{i-1,j}^{n+1}}{\triangle x}) - 2) \right] \\
-(\frac{1}{\triangle y}) &\left[(\frac{(\frac{h_{i,j}^{n+1} + h_{i,j+1}^{n+1}}{2})^3}{3})(\frac{p_{i,j+1}^{n+1} - p_{i,j}^{n+1}}{\triangle y}) - (\frac{(\frac{h_{i,j}^{n+1} + h_{i,j-1}^{n+1}}{2})^3}{3})(\frac{p_{i,j}^{n+1} - p_{i,j-1}^{n+1}}{\triangle y}) \right]. \qquad (5.55)
\end{aligned}
$$

In the following subsection, we have written in a compact way the expression of the analytical Jacobian matrix in the time-dependent problem which is a critical component in all three nonlinear multilevel algorithms.

5.8.2 The Jacobian Matrix in Time-Dependent Case

The analytical sparse Jacobian in the time-dependent case is almost exactly the same as the analytical Jacobian in the steady-state case in Equation (5.17) that we have illustrated in Section 5.4.2. The only modification is that the block J_{21} which represents the derivative of equation F_h with respect of h has the minor addition of the weighted identity block I as follows,

$$
J_T = \begin{bmatrix} J_{11} & J_{12} \\ (\frac{-1}{\Delta t})I + J_{21} & J_{22} \end{bmatrix}. \qquad (5.56)
$$

In our multilevel schemes, we need this Jacobian for FAS, as a part of smoother, and for Newton's method, which requires the Jacobian matrix at each nonlinear iteration. This can be calculated numerically, using finite differences, but for the FDM discretisation, it is also straightforward, fast and efficient to compute the Jacobian analytically in sparse matrix form. We presented here the analytical expressions for the Jacobian matrix in the time-dependent problem. From Equations (5.55) and (5.16), we can simply derive the Jacobian terms corresponding to the points away from the Dirichlet boundary conditions, with appropriate modifications for points next to the boundary, (see Appendix A). These terms can be used to build the analytical Jacobian efficiently and in the sparse format because all other entries in the Jacobian are zero.

In the next subsection, we demonstrate the performance of the three nonlinear multilevel algorithms for the time-dependent problem which will be described in the numerical results in subsection 5.10.

5.9 Time-Dependent Solvers

In this section, we discuss the three nonlinear solution techniques: FAS, Newton-MG and the new preconditioned Newton-Krylov-AMG algorithm applied to the time-dependent thin film flow problem.

5.9.1 FAS

We used the FAS nonlinear multigrid algorithm in order to solve the thin film flow for the time-dependent problem in 2D. We illustrated earlier the FAS algorithm in detail in Algorithm 2 in Chapter 4. The numerical results of our implementation tests for this algorithm are illustrated in Subsection 5.10.1.

5.9.2 Newton-MG

The application of Newton's method gives a global linearization of the thin film flow nonlinear system. The result of this linearization is the linear system for the correction solution δ, as shown in Equations (5.42) and (5.43). The Newton-MG algorithm as we explained earlier in Section 4.3.3.2 implements Newton iteration as outer iteration then applies linear multigrid iteration as the linear solver. We have presented the numerical results that we generated using Newton-MG algorithm in Subsection 5.10.2.

5.9.3 Newton-Krylov-AMG

We have described earlier the Newton-Krylov-AMG algorithm with the new preconditioner which is one of the main focuses of this thesis. We discuss in this subsection, the algebraic discrete nonlinear systems that arise from the discretisation of the time-dependent problem. These discretization and linearization lead to large sparse linear systems which require to be solved at each nonlinear iteration. Let us rewrite the linear system Equation (5.41) in the following matrix form:

$$\begin{pmatrix} \mathbf{K} & \mathbf{I} \\ \mathbf{B}_{\alpha}^{*} & \mathbf{K}_{\alpha} \end{pmatrix} \begin{pmatrix} \delta_h \\ \delta_p \end{pmatrix} = \begin{pmatrix} F_p \\ F_h \end{pmatrix} \tag{5.57}$$

where $\mathbf{K} = \frac{\partial F_p}{\partial h}$, $\mathbf{I} = \frac{\partial F_p}{\partial p}$, $\mathbf{B}_{\alpha}^{*} = \mathbf{B}_{\alpha} + \frac{-1}{\Delta t}\mathbf{I} = \frac{\partial F_h}{\partial h}$ and $\mathbf{K}_{\alpha} = \frac{\partial F_h}{\partial p}$ are block matrices.

As already observed in the steady-state case, crucial to the performance of a Krylov scheme such as GMRES is a good preconditioner [112]. We consider here potential preconditioners which are based upon the natural extension of P_1 and P_2 to the time-dependent case.
The first preconditioner P_3 in Equation (5.58) in the time-dependent problem uses a direct solver (backslash) with the exact Schur Complement. The advantage of using the exact Schur Complement is that it gives a perfect solution because this preconditioner just requires at maximum two iterations to converge. Nevertheless, clearly, we understood that this is not efficient

since it uses a lot of memory and it is expensive to compute. Consequently, we examined and improved this preconditioner by replacing the exact Schur Complement S_3 by the approximate matrix $S_3 \simeq K_\alpha$ in the different preconditioner P_{3a} and then apply this preconditioner by using the direct solver. The more practical preconditioner P_{3b}, uses one AMG iteration with the approximate Schur Complement $AMG(K_\alpha)$.

The first choice of a preconditioner is P_3 in the time-dependent case, which is the upper triangular block preconditioner written as follows:

$$P_3 = \begin{pmatrix} \mathbf{K} & \mathbf{I} \\ \mathbf{0} & \mathbf{S}_3 \end{pmatrix}, \tag{5.58}$$

where S_3 is the exact Schur Complement $S_3 = K_\alpha - B_\alpha^* K^{-1} I$.

The preconditioner P_{3a} is the upper triangular block preconditioner formulated as follows:

$$P_{3a} = \begin{pmatrix} \mathbf{K} & \mathbf{I} \\ \mathbf{0} & \mathbf{K}_\alpha \end{pmatrix}, \tag{5.59}$$

where $S_3 \simeq K_\alpha$ approximates the block matrix.

The preconditioner P_{3b} is the upper triangular block preconditioner written as follows:

$$P_{3b} = \begin{pmatrix} \mathbf{AMG(K)} & \mathbf{I} \\ \mathbf{0} & \mathbf{AMG(K}_\alpha\mathbf{)} \end{pmatrix}, \tag{5.60}$$

where **AMG(X)** represents the action of one AMG iteration applied to matrix X.

Our second choice of a preconditioner in the time-dependent case is P_4, the lower triangular block preconditioner written as follows:

$$P_4 = \begin{pmatrix} \mathbf{S}_4 & \mathbf{0} \\ \mathbf{B}_\alpha^* & \mathbf{K}_\alpha \end{pmatrix}, \tag{5.61}$$

where S_4 is the exact Schur Complement $S_4 = K - I K_\alpha^{-1} B_\alpha^*$.

The preconditioner P_{4a} is given by the block lower triangular matrix as follows:

$$P_{4a} = \begin{pmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{B}_\alpha^* & \mathbf{K}_\alpha \end{pmatrix}, \tag{5.62}$$

where $S_4 \simeq K$ the approximate of the Schur Complement.

The preconditioner P_{4b} is given by the block lower triangular matrix as follows:

$$P_{4b} = \begin{pmatrix} \mathbf{AMG(K)} & \mathbf{0} \\ \mathbf{B}_{\alpha}^{*} & \mathbf{AMG(K}_{\alpha}\mathbf{)} \end{pmatrix}. \tag{5.63}$$

We again use the software implementation that is available in Harwell Subroutine Library (HSL). This includes HSL-MI20 for the AMG method and HSL-MI24 for the GMRES method. The numerical results of the Newton-Krylov-AMG algorithm preconditioned with P_3, P_{3a}, P_{3b}, P_4, P_{4a} and P_{4b} are presented in the subsection 5.10.3. In the following section, we present the numerical results that have been reached using the FAS, Newton-MG and Newton-Krylov-AMG nonlinear multilevel algorithms for the time-dependent nonlinear problem.

5.10 Numerical Results in Time-Dependent Case

The computational results that are presented in this section have used the same scenarios and grid sizes that we used in the steady-state problem. We have implemented extensive numerical experiments to optimize the parameters selected for the all three nonlinear multilevel algorithms which are: FAS, Newton-MG and Newton-Krylov-AMG with new preconditioner. Several choices of the parameters were employed for each scheme. Moreover, in subsection 5.10.4, we will compare our selection of parameters for these three nonlinear multilevel schemes to decide the best among them.

Figure 5.19 shows the initial conditions that we have used for the thin film flow system. We display the numerical solutions for the thin film flow system at $T = 5$ (the end of the solution) in Figure 5.20. Throughout this subsection, the numerical results have been completed using the time step size $\Delta t = 0.1$ and the number of time steps taken to estimate the performance of the three nonlinear multilevel algorithms is always 10.

We have performed comprehensive numerical experiments to optimize the parameter selection for all three nonlinear multilevel schemes to solve this system. In the time-dependent case and the results that follow explore optimal choices for each method.

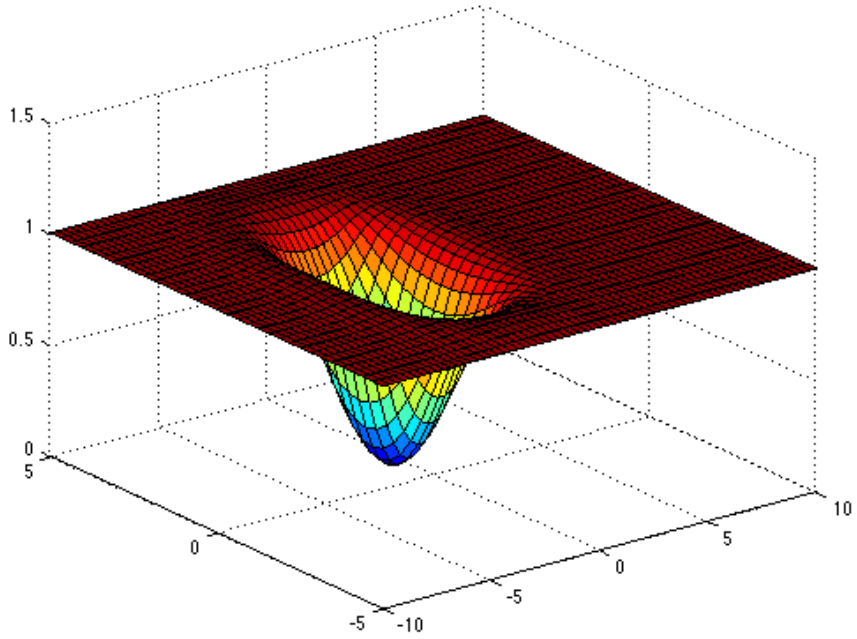


Figure 5.19: The free surface (numerical solution) $h + s$ at Time T=0.

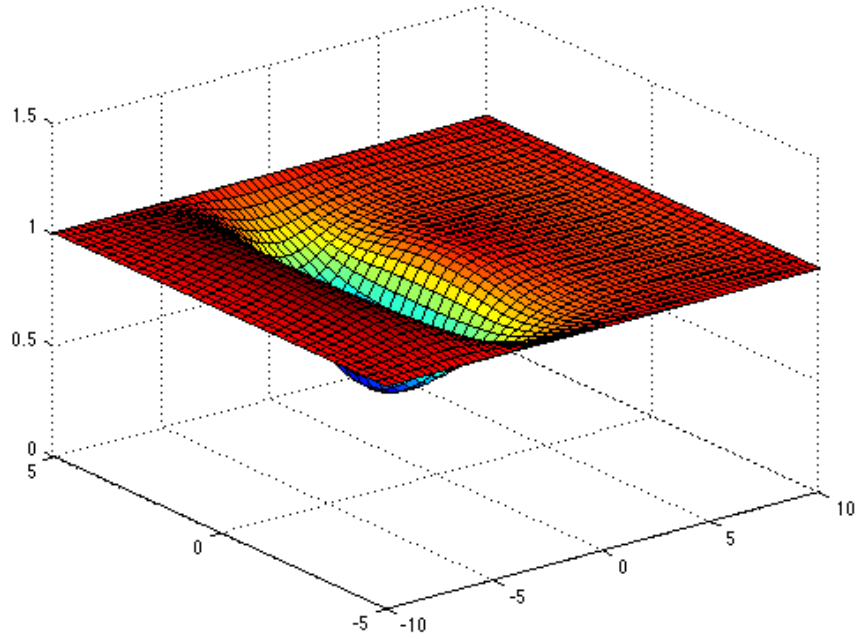


Figure 5.20: The free surface (numerical solution) $h + s$ at Time T=5.

5.10.1 Numerical Results using FAS

This subsection focuses on the numerical results of our nonlinear multigrid FAS solver in the time-dependent case. We start with the numerical results of the FAS algorithm with Jacobi smoother. We undertake numerical experiments of this algorithm with different values of ω. Then later we test the same algorithm with Red-Black-G-S smoother and other values of ω. We conclude by determining the best parameters for the FAS algorithm.

FAS with Jacobi smoother and varying damping factor ω

From Tables 5.112 to 5.115 we apply varying ω in the range of $\omega = 1/2,\ 2/3,\ 0.8$ and 0.9, with the purpose of finding the optimal value for the nonlinear Jacobi smoother with a $(1, 1)$ pre- and post-smoother and the coarsest grid level $G = 3$. We observed that the best selection for the free parameter ω for the Jacobi smoother is $\omega = 0.8$ since for ω less or greater than 0.8 the performance of FAS is poorer in terms of the number of V-cycles and the overall time of this algorithm. Therefore, we used $\omega = 0.8$ in the FAS algorithm with the Jacobi smoother as the best choice.

Table 5.112: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps=10, by using nonlinear Jacobi with $\omega = 1/2$ and the relative tolerance for FAS is Tol $= 1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	28	30	28.9	9.6797
6	3	1	1	29	32	30.9	33.2597
7	3	1	1	30	34	32.0	136.2223
8	3	1	1	31	35	32.8	564.2420
9	3	1	1	31	35	33.2	2.2962e+03

Table 5.113: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps $= 10$, by using nonlinear Jacobi with $\omega = 2/3$ and the relative tolerance for FAS is Tol $= 1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	20	22	20.7	7.3419
6	3	1	1	22	23	22.4	23.0547
7	3	1	1	22	25	23.4	93.5495
8	3	1	1	23	25	24.0	390.6789
9	3	1	1	23	26	24.2	1.5753e+03

Table 5.114: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps $= 10$, by using nonlinear Jacobi with $\omega = 0.8$ and the relative tolerance for FAS is Tol $= 1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	16	17	16.9	4.9573
6	3	1	1	18	19	18.3	18.2336
7	3	1	1	19	20	19.3	74.5960
8	3	1	1	19	21	19.7	307.1344
9	3	1	1	19	21	20.0	1.2785e+03

Table 5.115: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, by using nonlinear Jacobi with $\omega = 0.9$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	27	33	28.5	9.7206
6	3	1	1	26	35	28.2	28.3622
7	3	1	1	25	33	26.7	104.4816
8	3	1	1	23	36	25.4	404.4691
9	3	1	1	21	34	23.4	1.5339e+03

FAS with Jacobi smoother and varying *Coarse grid level*

We now seek the best coarse grid for the FAS algorithm with the Jacobi smoother. By examining the data in Tables 5.114, 5.116 and Table 5.117, we observed that the best coarse grid level for this problem is $G = 5$.

Table 5.116: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, by using nonlinear Jacobi with $\omega = 0.8$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	4	1	1	15	17	16.1	6.0567
6	4	1	1	17	19	17.8	19.5984
7	4	1	1	17	20	18.6	80.9420
8	4	1	1	18	21	19.1	333.7422
9	4	1	1	18	21	19.5	1.3757e+03

Table 5.117: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, by using nonlinear Jacobi with $\omega = 0.8$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
6	5	1	1	15	17	15.8	21.6336
7	5	1	1	16	18	17.0	74.5368
8	5	1	1	17	18	17.9	308.3932
9	5	1	1	17	19	18.0	1.2607e+03

FAS with Red Black Gauss-Seidel smoother and varying damping factor ω

We used an alternative smoother for the FAS algorithm which is the Red-Black Gauss-Seidel smoother; we vary the values ω and $(pre, post)_{smooth}$ with relative tolerance Tol = $1e-8$ in order to compare the data obtained from this smoother with the data obtained from the Jacobi smoother.

In Tables 5.118 to 5.122 we present the results of the numerical solution of the thin film flow system in the time-dependent using the FAS algorithm with the Red-Black Gauss-Seidel smoother; with $\omega = 0.8, 0.9, 1, 1.1, 1.2$, $(pre, post)_{smooth} = (1, 1)$ and the coarse grid $G = 3$. In Table 5.120 the best choice of ω is presented, $\omega = 1$ with the basis of $(1, 1)$ V-cycle and $G = 3$. These results again show that the Gauss-Seidel smoother seems to out-perform the Jacobi smoother.

Table 5.118: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 0.8$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	13	13	13.0	4.4897
6	3	1	1	14	14	14.0	17.5544
7	3	1	1	14	15	14.4	69.8695
8	3	1	1	14	15	14.9	291.1552
9	3	1	1	14	16	15.1	1.2169e+03

Table 5.119: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 0.9$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	11	11	11.0	3.8332
6	3	1	1	11	12	11.4	14.3914
7	3	1	1	12	12	12.0	58.5282
8	3	1	1	12	13	12.1	237.9691
9	3	1	1	12	13	12.3	1.0040e+03

Table 5.120: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	9	10	9.70	3.3760
6	3	1	1	10	10	10.0	12.5476
7	3	1	1	10	10	10.0	48.8703
8	3	1	1	10	10	10.0	197.8073
9	3	1	1	10	10	10.0	812.6604

Table 5.121: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 1.1$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	9	11	10.3	3.5917
6	3	1	1	9	11	10.7	13.4578
7	3	1	1	9	11	10.7	52.1803
8	3	1	1	9	11	10.7	210.9789
9	3	1	1	9	11	10.7	871.6197

Table 5.122: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 1.2$ and the relative tolerance for FAS is Tol $= 1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	11	13	12.7	4.7572
6	3	1	1	11	14	12.9	16.2018
7	3	1	1	11	14	13.0	63.2957
8	3	1	1	11	14	13.0	255.1201
9	3	1	1	11	14	13.0	1.0513e+03

FAS with Red Black Gauss-Seidel smoother and varying *Coarse grid level*

We seek the best coarse grid for the FAS algorithm with the Red-Black Gauss-Seidel smoother. By studying the data in Tables 5.123, 5.124 and Table 5.125, we see that the best coarse grid level for this problem is $G = 5$ in Table 5.124.

Table 5.123: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol $= 1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	4	1	1	8	8	8.00	3.7844
6	4	1	1	9	10	9.20	11.5405
7	4	1	1	9	10	9.90	47.4262
8	4	1	1	10	11	10.4	202.5769
9	4	1	1	10	11	10.5	841.7728

Table 5.124: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
6	5	1	1	7	8	7.6	11.4158
7	5	1	1	8	9	8.8	43.3668
8	5	1	1	9	10	9.4	180.2312
9	5	1	1	9	10	9.6	746.0546

Table 5.125: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
7	6	1	1	7	7	7.0	48.2318
8	6	1	1	8	9	8.1	177.2111
9	6	1	1	8	9	8.7	748.6868

FAS with Red Black Gauss-Seidel smoother and varying $(pre, post)_{smooth}$

We used grid level $G = 5$, in Table 5.126 and Table 5.127, when we are comparing the number of pre- and post-smooth V-cycles for this algorithm. We use $(pre, post)_{smooth} = (2, 1)$, and $(pre, post)_{smooth} = (2, 2)$ with $\omega = 1$ and $G = 5$. From these numerical results, we found that the best $(pre, post)_{smooth}$ is $(1, 1)$ in Table 5.124. This is because it delivers converged results in a lower time, despite requiring a greater number of FAS V-cycles per time step.

Table 5.126: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
6	5	2	1	7	7	7.0	13.6032
7	5	2	1	7	8	7.9	55.9508
8	5	2	1	8	9	8.1	228.2863
9	5	2	1	8	9	8.5	961.5642

Table 5.127: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol $= 1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
6	5	2	2	6	7	6.3	15.1146
7	5	2	2	7	7	7.0	63.2483
8	5	2	2	7	8	7.6	272.5561
9	5	2	2	7	8	7.9	1.1651e+03

FAS with Red Black Gauss-Seidel smoother and varying damping factor ω with $\triangle t = 0.5$

In order to consider the impact of the choices of the time step size on parameter choices Tables 5.128 to 5.129 illustrate the numerical results using the FAS algorithm with the Red-Black Gauss-Seidel smoother; $\omega = 1$, 1.2, with $\triangle t = 0.5$, $(pre, post)_{smooth} = (1, 1)$ and the coarse grid $G = 3$. As we can see from these tables when we have done some tests with a bigger value of the time step $\triangle t$, for example as $\triangle t = 0.5$, we observed that the optimal value of ω is $\omega = 1.2$ which is the same behaviour of the performance of the steady-state problem. However, when we take a smaller value of the time step, for example, $\triangle t = 0.1$, then we have a different optimal value of ω.

Table 5.128: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.5$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol $= 1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	10	10	10.00	5.5976
6	3	1	1	11	11	11.00	14.6976
7	3	1	1	11	12	10.10	55.3000
8	3	1	1	11	12	11.90	236.3196
9	3	1	1	11	12	11.90	967.7897

Table 5.129: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.5$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 1.2$ and the relative tolerance for FAS is Tol $= 1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	9	11	10.40	3.8698
6	3	1	1	9	11	10.60	12.9924
7	3	1	1	9	11	10.80	52.1182
8	3	1	1	9	11	10.80	211.3831
9	3	1	1	9	11	10.80	873.2995

Summary

All numerical results of our FAS algorithm that are presented have confirmed that its performance is close to optimal across a wide range of impact parameters. The timings show that our FAS implementation scales linearly with the number of unknowns.

Furthermore, for $\triangle t = 0.1$, the test in Table 5.124 shows that the FAS algorithm with the nonlinear Red-Black Gauss-Seidel smoother with parameter $\omega = 1$, $(pre, post)_{smooth} = (1, 1)$, $Coarse\ Grid = 5$ and the relative tolerance Tol $= 1e-8$ is the best choice based on all the experiments that we have performed. Although the optimal choice of ω for this time step size is different to that observed in the steady-state case, when $\triangle t$ is increased the optimal parameter size for ω becomes the same as for the steady-state case.

5.10.2 Numerical Results using Newton-MG

In this subsection, we consider the numerical solutions provided by the Newton-MG solver in the time-dependent problem in 2D.

Newton-MG with Jacobi smoother and varying damping factor ω

In Tables 5.130 to 5.133 ω is varied in the range of $\omega = 1/2$, $2/3$, 0.8 and 0.9, with the aim of determining the optimal value for the linear Jacobi smoother with a $(1, 1)$ pre- and post-smoother and with the coarsest grid level $G = 3$. We found that the best selection for the free parameter ω for the Jacobi smoother is $\omega = 0.8$ and if ω is less or greater than 0.8 the performance of Newton-MG is poorer as regards the number of V-cycles and the total time of the algorithm. Consequently, we applied $\omega = 0.8$ for the Newton-MG algorithm with the Jacobi smoother in order to determine the best values for other free parameters.

Table 5.130: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, by using Jacobi with $\omega = 1/2$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
5	3	1	1	3	11	13	11.0	7.4568
6	3	1	1	3	11	15	13.3	29.8859
7	3	1	1	3	12	15	13.1	130.2145
8	3	1	1	3	12	16	13.8	558.9905
9	3	1	1	3	13	17	14.4	2.3552e+03

Table 5.131: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, by using Jacobi with $\omega = 2/3$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
5	3	1	1	3	7	10	8.10	5.3074
6	3	1	1	3	8	11	8.80	21.9305
7	3	1	1	3	9	12	9.70	98.0223
8	3	1	1	3	9	12	10.2	420.4773
9	3	1	1	3	10	13	10.7	1.7997e+03

Table 5.132: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, by using Jacobi with $\omega = 0.8$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
5	3	1	1	3	6	8	6.6	5.9891
6	3	1	1	3	7	9	7.4	18.7619
7	3	1	1	3	7	10	7.9	81.3211
8	3	1	1	3	8	10	8.5	353.1812
9	3	1	1	3	8	10	8.7	1.4731e+03

Table 5.133: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, by using Jacobi with $\omega = 0.9$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
5	3	1	1	3	10	15	11.1	7.1385
6	3	1	1	3	10	17	11.6	28.7511
7	3	1	1	3	10	17	11.7	117.5133
8	3	1	1	3	10	18	11.7	478.9011
9	3	1	1	3	10	18	11.6	1.9431e+03

Newton-MG with Jacobi smoother and varying *Coarse grid level*

In Tables 5.134 to 5.135 we apply Newton-MG with varying coarsest grid level which are $Coarse\ Grid = 4$ and $Coarse\ Grid = 5$ with $\omega = 0.8$ for the Jacobi smoothing parameter. We see that there is only a small difference between these results and these in Table 5.132, however, $Coarse\ Grid = 5$ appears to be the best choice.

Table 5.134: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, by using Jacobi with $\omega = 0.8$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
5	4	1	1	3	5	8	6.4	5.6777
6	4	1	1	3	6	9	7.2	18.0794
7	4	1	1	3	7	10	7.8	80.2832
8	4	1	1	3	7	10	8.2	347.6653
9	4	1	1	3	7	11	8.6	1.4709e+03

Table 5.135: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, by using Jacobi with $\omega = 0.8$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
6	5	1	1	3	6	8	6.5	16.3939
7	5	1	1	3	6	9	7.1	72.8956
8	5	1	1	3	7	10	7.0	335.2895
9	5	1	1	3	7	10	8.0	1.4192e+03

Newton-MG with Red Black Gauss-Seidel smoother and varying damping factor ω

We now consider using Gauss-Seidel as the linear MG smoother. In Tables 5.136 to Table 5.140 we display the numerical results of Newton-MG performance with the Red-Black G-S smoother for a varying ω in the range of $\omega = 0.8, 0.9, 1, 1.1$ and 1.2, with the goal of determining the optimal value for the smoother with a $(1, 1)$ pre- and post-smoother. We see that $\omega = 1$ is the best choice, Table 5.138. Furthermore, this performance is better than the Jacobi smoother.

Table 5.136: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, Red Black G-S with $\omega = 0.8$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
5	3	1	1	3	5	6	5.2	5.4253
6	3	1	1	3	5	7	5.6	14.8932
7	3	1	1	3	5	7	6.0	65.8999
8	3	1	1	3	6	8	6.4	278.6617
9	3	1	1	3	6	8	6.6	1.1724e+03

Table 5.137: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, Red Black G-S with $\omega = 0.9$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
5	3	1	1	3	4	6	4.5	4.8026
6	3	1	1	3	4	6	4.9	12.2991
7	3	1	1	3	5	7	5.3	53.9997
8	3	1	1	3	5	7	5.5	229.8439
9	3	1	1	3	5	7	5.7	983.5401

Table 5.138: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, Red Black G-S with $\omega = 1$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
5	3	1	1	3	4	5	4.2	4.4412
6	3	1	1	3	4	6	4.4	11.4054
7	3	1	1	3	4	6	4.5	47.6886
8	3	1	1	3	4	6	4.7	204.0510
9	3	1	1	3	4	6	4.9	878.4558

Table 5.139: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, Red Black G-S with $\omega = 1.1$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
5	3	1	1	3	4	5	4.2	3.1313
6	3	1	1	3	4	5	4.3	11.2437
7	3	1	1	3	4	6	4.5	48.7204
8	3	1	1	3	4	6	4.5	200.3037
9	3	1	1	3	5	6	5.1	929.7864

Table 5.140: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, Red Black G-S with $\omega = 1.2$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
5	3	1	1	3	5	6	5.1	3.6864
6	3	1	1	3	5	6	5.2	13.4801
7	3	1	1	3	5	6	5.3	56.8154
8	3	1	1	3	6	7	6.1	267.1056
9	3	1	1	3	6	7	6.1	1.1014e+03

Newton-MG with Red Black Gauss-Seidel smoother and varying *Coarse grid level*

Our aim here is to find the best coarse grid for the Newton-MG algorithm with the Red-Black G-S smoother. By examining Table 5.141, 5.142 and 5.143 we found that the best coarse grid level for this problem is $Coarse\ Grid = 6$. This value is quite high, however, we note that the "backslash" solver in Matlab is able to exploit the sparsity of the Jacobian matrix on this grid.

Table 5.141: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, Red Black G-S with $\omega = 1$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
5	4	1	1	3	3	5	3.7	2.7224
6	4	1	1	3	4	5	4.2	10.6236
7	4	1	1	3	5	6	4.6	48.4196
8	4	1	1	3	5	6	4.8	209.6136
9	4	1	1	3	4	6	5.0	902.6206

Table 5.142: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, Red Black G-S with $\omega = 1$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol = $1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
6	5	1	1	3	3	5	3.8	9.8680
7	5	1	1	3	4	6	4.4	45.4568
8	5	1	1	3	4	6	4.6	199.3704
9	5	1	1	3	4	6	4.8	867.0142

Table 5.143: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, Red Black G-S with $\omega = 1$, fixed V-cycle, the number of $V-cycle = 3$ and the relative tolerance for Newton is Tol = $1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
7	6	1	1	3	3	5	3.8	39.8166
8	6	1	1	3	4	5	4.2	182.4501
9	6	1	1	3	4	6	4.4	793.0276

Newton-MG with Red Black Gauss-Seidel smoother and varying $(pre, post)_{smooth}$

In Table 5.144 and Table 5.145 we experimented with the grid level $G = 5$ with two values of pre- and post-smooth in order to determine the best value; we experimented with the following $(pre, post)_{smooth}$ values: $(2, 1)$ and $(2, 2)$. We found that the best value of the $(pre, post)_{smooth}$ in this algorithm is $(pre, post)_{smooth} = (1, 1)$, as we can see in Table 5.142.

Table 5.144: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, Red Black G-S with $\omega = 1$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol = $1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
6	5	2	1	3	3	5	3.6	12.1579
7	5	2	1	3	3	5	4.1	58.5123
8	5	2	1	3	4	5	4.2	257.2189
9	5	2	1	3	4	6	4.4	1.0977e+03

Table 5.145: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, Red Black G-S with $\omega = 1$, fixed V-cycle, the number of $Vcycle = 3$ and the relative tolerance for Newton is Tol = $1e-8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
$Fine$	$Coarse$	Pre	$Post$	No. MG V-Cycle	Min	Max	$Average$	-
6	5	2	2	3	3	5	3.5	14.4514
7	5	2	2	3	3	5	3.8	69.0380
8	5	2	2	3	3	5	4.1	314.7424
9	5	2	2	3	4	5	4.2	1.3230e+03

Summary

We observe that in all cases the number of linear V-cycles stays approximately constant as the grid level is increased. In addition, when the problem size is grown by a factor of 4 the total time is also grown by a factor of 4 which indicates a linear time complexity of $O(N)$.

When we compare the best results of the Newton-MG algorithm with Jacobi smoother with parameter $\omega = 0.8$, $(pre, post)_{smooth} = (1, 1)$, coarse grid $Coarse\ Grid = 5$ and the relative tolerance Tol = $1e-8$ in Table 5.135 with the best results of the Newton-MG algorithm with Red-Black G-S smoother with $\omega = 1$, $(pre, post)_{smooth} = (1, 1)$, coarse grid $Coarse\ Grid = 6$ and the relative tolerance Tol = $1e-8$ in Table 5.143, we conclude that the Newton-MG algorithm with Red-Black G-S smoother is the best in this case.

5.10.3 Numerical Results using Newton-Krylov-AMG

We consider the numerical solutions produced by using the Newton-Krylov-AMG with our new preconditioner to solve the thin film flow system in the time-dependent problem in 2D in this subsection. We show extensive numerical experiments to optimize the parameter selection for the Newton-Krylov-AMG.

Newton-Krylov-AMG with the Preconditioner P_3

Table 5.146 shows the preconditioner P_3 in Equation (5.58) with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 20$, $Tol = 1e-4$ for GMRES and $Restart = 20$ (i.e. if 20 iterations were to be taken then GMRES would restart: by fixing this parameter to be the same as Maxit we are ensuring no restart). This preconditioner is an optimal preconditioner in terms of iterations, it is prohibitively expensive, however, because we solve the diagonal blocks with direct linear algebra and the computation of the exact Schur Complement matrix.

Table 5.146: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_3 preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 20$, $Tol = 1e-4$ and $Restart = 20$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	4	3.3	1	2	1.03	20.0886
6	3	4	3.3	1	2	1.03	743.6201

Newton-Krylov-AMG with the Preconditioner P_{3a} and varying GMRES Tol

To improve the efficiency of the preconditioner, we took the upper triangular blocks of the matrix J and replaced the exact Schur Complement with an approximation, i.e. the block matrix $S_3 \simeq K_\alpha$ in the preconditioner P_{3a} in Equation (5.59). The numerical result of Tables 5.147, 5.148 and 5.149 show the performance of Newton-Krylov with the P_{3a} preconditioner with different tolerances for the GMRES iterations. The power of this preconditioner is that it has a constant number of iterations as reported in the previous tables. It is still relatively slow, however, since we solve the diagonal blocks with direct linear algebra (backslash).

Table 5.147: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{3a} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 20$, $Tol = 1e-2$ and $Restart = 20$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	6	5.1	7	14	11.70	3.0421
6	5	6	5.2	7	14	11.65	16.8074
7	5	6	5.4	7	14	11.75	69.4758
8	5	7	5.6	7	14	11.76	350.2836

Table 5.148: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{3a} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 20$, $Tol = 1e-3$ and $Restart = 20$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	4.0	10	17	14.29	2.8900
6	4	5	4.2	10	17	14.42	15.8726
7	4	5	4.2	10	17	14.35	64.2262
8	4	5	4.3	10	17	14.41	320.6469

Table 5.149: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{3a} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 20$, $Tol = 1e-4$ and $Restart = 20$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	3.5	12	19	16.97	4.8324
6	3	5	3.5	12	19	16.91	15.8431
7	3	5	3.7	12	20	16.91	65.5000
8	3	5	3.7	12	20	16.97	321.3882

Newton-Krylov-AMG with the Preconditioner P_{3b}, $(pre, post)_{smooth} = (1, 1)$ and varying GMRES Tol

As we can see from Tables 5.147, 5.148 and 5.149, the results displayed confirm a good performance of the P_{3a} preconditioner with respect to the GMRES iteration count, which is constant for all cases. On the other hand, the P_{3a} preconditioner is not optimal with respect to running time, due to the fact that we solve the diagonal blocks with direct linear algebra (backslash). Consequently, we require to improve this preconditioner to decrease the time needed to solve this system. In order to develop the efficiency of the algorithm, we implemented our new preconditioner P_{3b} in Equation (5.60) by solving the diagonal blocks approximately using one AMG V-cycle with Gauss-Seidel smoothing. Tables 5.150 to 5.153 demonstrate a series of experiments using P_{3b} in Equation (5.60) where we varied the tolerance for GMRES between Tol $= 1e-02$, Tol $= 1e-03$, Tol $= 1e-04$ and Tol $= 1e-05$ with Tol $= 1e-08$ as the tolerance for Newton iterations with $(pre, post)_{smooth} = (1, 1)$. We can see that all four choices yield close to optimal, $O(N)$ algorithms, however, as we can see in Table 5.152, Tol $= 1e-04$ represents the best choice.

Table 5.150: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{3b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-2$ and $Restart = 20$ with $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	5	6	5.3	8	16	12.96	1.8212
6	5	7	5.5	8	17	13.07	7.7314
7	6	7	6.1	7	17	13.11	30.2351
8	6	7	6.2	8	17	13.04	126.0949
9	6	7	6.2	8	17	12.88	538.1471

Table 5.151: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{3b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-3$ and $Restart = 20$ with $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	5	4.2	12	20	16.09	1.8333
6	4	5	4.2	12	20	16.04	6.8254
7	4	6	4.4	12	20	16.06	25.0371
8	4	6	4.6	12	20	16.02	107.2297
9	4	6	4.8	12	20	16.04	485.2609

Table 5.152: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{3b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-4$ and $Restart = 20$ with $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	3.5	16	27	18.80	1.7292
6	3	5	3.7	16	27	19.00	6.7337
7	3	5	3.9	16	27	18.92	24.8841
8	4	5	4.1	16	26	18.95	107.3758
9	4	5	4.2	16	26	18.80	480.2390

Table 5.153: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{3b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-5$ and $Restart = 20$ with $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	4	3.3	18	33	23.18	1.9120
6	3	5	3.5	17	32	23.05	8.2297
7	3	5	3.6	18	32	22.97	27.8777
8	3	5	3.6	18	32	22.69	108.4429
9	3	5	3.7	18	31	22.67	488.7844

Newton-Krylov-AMG with the Preconditioner P_{3b} and $(pre, post)_{smooth} = (2, 1)$

Table 5.154 shows the performance of the Newton-Krylov-AMG with preconditioner P_{3b} with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-4$ and $Restart = 20$ and with $(pre, post)_{smooth} = (2, 1)$ for the P_{3b} preconditioner. Whilst this is still optimal it is slightly slower than the $(pre, post)_{smooth} = (1, 1)$ case in Table 5.152.

Table 5.154: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{3b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-4$ and $Restart = 20$ with $(pre, post)_{smooth} = (2, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	3.5	15	26	18.42	1.7551
6	3	5	3.5	16	26	18.54	6.4956
7	3	5	3.7	15	26	18.59	24.1367
8	3	5	4.0	15	26	18.66	107.6126
9	4	5	4.1	16	25	18.58	483.2393

Newton-Krylov-AMG with the Preconditioner P_{3b} and $(pre, post)_{smooth} = (2, 2)$

Table 5.155 presents an experiment with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-4$ and $Restart = 20$ and with $(pre, post)_{smooth} = (2, 2)$ for the P_{3b} preconditioner. Again the result is close to optimal but slower than the $(pre, post)_{smooth} = (1, 1)$ case (see Table 5.152).

Table 5.155: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{3b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-4$ and $Restart = 20$ with $(pre, post)_{smooth} = (2, 2)$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	3.5	15	25	18.31	1.6221
6	3	5	3.5	16	25	18.54	6.6744
7	3	5	3.7	16	25	18.52	25.0829
8	3	5	3.7	15	25	18.51	103.5058
9	4	5	4.1	16	25	18.53	493.0444

Now we repeat all experiments of the preconditioner for P_3 to the preconditioner for P_4.

Newton-Krylov-AMG with the Preconditioner P_4

Table 5.156 presents the solver with the preconditioner P_4 in Equation (5.61) with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 20$, $Tol = 1e-4$ and $Restart = 20$. As expected, whilst this preconditioner is optimal in terms of iterations, it is prohibitively expensive due to the computation of the exact Schur Complement matrix and direct linear algebra.

Table 5.156: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_4 preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 20$, $Tol = 1e-4$ and $Restart = 20$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	4	3.3	1	2	1.96	31.6763
6	3	4	3.3	1	2	1.96	1.0292e+03

Newton-Krylov-AMG with the Preconditioner P_{4a} and varying GMRES Tol

In order to improve the efficiency of the preconditioner, we took the lower triangular blocks of the matrix J and replaced the exact Schur complement with an approximation, i.e. the block matrix $S4 \simeq K$ in the preconditioner P_{4a} in Equation (5.62). The numerical results of Tables 5.157, 5.158 and 5.159 display the performance of Newton-Krylov with the P_{4a} preconditioner with different tolerances for the GMRES iterations. It is still relatively slow, due to the fact that we solve the diagonal blocks using direct linear algebra. Nevertheless, the advantage of

this preconditioner is that it still has a constant number of iterations as shown in the previous tables.

Table 5.157: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4a} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 20$, $Tol = 1e-2$ and $Restart = 20$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	5	7	5.4	7	14	11.46	3.2134
6	5	7	5.7	7	14	11.61	17.5422
7	5	7	5.9	7	14	11.71	74.8088
8	6	7	6.1	7	14	11.78	377.0737

Table 5.158: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4a} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 20$, $Tol = 1e-3$ and $Restart = 20$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	5	4.1	10	17	14.43	2.9346
6	4	6	4.4	10	17	14.56	16.6231
7	4	6	4.4	10	17	14.54	68.4870
8	4	6	4.5	10	17	14.55	338.7543

Table 5.159: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4a} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 20$, $Tol = 1e-4$ and $Restart = 20$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	3.6	12	20	17.19	2.8957
6	3	5	3.6	12	19	17.11	15.8467
7	3	5	3.7	12	19	17.13	65.9065
8	3	5	3.8	12	19	17.18	331.7706

Newton-Krylov-AMG with the Preconditioner P_{4b}, $(pre, post)_{smooth} = (1, 1)$ and varying GMRES Tol

Similarly, we analyse the performance of the lower triangular block preconditioner P_{4b} in Equation (5.63) where the diagonal blocks systems are solved by using one AMG V-cycle with Gauss-Seidel smoothing and varying pre- and post-smoothing iterations. Tables 5.160 to 5.163 display a series of experiments where we varied the tolerance of GMRES between Tol $= 1e-02$, Tol $= 1e-03$, Tol $= 1e-04$ and Tol $= 1e-05$, with $\triangle t = 0.1$ and the number of time step $= 10$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-2$ and $Restart = 20$, where the tolerance of Newton iterations is Tol $= 1e-08$ with $(pre, post)_{smooth} = (1, 1)$ for the P_{4b} preconditioner. Table 5.162 shows the best performance of the Newton-Krylov-AMG for the P_{4b} preconditioner with Tol $= 1e-04$ of GMRES iterations with $(pre, post)_{smooth} = (1, 1)$.

Table 5.160: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4b} preconditioner with $\triangle t = 0.1$ and the number of time step $= 10$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-2$ and $Restart = 20$ with $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	5	7	5.6	8	15	12.85	1.8731
6	6	7	6.1	8	15	12.95	8.4716
7	6	8	6.2	8	15	12.95	30.4000
8	6	8	6.6	8	15	12.87	132.8622
9	7	8	7.1	8	15	12.73	610.7726

Table 5.161: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4b} preconditioner with $\triangle t = 0.1$ and the number of time step $= 10$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-3$ and $Restart = 20$ with $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	5	4.2	12	18	16.09	1.6516
6	4	6	4.3	12	18	16.02	6.9997
7	4	6	4.6	13	17	16.06	26.1467
8	4	6	4.8	13	17	16.02	111.1767
9	5	6	5.1	12	17	16.02	511.5625

Table 5.162: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-4$ and $Restart = 20$ with $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	3.7	15	21	18.72	1.8223
6	3	5	3.9	15	20	18.79	7.0630
7	4	5	4.1	16	20	18.82	25.9074
8	4	5	4.2	16	20	18.82	109.0045
9	4	5	4.2	16	20	18.73	476.1476

Table 5.163: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-5$ and $Restart = 20$ with $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	3.5	19	28	23.37	1.8252
6	3	5	3.5	19	27	23.05	7.4963
7	3	5	3.6	18	27	23.00	27.1795
8	3	5	3.8	19	27	22.91	115.6118
9	3	5	3.9	18	27	22.82	667.1306

Newton-Krylov-AMG with the Preconditioner P_{4b}, $(pre, post)_{smooth} = (2, 1)$ and varying GMRES Tol

Table 5.164, Table 5.165 and Table 5.166 display the performance of the Newton-Krylov-AMG with preconditioner P_{4b} with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Restart = 20$, Tol $= 1e-2$, Tol $= 1e-3$ and Tol $= 1e-4$ and with $(pre, post)_{smooth} = (2, 1)$ for the P_{4b} preconditioner. As with P_{3b} this smoothing cycle is always slower than the $(pre, post)_{smooth} = (1, 1)$ case.

Table 5.164: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-2$ and $Restart = 20$ with $(pre, post)_{smooth} = (2, 1)$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	5	7	5.8	8	14	12.63	1.9969
6	6	7	6.1	8	14	12.67	8.6558
7	6	7	6.2	8	14	12.67	31.0850
8	6	8	6.4	8	14	12.68	132.0033
9	6	8	6.9	8	14	12.57	607.8185

Table 5.165: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-3$ and $Restart = 20$ with $(pre, post)_{smooth} = (2, 1)$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	5	4.2	12	17	15.90	1.6929
6	4	5	4.2	12	17	15.90	7.0146
7	4	6	4.4	12	17	15.88	25.6084
8	4	6	4.6	12	17	15.87	109.8941
9	4	6	5.0	12	17	15.80	515.3217

Table 5.166: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-4$ and $Restart = 20$ with $(pre, post)_{smooth} = (2, 1)$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	3.6	16	20	18.52	1.6247
6	3	5	3.8	15	20	18.57	7.0786
7	3	5	3.9	15	20	18.64	25.3014
8	3	5	4.1	16	20	18.58	109.3549
9	4	5	4.2	15	20	18.51	687.2969

Newton-Krylov-AMG with the Preconditioner P_{4b}, $(pre, post)_{smooth} = (2, 2)$ and varying GMRES Tol

Tables 5.167 to Table 5.169 show the performance of the Newton-Krylov-AMG with preconditioner P_{4b} with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Restart = 20$, Tol $= 1e-2$, Tol $= 1e-3$ and Tol $= 1e-4$ and with $(pre, post)_{smooth} = (2, 2)$ for the P_{4b} preconditioner. Again this is always slower than the $(pre, post)_{smooth} = (1, 1)$ case (see Table 5.162).

Table 5.167: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-2$ and $Restart = 20$ with $(pre, post)_{smooth} = (2, 2)$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	5	7	6.0	7	14	12.51	2.2768
6	6	7	6.1	8	14	12.52	8.9916
7	6	7	6.1	8	14	12.56	32.7711
8	6	8	6.3	7	14	12.50	136.2320
9	6	8	6.4	8	14	12.53	742.2679

Table 5.168: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-3$ and $Restart = 20$ with $(pre, post)_{smooth} = (2, 2)$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	5	4.2	12	17	15.78	1.8425
6	4	6	4.3	12	17	15.86	7.4499
7	4	6	4.4	12	17	15.86	26.8078
8	4	6	4.5	12	17	15.72	111.5711
9	4	6	4.6	12	17	15.76	490.3956

Table 5.169: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-4$ and $Restart = 20$ with $(pre, post)_{smooth} = (2, 2)$ and the relative tolerance for Newton is Tol = $1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	3.5	15	20	18.42	1.9267
6	3	5	3.7	15	20	18.51	7.1637
7	3	5	4.0	15	20	18.57	27.0173
8	3	5	3.9	15	20	18.41	108.3852
9	3	5	4.0	15	20	18.42	482.7263

Summary

When we compare the Tables 5.150, 5.151, 5.152 and 5.153 of our preconditioner P_{3b}, we observe that when we make Tol smaller for the numerical solution of this model we found that the average number of GMRES iterations goes up but the average number of Newton iterations goes down. Likewise, when we compare the Tables 5.160, 5.161, 5.162 and 5.163 of our preconditioner P_{4b}, we found the same results, which is when we increase the average number of GMRES iterations the average number of Newton iterations decrease. This is to be expected and shown that there is an optimal accuracy with which to solve the linear system.

We can draw the conclusion based upon these numerical results with the steady-state case we found that the better choice of GMRES tolerance is to take Tol = $1e-3$, whereas, in the time-dependent case the choice of GMRES tolerance is to take Tol = $1e-4$. Note that when we increase the accuracy by taking the GMRES tolerance below Tol = $1e-4$, the number of Newton iterations does not decrease any more, therefore, this choice is the best choice.

In the time-dependent case, we observed that the optimal values are slightly different when we take a small time step. In this case, the solution at the previous step gives a good initial guess and hence we take only a few Newton iterations. However, when we do a large time step, as we can see in Tables 5.128 and 5.129, the performance is closer to the steady-state case. One of the challenges of solving the nonlinear problem efficiently is finding the best choice of the parameters which can depend on the problem itself.

From all the former numerical results for the P_{3b} and P_{4b} preconditioners we found that the best performance of the Newton-Krylov-AMG algorithm is given by Table 5.152 and Table 5.162 from P_{3b} and P_{4b} preconditioners respectively. These results are almost identical, where GMRES tolerance is Tol = $1e-04$ and $(pre, post)_{smooth} = (1, 1)$.

To conclude, from all these numerical results, we can observe that the best new preconditioners for solving the thin film flow in the time-dependent case system are the P_{3b} and P_{4b} preconditioners which significantly reduce the number of Newton and GMRES iterations as well as the execution time. Furthermore, their execution times scale almost linearly with the size of the problem.

In the following subsection, we discuss and compare the numerical results of the numerical solution of the thin film flow system by applying the three separate nonlinear multilevel algorithms in order to obtain the best numerical solution for the time-dependent problem in two dimensions.

5.10.4 Discussion and Comparison

In this Chapter, we are particularly interested in the numerical solution of the thin film flow model using the three different nonlinear multilevel algorithms for the discrete systems of equations representing the steady-state and time-dependent problems. In this particular section, the numerical experiments for the time-dependent solution of the thin film flow system have been used to demonstrate close to an optimal solution using each approach.

We have used the analytic Jacobian for all three nonlinear multilevel algorithms to develop their efficiency. Although, the best parameters in each case were achieved by extensive numerical tests; we observed that these algorithms are sensitive to the selection of free parameters and that the best choice of parameters for one algorithm is no longer the best choice for the other two algorithms. As shown in Table 5.124, for the FAS algorithm we found that the best value of the parameter ω is $\omega = 1$ with the best smoother is Red Black G-S, with the best value of the pre- and post-smooth $(pre, post) = (1, 1)$ and the best coarse grid size is $G = 5$. For clarity, we present here the best numerical result of the FAS algorithm again as Table 5.170.

Table 5.170: FAS performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol $= 1e - 8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
6	5	1	1	7	8	7.6	11.4158
7	5	1	1	8	9	8.8	43.3668
8	5	1	1	9	10	9.4	180.2312
9	5	1	1	9	10	9.6	746.0546

We observed that the execution time is grown by approximately a factor of 4 as the problem size is grown by a factor of 4, this indicates that our FAS algorithm converges with a linear time complexity of $O(N)$.

For the Newton-MG algorithm we determined that the best value of the parameter ω is $\omega = 1$ with the best smoother which is Red Black G-S smoother, with the best value of the parameter pre- and post-smooth is $(pre, post) = (1, 1)$ and the best coarse grid size is $G = 6$, as shown in Table 5.143. We displayed the best numerical result of the Newton-MG algorithm again here in Table 5.171, for simplicity, as follows,

Table 5.171: Newton-MG performance for varying grid size for thin film flow in time-dependent case with $\triangle t = 0.1$ and No. of time steps = 10, Red Black G-S with $\omega = 1$, fixed V-cycle, the number of $V - cycle = 3$ and the relative tolerance for Newton is Tol $= 1e - 8$.

Input					Output			
Grid Size		Smoother		MG Solver	Newton Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	No. MG V-Cycle	*Min*	*Max*	*Average*	-
7	6	1	1	3	3	5	3.8	39.8166
8	6	1	1	3	4	5	4.2	182.4501
9	6	1	1	3	4	6	4.4	793.0276

We noted that the execution time is increased by just a little more than a factor of 4 from one run to the next; as the problem size is increased by a factor of 4. This implies that our Newton-MG algorithm converges with close to linear complexity of $O(N)$.

For our newly proposed preconditioners applied with Newton-Krylov, we observed that the best results are presented in Table 5.152 for the P_{3b} preconditioner and in Table 5.162 for the P_{4b} preconditioner with GMRES $Tol = 1e - 4$. We presented here these numerical results again, for simplicity, as Table 5.172 and Table 5.173.

Table 5.172: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{3b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-4$ and $Restart = 20$ with $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	3.5	16	27	18.80	1.7292
6	3	5	3.7	16	27	19.00	6.7337
7	3	5	3.9	16	27	18.92	24.8841
8	4	5	4.1	16	26	18.95	107.3758
9	4	5	4.2	16	26	18.80	480.2390

Table 5.173: Newton-Krylov-AMG with AMG preconditioned solver in the time-dependent case, using P_{4b} preconditioner with $\triangle t = 0.1$ and the number of time step = 10, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-4$ and $Restart = 20$ with $(pre, post)_{smooth} = (1, 1)$ and the relative tolerance for Newton is Tol $= 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time (sec)
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	5	3.7	15	21	18.72	1.8223
6	3	5	3.9	15	20	18.79	7.0630
7	4	5	4.1	16	20	18.82	25.9074
8	4	5	4.2	16	20	18.82	109.0045
9	4	5	4.2	16	20	18.73	476.1476

As we can see from these tables, they have close to an optimally of the computational time and the required number of GMRES iterations is constant as the grid is refined. While all three nonlinear multilevel algorithms yield close to an optimal solution time for the thin film flow in the time-dependent problem, the best performance is achieved by the Newton-Krylov-AMG with our new preconditioner. The most obvious conclusion to arise from this examination is that the Newton-Krylov-AMG with our new preconditioner is the best and quickest nonlinear multilevel algorithm (of these we have considered) for solving the thin film flow in the time-dependent problem in 2D.

To sum up, we have achieved close to an optimal numerical solution by using FAS and Newton-MG algorithms, furthermore, FAS is superior to Newton-MG for the thin film flow model in the time-dependent problem. However, our preconditioned Newton-Krylov-AMG is superior to both. Hence, the numerical results that we have shown in this chapter are confirmed that

Newton-Krylov-AMG with our new preconditioner is the best approach to solve the thin film flow system in the time-dependent problem.

5.11 Summary

A number of approaches have been considered for both the steady-state and time-dependent problems to solve the thin film flow system. In this chapter, we demonstrated that the three nonlinear multilevel algorithms can provide grid independent convergence and close to linear time complexity. As far as we know this is the first time comparison for the thin film flow system has been made using the three nonlinear multilevel algorithms. Comparison of the numerical results illustrates the superior computational efficiency the implementation of Newton-Krylov-AMG over FAS and Newton-MG algorithms for the thin film flow system in both the steady-state and the time-dependent problems. A further contribution of this research to the current literature lies in the fact that the optimal behaviour of the Newton-Krylov-AMG algorithm requires the design of a new block preconditioner for the Jacobian system.

We have found the numerical solutions of thin film flow model using the nonlinear multilevel methods outlined in this thesis for the steady-state and the time-dependent problems. The main components are nonlinear multilevel approaches in conjunction with a linearisation strategy. Furthermore, it is shown that the nonlinear fourth-order system can be solved almost optimally using FAS, Newton-MG, and Newton-Krylov with our new preconditioning Newton-Krylov-AMG solvers. There are three nonlinear methods and we have tried to do a fair comparison between them: we did this by putting our best effort into optimizing each method. We have compared them with the best choice of the parameters.

As a final observation in this chapter, we note that the Newton-MG approach is consistently slower than the other two. Consequently, in the following chapter, when we consider a different time-dependent nonlinear system, we focus on the FAS and Newton-Krylov approaches.

Chapter 6

The Cahn-Hilliard-Hele-Shaw System

6.1 Introduction

In this chapter, we present the Cahn-Hilliard-Hele-Shaw (CHHS) system of equations previously described in Chapter 2, Section 2.3.2. This system of equations arises in various models such as spinodal decomposition of a binary fluid in a Hele-Shaw cell, cell sorting, and tumour growth as well as in two-phase flows in porous media. Recall that the Cahn-Hilliard-Hele-Shaw system of equations is given as follows:

$$\frac{\partial \phi}{\partial t} = \nabla.((1+\gamma\phi^2)\nabla\mu) + \nabla.(\phi\nabla p), \tag{6.1}$$

$$\mu = \phi^3 - \phi - \varepsilon^2\,\Delta\phi, \tag{6.2}$$

$$-\Delta p = \gamma\,\nabla.(\phi\nabla\mu). \tag{6.3}$$

In this chapter, we will solve this model on the domain $[-L_x, L_x] \times [-L_y, L_y]$ where $L_x = L_y = 3.2$ in 2D and $t > 0$. This system is specified with Dirichlet boundary conditions throughout, which are:

$$\begin{aligned} \phi &= -1 \\ \mu &= 0 \\ p &= 0. \end{aligned} \tag{6.4}$$

To demonstrate this solver, we consider the initial conditions $\phi(x, y, t = 0)$, $\mu(x, y, t = 0)$ and $p(x, y, t = 0)$. For Equation (6.1), we require the initial conditions $\phi(x, y, t = 0)$ and define

these as follows:

$$\phi = \tanh\left(10 * \left(\frac{1}{2} - R\right)\right), \tag{6.5}$$

where

$$R = (x^2 + y^2)^{\frac{1}{2}}.$$

Once we have specified ϕ we can obtain μ by evaluating Equation (6.2) to compute $\mu(x, y, t = 0)$. We can then solve the Laplace Equation (6.3) to compute $p(x, y, t = 0)$. To uniquely solve Equation (6.3) boundary conditions for p are required; we choose $p = 0$ on the boundary, which is compatible with (6.4). In the remainder of this chapter, we present a discussion and comparison of two different nonlinear multilevel algorithms in order to solve the discretised CHHS system of equations efficiently.

6.2 The Fully Discrete System for the Cahn-Hilliard-Hele-Shaw Model

In this section, we introduce the fully discrete system for this problem, where the time derivative is discretised using BDF1 scheme and the five points FDM is used in space. We require a temporal discretisation method in Equation (6.1) only. We applied the BDF1 method with a fixed $\triangle t$ to replace the derivative $\frac{\partial \phi}{\partial t}$. The discrete CHHS system is given by the approximation of Equation (6.1) as follows,

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^{n}}{\triangle t} =$$

$$\left(\frac{1}{\triangle x^2}\right)\left[(1+\gamma\,(\frac{\phi_{i+1,j}^{n+1}+\phi_{i,j}^{n+1}}{2})^2)(\mu_{i+1,j}^{n+1}-\mu_{i,j}^{n+1}) - (1+\gamma\,(\frac{\phi_{i,j}^{n+1}+\phi_{i-1,j}^{n+1}}{2})^2)(\mu_{i,j}^{n+1}-\mu_{i-1,j}^{n+1})\right]$$

$$+\left(\frac{1}{\triangle y^2}\right)\left[(1+\gamma\,(\frac{\phi_{i,j+1}^{n+1}+\phi_{i,j}^{n+1}}{2})^2)(\mu_{i,j+1}^{n+1}-\mu_{i,j}^{n+1}) - (1+\gamma\,(\frac{\phi_{i,j}^{n+1}+\phi_{i,j-1}^{n+1}}{2})^2)(\mu_{i,j}^{n+1}-\mu_{i,j-1}^{n+1})\right]$$

$$+\left(\frac{1}{\triangle x^2}\right)\left[(\frac{\phi_{i+1,j}^{n+1}+\phi_{i,j}^{n+1}}{2})(p_{i+1,j}^{n+1}-p_{i,j}^{n+1}) - (\frac{\phi_{i,j}^{n+1}+\phi_{i-1,j}^{n+1}}{2})(p_{i,j}^{n+1}-p_{i-1,j}^{n+1})\right]$$

$$+\left(\frac{1}{\triangle y^2}\right)\left[(\frac{\phi_{i,j+1}^{n+1}+\phi_{i,j}^{n+1}}{2})(p_{i,j+1}^{n+1}-p_{i,j}^{n+1}) - (\frac{\phi_{i,j}^{n+1}+\phi_{i,j-1}^{n+1}}{2})(p_{i,j}^{n+1}-p_{i,j-1}^{n+1})\right]. \tag{6.6}$$

Furthermore, the discrete form of Equation (6.2) is

$$\mu_{i,j}^{n+1} = (\phi_{i,j}^{n+1})^3 - \phi_{i,j}^{n+1}$$

$$-\epsilon^2\left[\frac{1}{\triangle x^2}(\phi_{i+1,j}^{n+1} - 2\phi_{i,j}^{n+1} + \phi_{i-1,j}^{n+1}) + \frac{1}{\triangle y^2}(\phi_{i,j+j}^{n+1} - 2\phi_{i,j}^{n+1} + \phi_{i,j-1}^{n+1})\right]. \tag{6.7}$$

and Equation (6.3) becomes

$$-\frac{1}{\triangle x^2}(p_{i+1,j}^{n+1}-2p_{i,j}^{n+1}+p_{i-1,j}^{n+1})-\frac{1}{\triangle y^2}(p_{i,j+1}^{n+1}-2p_{i,j}^{n+1}+p_{i,j-1}^{n+1})$$

$$=\frac{\gamma}{\triangle x^2}\left[\left(\frac{\phi_{i+1,j}^{n+1}+\phi_{i,j}^{n+1}}{2}\right)(\mu_{i+1,j}^{n+1}-\mu_{i,j}^{n+1})-\left(\frac{\phi_{i,j}^{n+1}+\phi_{i-1,j}^{n+1}}{2}\right)(\mu_{i,j}^{n+1}-\mu_{i-1,j}^{n+1})\right]$$

$$+\frac{\gamma}{\triangle y^2}\left[\left(\frac{\phi_{i,j+1}^{n+1}+\phi_{i,j}^{n+1}}{2}\right)(\mu_{i,j+1}^{n+1}-\mu_{i,j}^{n+1})-\left(\frac{\phi_{i,j}^{n+1}+\phi_{i,j-1}^{n+1}}{2}\right)(\mu_{i,j}^{n+1}-\mu_{i,j-1}^{n+1})\right]. \tag{6.8}$$

This is a nonlinear system involving the unknown values of ϕ, μ and p at each grid point (i,j) on the new time level $(n+1)$. Having shown the discrete CHHS system, in the two different following section we will implement the two different multilevel solvers that we are investigating in this chapter. Before doing so, however, we consider the Jacobian of this system.

6.2.1 The Jacobian Matrix

The FDM spatial discretisation with the temporal discretisation scheme BDF1 is presented in Equations (6.6), (6.7) and (6.8). Our aim here is to examine the analytical Jacobian of this discrete nonlinear CHHS system. This full analytical Jacobian matrix is a sparse matrix; since we are applying a discretisation scheme that is based upon local approximation, i.e. the FDM. Consequently, we offer here the expression of this sparse matrix that will be needed for the two nonlinear solution algorithms that we will apply in this chapter.

Let J_1 be the analytical Jacobian matrix for the CHHS system which is arranged in the following nine blocks:

$$J_1=\begin{bmatrix}J_{11}&J_{12}&J_{13}\\J_{21}&J_{22}&J_{23}\\J_{31}&J_{32}&J_{33}\end{bmatrix}, \tag{6.9}$$

or

$$J_1=\begin{bmatrix}\frac{\partial F_\phi}{\partial\phi}&\frac{\partial F_\phi}{\partial\mu}&\frac{\partial F_\phi}{\partial p}\\\frac{\partial F_\mu}{\partial\phi}&\frac{\partial F_\mu}{\partial\mu}&\frac{\partial F_\mu}{\partial p}\\\frac{\partial F_p}{\partial\phi}&\frac{\partial F_p}{\partial\mu}&\frac{\partial F_p}{\partial p}\end{bmatrix}, \tag{6.10}$$

where

$$[F_\phi(U)]_{i,j}=-\frac{\phi_{i,j}^{n+1}-\phi_{i,j}^{n}}{\triangle t}$$

$$+\left(\frac{1}{\triangle x^2}\right)\left[(1+\gamma\,(\frac{\phi_{i+1,j}^{n+1}+\phi_{i,j}^{n+1}}{2})^2)(\mu_{i+1,j}^{n+1}-\mu_{i,j}^{n+1})-(1+\gamma\,(\frac{\phi_{i,j}^{n+1}+\phi_{i-1,j}^{n+1}}{2})^2)(\mu_{i,j}^{n+1}-\mu_{i-1,j}^{n+1})\right]$$

$$+\left(\frac{1}{\triangle y^2}\right)\left[(1+\gamma\,(\frac{\phi_{i,j+1}^{n+1}+\phi_{i,j}^{n+1}}{2})^2)(\mu_{i,j+1}^{n+1}-\mu_{i,j}^{n+1})-(1+\gamma\,(\frac{\phi_{i,j}^{n+1}+\phi_{i,j-1}^{n+1}}{2})^2)(\mu_{i,j}^{n+1}-\mu_{i,j-1}^{n+1})\right]$$

$$+\left(\frac{1}{\triangle x^2}\right)\left[(\frac{\phi_{i+1,j}^{n+1}+\phi_{i,j}^{n+1}}{2})(p_{i+1,j}^{n+1}-p_{i,j}^{n+1})-(\frac{\phi_{i,j}^{n+1}+\phi_{i-1,j}^{n+1}}{2})(p_{i,j}^{n+1}-p_{i-1,j}^{n+1})\right]$$

$$+\left(\frac{1}{\triangle y^2}\right)\left[(\frac{\phi_{i,j+1}^{n+1}+\phi_{i,j}^{n+1}}{2})(p_{i,j+1}^{n+1}-p_{i,j}^{n+1})-(\frac{\phi_{i,j}^{n+1}+\phi_{i,j-1}^{n+1}}{2})(p_{i,j}^{n+1}-p_{i,j-1}^{n+1})\right], \tag{6.11}$$

$$[F_\mu(U)]_{i,j}=-\mu_{i,j}^{n+1}+(\phi_{i,j}^{n+1})^3-\phi_{i,j}^{n+1}$$

$$-\epsilon^2\left[\frac{1}{\triangle x^2}(\phi_{i+1,j}^{n+1}-2\phi_{i,j}^{n+1}+\phi_{i-1,j}^{n+1})+\frac{1}{\triangle y^2}(\phi_{i,j+j}^{n+1}-2\phi_{i,j}^{n+1}+\phi_{i,j-1}^{n+1})\right], \tag{6.12}$$

and

$$[F_p(U)]_{i,j}=\frac{1}{\triangle x^2}(p_{i+1,j}^{n+1}-2p_{i,j}^{n+1}+p_{i-1,j}^{n+1})+\frac{1}{\triangle y^2}(p_{i,j+1}^{n+1}-2p_{i,j}^{n+1}+p_{i,j-1}^{n+1})$$

$$+\frac{\gamma}{\triangle x^2}\left[\left(\frac{\phi_{i+1,j}^{n+1}+\phi_{i,j}^{n+1}}{2}\right)(\mu_{i+1,j}^{n+1}-\mu_{i,j}^{n+1})-\left(\frac{\phi_{i,j}^{n+1}+\phi_{i-1,j}^{n+1}}{2}\right)(\mu_{i,j}^{n+1}-\mu_{i-1,j}^{n+1})\right]$$

$$+\frac{\gamma}{\triangle y^2}\left[\left(\frac{\phi_{i,j+1}^{n+1}+\phi_{i,j}^{n+1}}{2}\right)(\mu_{i,j+1}^{n+1}-\mu_{i,j}^{n+1})-\left(\frac{\phi_{i,j}^{n+1}+\phi_{i,j-1}^{n+1}}{2}\right)(\mu_{i,j}^{n+1}-\mu_{i,j-1}^{n+1})\right]. \tag{6.13}$$

Furthermore, we define

$$F=\begin{pmatrix}F_\phi\\F_\mu\\F_p\end{pmatrix}, \tag{6.14}$$

U to be the vector of unknowns as follows:

$$U=\begin{pmatrix}\phi^{n+1}\\\mu^{n+1}\\p^{n+1}\end{pmatrix}, \tag{6.15}$$

and ϕ^{n+1}, μ^{n+1} and p^{n+1} to be the vectors of unknown values on grid points at time level $(n+1)$. We produce in this subsection the entries J_{11}, J_{12}, J_{13}, J_{21}, J_{22}, J_{23}, J_{31}, J_{32}, and J_{33} of the analytical matrix (6.9) from Equations (6.6), (6.7) and (6.8). The terms that we show here are for typical interior points of the domain, however, they may likewise be modified according to the Dirichlet boundary conditions (see Appendix B). We consider only here the non-zero entries of each block, which can be applied to produce the analytical Jacobian efficiently in a sparse format.

In the following presentation, we drop superscript $(n+1)$ to make the notation simpler. We differentiate Equation (6.11) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the

non-zero entries of J_{11} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i+1,j}} = \frac{1}{\Delta x^2}\left[\gamma\ (\phi_{i+1,j}+\phi_{i,j})\ (\mu_{i+1,j}-\mu_{i,j}) + \left(\frac{p_{i+1,j}-p_{i,j}}{2}\right)\right], \tag{6.16}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i-1,j}} = -\frac{1}{\Delta x^2}\left[\gamma\ (\phi_{i,j}+\phi_{i-1,j})\ (\mu_{i,j}-\mu_{i-1,j}) + \left(\frac{p_{i,j}-p_{i-1,j}}{2}\right)\right], \tag{6.17}$$

$$\begin{aligned}\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j}} = &-\frac{1}{\Delta t} + \frac{1}{\Delta x^2}\left[\gamma\ (\phi_{i+1,j}+\phi_{i,j})\ (\mu_{i+1,j}-\mu_{i,j}) + \left(\frac{p_{i+1,j}-p_{i,j}}{2}\right)\right]\\ &-\frac{1}{\Delta x^2}\left[\gamma\ (\phi_{i,j}+\phi_{i-1,j})\ (\mu_{i,j}-\mu_{i-1,j}) + \left(\frac{p_{i,j}-p_{i-1,j}}{2}\right)\right]\\ &+\frac{1}{\Delta y^2}\left[\gamma\ (\phi_{i,j+1}+\phi_{i,j})\ (\mu_{i,j+1}-\mu_{i,j}) + \left(\frac{p_{i,j+1}-p_{i,j}}{2}\right)\right]\\ &-\frac{1}{\Delta y^2}\left[\gamma\ (\phi_{i,j}+\phi_{i,j-1})\ (\mu_{i,j}-\mu_{i,j-1}) + \left(\frac{p_{i,j}-p_{i,j-1}}{2}\right)\right],\end{aligned} \tag{6.18}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j+1}} = \frac{1}{\Delta y^2}\left[\gamma\ (\phi_{i,j+1}+\phi_{i,j})\ (\mu_{i,j+1}-\mu_{i,j}) + \left(\frac{p_{i,j+1}-p_{i,j}}{2}\right)\right], \tag{6.19}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j-1}} = -\frac{1}{\Delta y^2}\left[\gamma\,(\phi_{i,j}+\phi_{i,j-1})\,(\mu_{i,j}-\mu_{i,j-1}) + (\frac{p_{i,j}-p_{i,j-1}}{2})\right], \tag{6.20}$$

where $i = 2, \ldots, N$ and $j = 2, \ldots, M$.

We differentiate Equation (6.11) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{12} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{1}{\Delta x^2})\left[(1+\gamma\,((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2))\right], \tag{6.21}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{1}{\Delta x^2})\left[(1+\gamma\,((\frac{\phi_{i,j}+\phi_{i-1,j}}{2})^2))\right], \tag{6.22}$$

$$\begin{aligned}\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j}} = &-(\frac{1}{\Delta x^2})\left[(1+\gamma\,((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i-1,j}}{2})^2))\right]\\ &-(\frac{1}{\Delta y^2})\left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i,j-1}}{2})^2))\right],\end{aligned} \tag{6.23}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{1}{\Delta y^2})\left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2))\right], \tag{6.24}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{1}{\Delta y^2})\left[(1+\gamma\,((\frac{\phi_{i,j}+\phi_{i,j-1}}{2})^2))\right], \tag{6.25}$$

where $i = 2, ..., N.$, $j = 2, ..., M.$

We differentiate Equation (6.11) with respect to $p_{i+1,j}$, $p_{i-1,j}$, $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{13} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} \right], \tag{6.26}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right], \tag{6.27}$$

$$\begin{aligned} \frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j}} = &-(\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right] \\ &-(\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \end{aligned} \tag{6.28}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} \right], \tag{6.29}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{6.30}$$

where $i = 2, \ldots, N$ and $j = 2, \ldots, M$.

We differentiate Equation (6.12) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{21} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i+1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{6.31}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{6.32}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j}} = -1 + 3\,(\phi_{i,j})^2 + 2\,(\epsilon)^2 \left[(\frac{1}{\Delta x^2}) + (\frac{1}{\Delta y^2}) \right], \tag{6.33}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j+1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{6.34}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{6.35}$$

where $i = 2, \ldots, N$, $j = 2, \ldots, M$ and $\sigma = -1 + 3\,(\phi_{i,j})^2$.

We differentiate Equation (6.12) with respect to $\mu_{i,j}$ to obtain the non-zero entries of J_{22} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \mu_{i,j}} = -1, \tag{6.36}$$

where $i = 2, \ldots, N$ and $j = 2, \ldots, M$.

Since Equation (6.12) does not feature the pressure, when we differentiate with respect to $p_{i+1,j}$, $p_{i-1,j}$, $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j-1}$ all entries of J_{23} are zero.

We differentiate Equation (6.13) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{31} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i+1,j}} = (\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i+1,j} - \mu_{i,j}), \tag{6.37}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i,j} - \mu_{i-1,j}), \tag{6.38}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j}} &= (\frac{\gamma}{\Delta x^2})(\frac{1}{2})\,[(\mu_{i+1,j} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i-1,j})] \\ &+(\frac{\gamma}{\Delta y^2})(\frac{1}{2})\,[(\mu_{i,j+1} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i,j-1})]\,,\end{aligned} \tag{6.39}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j+1}} = (\frac{\gamma}{\Delta y^2})\,(\frac{1}{2})\,(\mu_{i,j+1} - \mu_{i,j}), \tag{6.40}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{\gamma}{\Delta y^2})\,(\frac{1}{2})\,(\mu_{i,j} - \mu_{i,j-1}), \tag{6.41}$$

where $i = 2, \ldots, N$ and $j = 2, \ldots, M$.

We differentiate Equation (6.13) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{32} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{\gamma}{\Delta x^2})\,\frac{(\phi_{i+1,j} + \phi_{i,j})}{2}, \tag{6.42}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{\gamma}{\Delta x^2})\,\frac{(\phi_{i,j} + \phi_{i-1,j})}{2}, \tag{6.43}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j}} &= -(\frac{\gamma}{\Delta x^2})\,\left[\frac{(\phi_{i+1,j} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i-1,j})}{2}\right] \\ &-(\frac{\gamma}{\Delta y^2})\,\left[\frac{(\phi_{i,j+1} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i,j-1})}{2}\right],\end{aligned} \tag{6.44}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{\gamma}{\Delta y^2})\,\frac{(\phi_{i,j+1} + \phi_{i,j})}{2}, \tag{6.45}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{\gamma}{\Delta y^2})\,\frac{(\phi_{i,j} + \phi_{i,j-1})}{2}, \tag{6.46}$$

where $i = 2, \ldots, N$ and $j = 2, \ldots, M$.

We differentiate Equation (6.13) with respect to $p_{i+1,j}$, $p_{i-1,j}$, $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{33} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}), \tag{6.47}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}), \tag{6.48}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j}} = -(2)\,(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}), \tag{6.49}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}), \tag{6.50}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}), \tag{6.51}$$

where $i = 2, \ldots, N$, $j = 2, \ldots, M$. Now let us rewrite the analytical Jacobian matrix J_1 in Equation (6.10) of the CHHS system as follows:

$$J_1 = \begin{pmatrix} \frac{-1}{\triangle t} I + B_1 & K_1 & K_2 \\ \sigma + K_\epsilon & -I & 0 \\ B_2 & K_3 & K \end{pmatrix}, \tag{6.52}$$

where $\frac{-1}{\triangle t} I + B_1 = \frac{\partial F_\phi}{\partial \phi}$, $K_1 = \frac{\partial F_\phi}{\partial \mu}$, $K_2 = \frac{\partial F_\phi}{\partial p}$, $\sigma + K_\epsilon = \frac{\partial F_\mu}{\partial \phi}$, $-I = \frac{\partial F_\mu}{\partial \mu}$, $B_2 = \frac{\partial F_p}{\partial \phi}$, $K_3 = \frac{\partial F_p}{\partial \mu}$, and $K = \frac{\partial F_p}{\partial p}$ are block matrices.

We can change the row (and/or column) ordering of the sparse analytical Jacobian matrix J_1 in Equation (6.10) as an alternative order for the sparse Jacobian. Let J_2 be the analytical Jacobian matrix for the alternative ordering which is presented as in the following nine blocks:

$$J_2 = \begin{pmatrix} \sigma + K_\epsilon & -I & 0 \\ \frac{-1}{\triangle t} I + B_1 & K_1 & K_2 \\ B_2 & K_3 & K \end{pmatrix}, \tag{6.53}$$

where

$$F = \begin{pmatrix} F_\mu \\ F_\phi \\ F_p \end{pmatrix}, \tag{6.54}$$

and the ordering of the unknowns is still given by

$$U = \begin{pmatrix} \phi \\ \mu \\ p \end{pmatrix}. \tag{6.55}$$

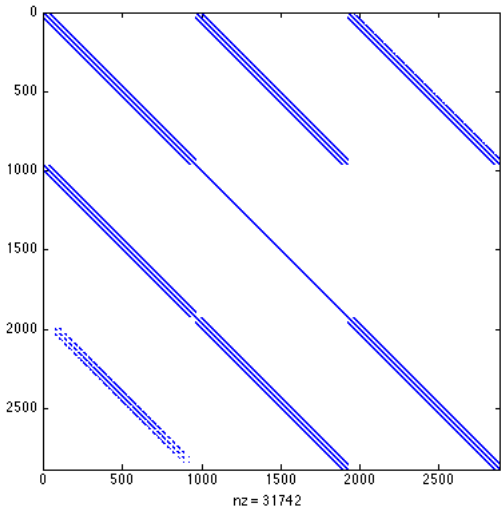


Figure 6.1: The sparse Jacobian matrix J_1 for the Cahn-Hilliard-Hele-Shaw model in grid size 65^2 and the number of unknowns $N = 2883$.

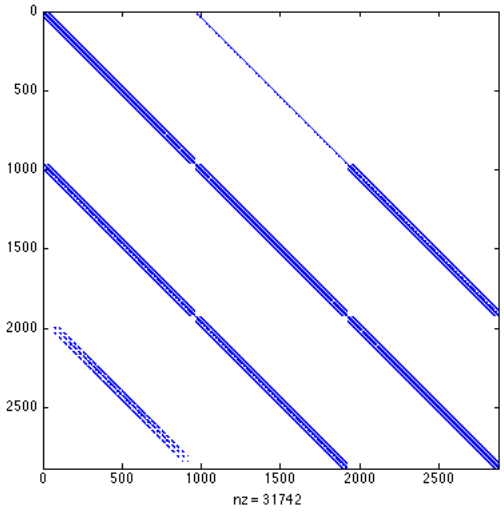


Figure 6.2: The sparse Jacobian matrix J_2 for the Cahn-Hilliard-Hele-Shaw model in grid size 65^2 and the number of unknowns $N = 2883$.

The sparsity pattern of the J_1 and J_2 matrices is the distribution of the non-zero entries in these matrices. Shown in Figure 6.1 and Figure 6.2 are the sparse analytical Jacobian matrices J_1 and J_2 for the CHHS system in their different orders; both have block sparse structure.

In the following subsection, we describe the use of the Jacobian matrix as part of two different multilevel solution methods for the solution of the nonlinear system of equations (6.6), (6.7) and (6.8).

6.3 Two Different Solvers

In this section, we consider two nonlinear multilevel algorithms: the FAS and the Newton-Krylov-AMG algorithm. Recall that in Chapter 5 we compared FAS against two Newton methods, Newton-MG and Newton-Krylov-AMG, and found that the Newton-Krylov-AMG algorithm consistently and significantly outperforms Newton-MG provided we find a good preconditioner. Therefore, we focus only on FAS and Newton-Krylov-AMG algorithms (and generating good AMG-based preconditioners) in this Chapter.

We consider here the fully-discrete system that arises from the discrete CHHS system at each time step and the steps required to apply the FAS and Newton-Krylov-AMG algorithms in Subsection 6.3.1 and Subsection 6.3.2. We present their results in Subsections 6.5.1 and 6.5.2 respectively.

6.3.1 FAS

In this subsection, we consider the FAS algorithm in order to solve the discrete CHHS system. We displayed the FAS scheme in detail in Algorithm 2 in Chapter 4. This algorithm requires a nonlinear smoother and uses a coarse grid correction scheme on a hierarchy of grids. We have chosen the simplest transfer operators in the FAS algorithm, where we are applying a linear interpolation operator and a full-weighting restriction operator in the same way as we have done with the first model in Chapter 5.

The nonlinear discrete algebraic system of equations of the CHHS model can be written as follows:

$$F_j(U) = 0,\ j = 1, \ldots, neq, \tag{6.56}$$

where $neq = 3 \times nu$ is the number of equations with nu the number of unknown grid points in the Cahn-Hilliard-Hele-Shaw system of equations. In other words, neq is the total number of unknowns in (6.15).
We can specify the nonlinear Gauss-Seidel iteration, for unknown u_j^{k+1}, as solving for u_j^{k+1} in

$$F_j(u_1^{k+1}, ..., u_{j-1}^{k+1}, u_j^{k+1}, u_{j+1}^{k}, ..., u_{neq}^{k}) = 0, \tag{6.57}$$

where k and $k+1$ denote the current and new approximations. Similarly to Chapter 5 we use the Red-Black G-S smoother extended to a point-wise (3×3) block update as follows. We

produce a smoother based on a simultaneous update of ϕ, μ and p at each point of the grid. In order to use this smoother, we define J_{bj} which is the (3×3) block-diagonal system of our analytical Jacobian matrix at point j of the grid. We can define these values analytically from the expressions for the diagonal entries in Equations (6.16)-(6.51). We can solve such a system as follows:

$$\underbrace{\begin{pmatrix} \frac{\partial F_{\phi_j}}{\partial \phi_j} & \frac{\partial F_{\phi_j}}{\partial \mu_j} & \frac{\partial F_{\phi_j}}{\partial p_j} \\ \frac{\partial F_{\mu_j}}{\partial \phi_j} & \frac{\partial F_{\mu_j}}{\partial \mu_j} & \frac{\partial F_{\mu_j}}{\partial p_j} \\ \frac{\partial F_{p_j}}{\partial \phi_j} & \frac{\partial F_{p_j}}{\partial \mu_j} & \frac{\partial F_{pj}}{\partial p_j} \end{pmatrix}}_{J_{bj}} \begin{pmatrix} \delta_{\phi_j} \\ \delta_{\mu_j} \\ \delta_{p_j} \end{pmatrix} = - \begin{pmatrix} F_{\phi_j} \\ F_{\mu_j} \\ F_{p_j} \end{pmatrix}, \tag{6.58}$$

for $j = 1, \ldots, nu$, where nu is the number of unknown grid points.

Having solved (6.58) we update the solution at node j to obtain:

$$\begin{pmatrix} \phi_j^{k+1} \\ \mu_j^{k+1} \\ p_j^{k+1} \end{pmatrix} = \begin{pmatrix} \phi_j^{k} \\ \mu_j^{k} \\ p_j^{k} \end{pmatrix} + \begin{pmatrix} \delta_{\phi_j} \\ \delta_{\mu_j} \\ \delta_{p_j} \end{pmatrix}. \tag{6.59}$$

In each step, we build and solve the j^{th} (3×3) system: the updated Equation (6.59) may be applied in a Gauss-Seidel manner by updating each $j = 1, ..., nu$ in turn.

We have tested the FAS algorithm with the nonlinear weighted-Red-Black Gauss-Seidel smoother with various parameters of ω in order to solve the CHHS model. Results are presented in subsection 6.5.1.

6.3.2 Newton-Krylov-AMG

This subsection concentrates on the nonlinear Newton-Krylov algorithm with a new AMG-based preconditioner. Krylov-subspace methods are generally used with a preconditioner that accelerates the convergence of the linear iterations as we mentioned before in Chapter 4. Consequently, we consider an implementation of this algorithm to solve the Cahn-Hilliard-Hele-Shaw model. In this subsection, we develop an algorithm for solving the nonlinear Cahn-Hilliard-Hele-Shaw system (almost) optimally with our new preconditioners.

The Cahn-Hilliard-Hele-Shaw discrete system is a non-symmetric system; therefore, to solve this system iteratively we choose a preconditioned GMRES scheme. Since Krylov methods converge slowly when used to this sort of a system, we will need a preconditioner to achieve fast and robust convergence rates. As we have mentioned earlier in Chapter 4 the AMG iterative methods are one of the most efficient for solving large sparse linear systems that are obtained from the discretisation of single elliptic partial differential equations. Our purpose here is to combine AMG methods as a component of a preconditioner for Krylov subspace methods for

our, more complex, linearised system.

We have developed and implemented this algorithm as discussed in Chapter 4 but generalized to reflect that we have now a more complicated system arising from three discretised equations. The Newton-Krylov algorithm as we described earlier has two different iterations: outer iterations and inner iterations. Newton's iterations which are the outer iterations and the inner iterations which are the linear Krylov methods. Moreover, we will apply a new preconditioner that includes AMG, for which we employ a software implementation that is available in the Harwell Subroutine Library (HSL) [70]. We will implement new optimal (or near optimal) preconditioners for the Newton-Krylov iterations as a part of the solution of the discrete nonlinear system.

We can write the linearised system Equation (6.58) in the following block form:

$$\begin{pmatrix} \frac{-1}{\triangle t} I + B_1 & K_1 & K_2 \\ \sigma + K_\epsilon & -I & 0 \\ B_2 & K_3 & K \end{pmatrix} \begin{pmatrix} \delta_\phi \\ \delta_\mu \\ \delta_p \end{pmatrix} = \begin{pmatrix} F_\phi \\ F_\mu \\ F_p \end{pmatrix}, \tag{6.60}$$

where each block was defined earlier by the analytical expression in Equations (6.16) to (6.51). We note that all blocks denoted K represent discrete elliptic operations. To apply AMG it is advantageous to have blocks on the diagonal that represent elliptic operators, hence, we make use of the reordering that gives J_2 as the Jacobian. For this Jacobian, we define a perfect preconditioner P such that

$$P\, z = r. \tag{6.61}$$

Let $z = (z_\phi \;\; z_\mu \;\; z_p)^\top$ and $r = (r_\mu \;\; r_\phi \;\; r_p)^\top$ be column-vectors so that Equation (6.61) has the following block form:

$$P\, z = \begin{pmatrix} \sigma + K_\epsilon & -I & 0 \\ \frac{-1}{\triangle t} I + B_1 & K_1 & K_2 \\ B_2 & K_3 & K \end{pmatrix} \begin{pmatrix} z_\phi \\ z_\mu \\ z_p \end{pmatrix} = \begin{pmatrix} r_\mu \\ r_\phi \\ r_p \end{pmatrix}. \tag{6.62}$$

Before we describe our practical preconditioners we need to implement the Gauss elimination procedure on the analytical Jacobian J_2, Equation (6.53) in similar processes to obtain an exact block reduction for a (3×3) system. To simplify our notation, we will rewrite the analytical Jacobian J_2 in Equation (6.53) in a compact way as follows,

$$\begin{pmatrix} A & -I & 0 \\ B & C & D \\ E & F & G \end{pmatrix}. \tag{6.63}$$

If we now apply block elimination we obtain the following sequence:

$$\begin{pmatrix} A & -I & 0 \\ 0 & H_1 & D \\ E & F & G \end{pmatrix}, \tag{6.64}$$

where $H_1 = C + A^{-1}(B)$,

$$\begin{pmatrix} A & -I & 0 \\ 0 & H_1 & D \\ 0 & H_2 & G \end{pmatrix}, \tag{6.65}$$

where $H_2 = F + A^{-1}(E)$, and

$$\begin{pmatrix} A & -I & 0 \\ 0 & H_1 & D \\ 0 & 0 & H_3 \end{pmatrix}, \tag{6.66}$$

where $H_3 = G - D\,H_1^{-1}(H_2)$.

We may write these simple expressions in terms of the original matrix entries, such as,

$$\begin{aligned} H_1 &= K_1 + (\sigma + K_\epsilon)^{-1}(\tfrac{-1}{\triangle t} I + B_1), \\ H_2 &= K_3 + (\sigma + K_\epsilon)^{-1}(B_2), \\ H_3 &= K - K_2[K_1 + (\sigma + K_\epsilon)^{-1}(\tfrac{-1}{\triangle t} I + B_1)]^{-1}[K_3 + (\sigma + K_\epsilon)^{-1}(B_2)]. \end{aligned}$$

The Schur Complement blocks H_1 and H_3 are expensive to compute but we note that their form may suggest suitable approximations, such as,

$$\begin{aligned} H_1 &= K_1 + (\sigma + K_\epsilon)^{-1}(\tfrac{-1}{\triangle t} I + B_1) \simeq K_1, \\ H_3 &= K - K_2[K_1 + (\sigma + K_\epsilon)^{-1}(\tfrac{-1}{\triangle t} I + B_1)]^{-1}[K_3 + (\sigma + K_\epsilon)^{-1}(B_2)] \simeq K. \end{aligned}$$

These practical approximations will allow us to apply AMG to each block on the diagonal but we must evaluate the effect of these approximations on the overall performance of the algorithm.

We will develop various preconditioners here as proposed, potentially efficient, preconditioners for the CHHS model. These preconditioners come from the approximation of an exact Schur Complement, which is the block reduction for (3×3) system in Equation (6.66).

The first two possible selections of a preconditioner for the CHHS model are P_5 and P_6 for J_1, based on the original ordering, Equation (6.52). The P_5 and P_6 preconditioners are the diagonal blocks and upper triangular block preconditioners respectively, where the Schur Complement,

are approximated as above:

$$P_5 = \begin{pmatrix} \frac{-1}{\triangle t} I + B_1 & 0 & 0 \\ 0 & -I & 0 \\ 0 & 0 & K \end{pmatrix}. \tag{6.67}$$

and

$$P_6 = \begin{pmatrix} \frac{-1}{\triangle t} I + B_1 & K_1 & K_2 \\ 0 & -I & 0 \\ 0 & 0 & K \end{pmatrix}. \tag{6.68}$$

Now we will consider the alternative ordering in the Jacobian, J_2, based on the ordering Equation (6.53). The P_7 and P_8 preconditioners are diagonal blocks and the upper triangular block preconditioners respectively. Again the Schur Complement, blocks are approximated to give:

$$P_7 = \begin{pmatrix} \sigma + K_\epsilon & 0 & 0 \\ 0 & K_1 & 0 \\ 0 & 0 & K \end{pmatrix}, \tag{6.69}$$

and

$$P_8 = \begin{pmatrix} \sigma + K_\epsilon & -I & 0 \\ 0 & K_1 & K_2 \\ 0 & 0 & K \end{pmatrix}. \tag{6.70}$$

Moreover, we also consider P_{8a} which is a different option for P_8 which replaces, two blocks in the diagonal with one V-cycle AMG iteration. As in Chapter 5, the notation AMG(X) means that applying one V-cycle of AMG to the matrix X, for example:

$$P_{8a} = \begin{pmatrix} \sigma + K_\epsilon & -I & 0 \\ 0 & AMG(K_1) & K_2 \\ 0 & 0 & AMG(K) \end{pmatrix}. \tag{6.71}$$

In the following preconditioner, we will apply one V-cycle of AGM to all three blocks in the diagonal of the matrix J_2 in Equation (6.53), which indicates the P_{8b} preconditioner:

$$P_{8b} = \begin{pmatrix} AMG(\sigma + K_\epsilon) & -I & 0 \\ 0 & AMG(K_1) & K_2 \\ 0 & 0 & AMG(K) \end{pmatrix}. \tag{6.72}$$

In the following preconditioner, we will approximate and replace the first block in the diagonal only by the identity matrix whereas the other blocks in the diagonal are still the same as in P_8:

$$P_9 = \begin{pmatrix} -I & -I & 0 \\ 0 & K_1 & K_2 \\ 0 & 0 & K \end{pmatrix}. \tag{6.73}$$

In the next preconditioner, we will approximate and replace the first block in the diagonal only by the second term of the first block which is K_ϵ, however, the other blocks in the diagonal are still the same as P_8 and P_9:

$$P_{10} = \begin{pmatrix} K_\epsilon & -I & 0 \\ 0 & K_1 & K_2 \\ 0 & 0 & K \end{pmatrix}. \tag{6.74}$$

In the following preconditioner, we replace two of the diagonal blocks of P_{10} with one V-cycle AMG iteration, which defines the P_{10a} preconditioner:

$$P_{10a} = \begin{pmatrix} K_\epsilon & -I & 0 \\ 0 & AMG(K_1) & K_2 \\ 0 & 0 & AMG(K) \end{pmatrix}. \tag{6.75}$$

Here P_{10a} is an upper triangular block preconditioner, we solved by using a direct solver (back-slash) for the first block in the diagonal and replacing the two other blocks in the diagonal of P_{10} with one V-cycle AMG iteration.

In the final preconditioner that we consider, we modify P_{10} by using one V-cycle AMG iteration for all three blocks on the diagonal:

$$P_{10b} = \begin{pmatrix} AMG(K_\epsilon) & -I & 0 \\ 0 & AMG(K_1) & K_2 \\ 0 & 0 & AMG(K) \end{pmatrix}. \tag{6.76}$$

Here P_{10b} is an upper triangular block preconditioner, we solved by using one V-cycle AMG iteration for all three blocks in the diagonal.

In this subsection, we have presented a large number of potential preconditioning strategies designed for use with the Newton-Krylov algorithm for solving the CHHS system in 2D. In Subsection 6.5.2, we will display some numerical results of the Newton-Krylov-AMG algorithm with these preconditioners for the discrete the CHHS model.

6.4 Behaviour of Eigenvalues

We investigate here the behaviour of the eigenvalues of our proposed preconditioned matrices. It might be reasonable to obtain insight into the properties of the proposed preconditioners from the eigenvalues of the preconditioned matrix JP^{-1} for each preconditioner. In particular, if the effect of the preconditioner P is to cluster the eigenvalues of JP^{-1} and to bound the eigenvalues away from zero, we may expect it to be effective in practice. As we mentioned before in Section 5.6, the eigenvalues do not provide any guarantee regarding the performance,

however, they often provide a good indicator [57].

Therefore, in this subsection, we will present selections of plots of the eigenvalues resulting from the use of our new preconditioner for the CHHS model. Figures from Figure 6.3 to Figure 6.10 display the eigenvalues of the original matrix J_1 and the preconditioned matrices $J_1P_5^{-1}$ and $J_1P_6^{-1}$ respectively.

Our goal here is to investigate and understand the quality of our preconditioners for the GMRES solution. Due to the eigenvalues of the matrix J_1 having a broad spread as shown in Figures 6.3 and 6.5, the GMRES scheme has slow convergence [136]. We can see clearly from Figure 6.3 until Figure 6.10 the effect of the grid size and the impact of the preconditioners P_5 and P_6 for the matrix J_1 in Equation (6.52). The eigenvalues of the diagonal preconditioned matrix P_5 and the upper triangular preconditioned matrix P_6 are clustered in a small range around one and bounded away from the origin and infinity, and appear independent of the grid size. We show in Table 6.1 and Table 6.2 the minimum and the maximum numbers of the eigenvalues of the coefficient matrix J_1 and the preconditioned matrices $J_1P_5^{-1}$ and $J_1P_6^{-1}$ respectively.

As previously stated Figure 6.3 to Figure 6.10 provide a representative depiction of the eigenvalues of our preconditioner for the preconditioners P_5 and P_6 for the original matrix J_1. Similarly, for the re-ordered Jacobian J_2, Figure 6.11 to Figure 6.18 provide a representative depiction of the eigenvalues when applying preconditioners P_7, Equation (6.69), and P_8 Equation (6.70), respectively. The plots on the left side of these figures show eigenvalues for the original matrices and those on the right of these figures show eigenvalues with our preconditioners. These figures indicate that the preconditioned systems have spectra which are clustered around one and bounded away from the origin. Consequently, the application of P_7 and P_8 also results in systems that are good in terms of the behaviour of their eigenvalues. We will consider the second ordering, given by J_2 in Equation (6.53), for the rest of our numerical tests.

Table 6.1: The maximum and minimum eigenvalues (or real part in the complex case) of the matrix J_1 and the diagonal preconditioned matrix $J_1P_5^{-1}$ of the CHHS model, where the imaginary parts do change slightly as can be seen in the figures 6.3-6.7. The main effect is to bunch the eigenvalues much more closely to 1.

Grid level	Min ($Re\,\lambda(J_1)$)	Max ($Re\,\lambda(J_1)$)	Min ($Re\,\lambda(J_1P_5^{-1})$)	Max ($Re\,\lambda(J_1P_5^{-1})$)
9^2	0.4751	1.0001	0.7001	1.1499
17^2	0.4797	1.0023	0.5627	1.2186
33^2	0.4809	1.0127	0.4167	1.2916

Table 6.2: The maximum and minimum eigenvalues (or real part in the complex case) of the matrix J_1 and the upper triangular preconditioned matrix $J_1 P_6^{-1}$ of the CHHS model, where the imaginary parts do change slightly as can be seen in the figures 6.8-6.10. The main effect is to bunch the eigenvalues much more closely to 1.

Grid level	Min $(Re\,\lambda(J_1))$	Max $(Re\,\lambda(J_1))$	Min $(Re\,\lambda(J_1 P_6^{-1}))$	Max $(Re\,\lambda(J_1 P_6^{-1}))$
9^2	0.4751	1.0001	0.8122	1.2617
17^2	0.4797	1.0023	0.6566	1.7087
33^2	0.4809	1.0127	0.4572	3.9230

Figure 6.3: The eigenvalues of the coefficients of the original matrix J_1 on grid size 17^2.

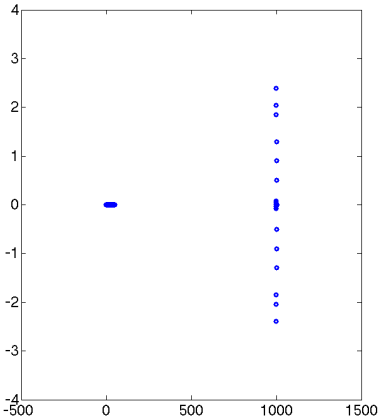


Figure 6.4: The eigenvalues of the diagonal preconditioned matrix $J_1 P_5^{-1}$ on grid size 17^2.

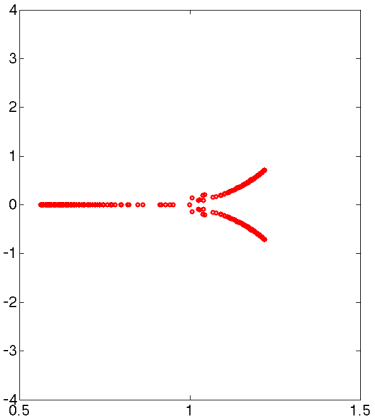


Figure 6.5: The eigenvalues of the original matrix J_1 on the grid size 33^2.

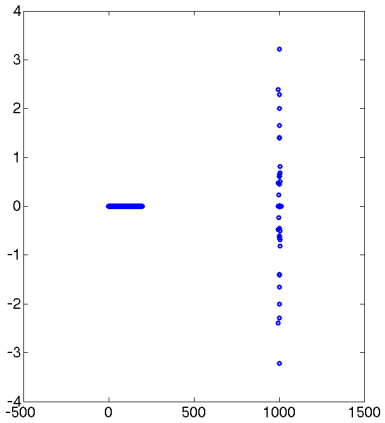


Figure 6.6: The eigenvalues of the diagonal preconditioned matrix $J_1 P_5^{-1}$ on grid size 33^2.

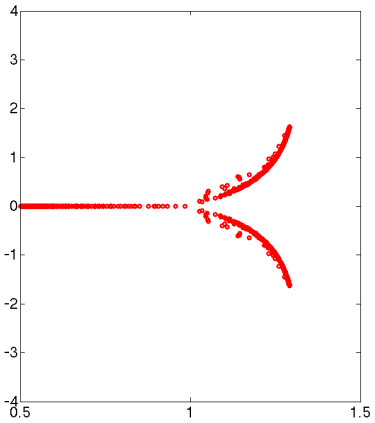


Figure 6.7: The eigenvalues of the original matrix J_1 on grid size 17^2.

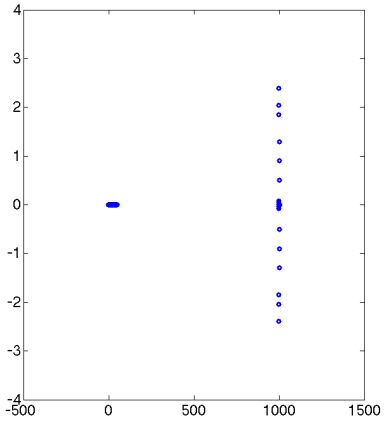


Figure 6.8: The eigenvalues of the upper triangular preconditioned matrix $J_1 P_6^{-1}$ on grid size 17^2.

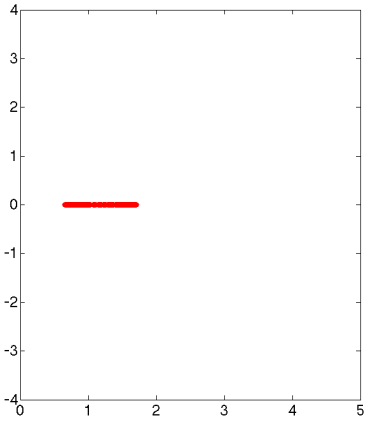


Figure 6.9: The eigenvalues of the original matrix J_1 on grid size 33^2.

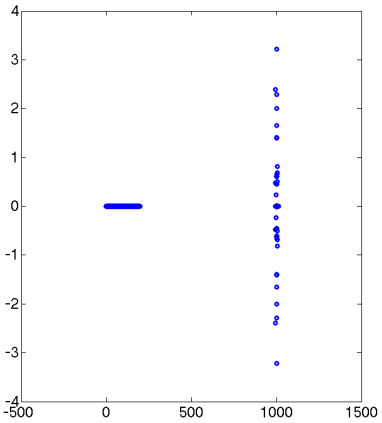
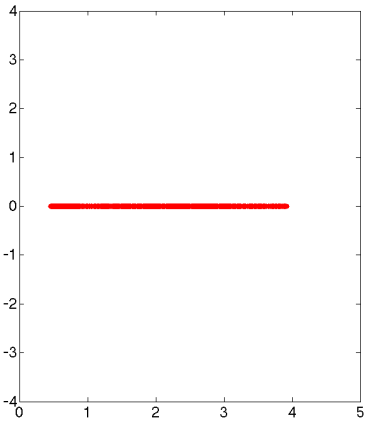


Figure 6.10: The eigenvalues of the upper triangular preconditioned matrix $J_1 P_6^{-1}$ on grid size 33^2.



Now we consider the maximum and minimum eigenvalues of the systems resulting from the P_7 and P_8 preconditioners. In Tables 6.3 and 6.4 we present the minimum and maximum eigenvalues of the original matrix and the minimum and maximum eigenvalues of the preconditioned systems $J_2 P_7^{-1}$ and $J_2 P_8^{-1}$. As we can see from these tables, the eigenvalues of the original matrix J_2 are spread out across a wide range. Conversely, the maximum and minimum eigenvalues of our preconditioned system $J_2 P_7^{-1}$ is bounded in a small range compared to the eigenvalues of the matrix J_2. Furthermore, the range of the eigenvalues of the original matrix J_2 rises with the grid size rises, whereas the range of the eigenvalues of our preconditioner P_7 is fixed and appears independent of the grid size, as shown in Table 6.3.

Although, as we can see in Table 6.4, the maximum eigenvalues for $J_2 P_8^{-1}$ increases as the grid size increases this appears from Figures 6.11 and 6.18 to result from one isolated eigenvalue away from the cluster of eigenvalues. However, this one isolated eigenvalue could be taken care with one iteration of GMRES and it may not affect the convergence of our algorithm. Figure 6.11 to Figure 6.18 shows the spectrum of J_2, $J_2 P_7^{-1}$ and $J_2 P_8^{-1}$ on different grid sizes. We can see clearly from these Figures the impact of the grid size and the effect of the preconditioners on the matrix J_2 in Equation (6.53). From all these figures, we can observe that the eigenvalues of our preconditioned system are limited in a small range far from zero and infinity, and independent of the grid size (except for isolated eigenvalues). In contrast, the eigenvalues of the matrix J_2 are spread out in a broad range and this range increases as the grid size increments.

Table 6.3: The maximum and minimum eigenvalues (or real part in the complex case) of the matrix J_2 and the diagonal preconditioned matrix $J_2P_7^{-1}$ of the CHHS model, where the imaginary parts do change slightly as can be seen in the figures 6.11-6.14. The main effect is to bunch the eigenvalues much more closely to 1.

Grid level	Min $(Re\,\lambda(J_2))$	Max $(Re\,\lambda(J_2))$	Min $(Re\,\lambda(J_2P_7^{-1}))$	Max $(Re\,\lambda(J_2P_7^{-1}))$
9^2	0.4750	5.2321	0.9998	1.0000
17^2	0.4797	178.1767	0.9978	0.9998
33^2	0.4808	741.2419	0.5036	1.0000

Table 6.4: The maximum and minimum eigenvalues (or real part in the complex case) of the matrix J_2 and the upper triangular preconditioned matrix $J_2P_8^{-1}$ of the CHHS model, where there are still extreme eigenvalues present, it does not bunch them all near to 1, but the effect of a small number of outlying eigenvalues can be effectively removed with the Krylov subspace iterations. (e.g. Figure 6.16 and Figure 6.18).

Grid level	Min $(Re\,\lambda(J_2))$	Max $(Re\,\lambda(J_2))$	Min $(Re\,\lambda(J_2P_8^{-1}))$	Max $(Re\,\lambda(J_2P_8^{-1}))$
9^2	0.4750	5.2321	1.0000	354.9371
17^2	0.4797	178.1767	1.0000	732.8284
33^2	0.4808	741.2419	0.7536	588.2470

Figure 6.11: The eigenvalues of the original matrix J_2 on grid size 17^2.

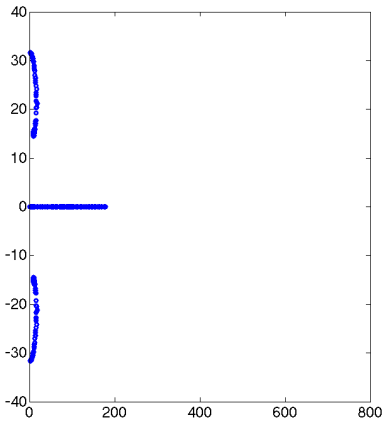


Figure 6.12: The eigenvalues of the diagonal preconditioned matrix $J_2P_7^{-1}$ on grid size 17^2.

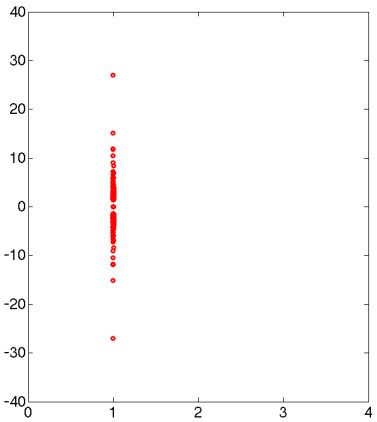


Figure 6.13: The eigenvalues of the original matrix J_2 on grid size 33^2.

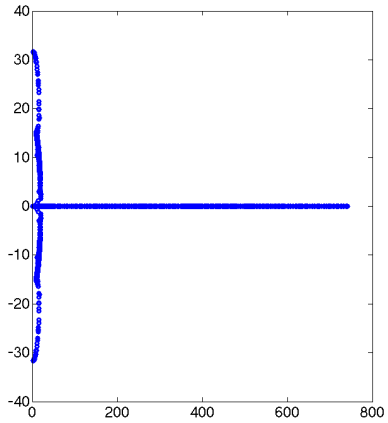


Figure 6.14: The eigenvalues of the diagonal preconditioned matrix $J_2 P_7^{-1}$ on grid size 33^2.

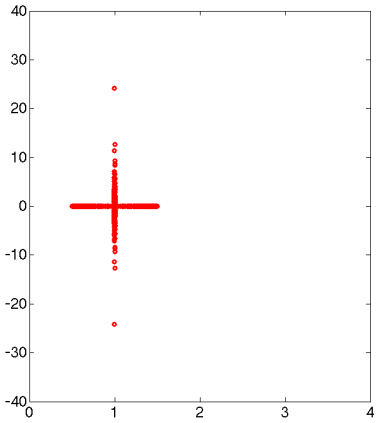


Figure 6.15: The eigenvalues of the original matrix J_2 on grid size 17^2.

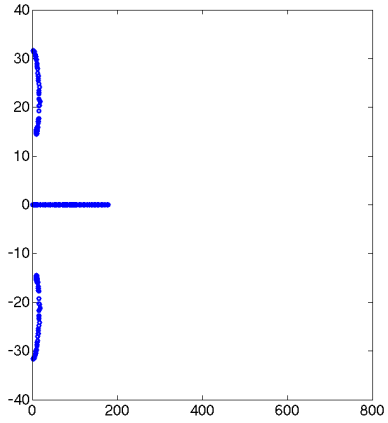


Figure 6.16: The eigenvalues of the upper triangular preconditioned matrix $J_2 P_8^{-1}$ on grid size 17^2.

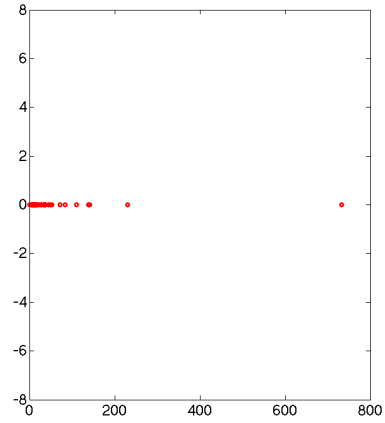


Figure 6.17: The eigenvalues of the original matrix J_2 on grid size 33^2.

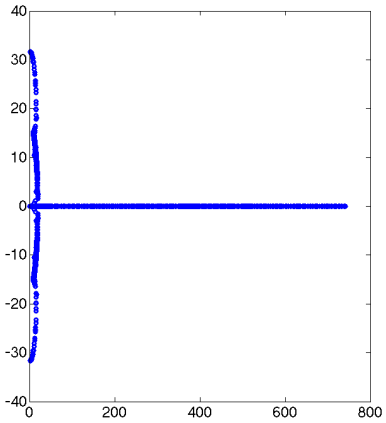
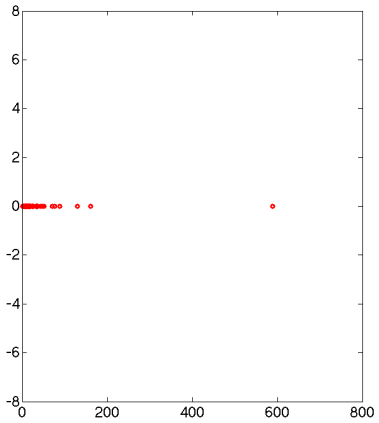


Figure 6.18: The eigenvalues of the upper triangular preconditioned matrix $J_2 P_8^{-1}$ on grid size 33^2.



The computational cost of applying preconditioner P_8 makes it impractical to apply, however in the next subsection, we consider the performance of the variants P_{8a} and P_{8b}, with other variations. The following section also considers FAS performance on the discrete CHHS system of equations by way of comparison.

6.5 Numerical Results

The purpose of this section is to demonstrate the efficacy of our nonlinear multilevel algorithms which are FAS and Newton-Krylov-AMG for solving the CHHS model. In Subsection 6.5.1, we test the performance of the FAS algorithm with various options in order to select the best parameters. Furthermore, we examined the Newton-Krylov-AMG algorithm and tested various parameters and preconditioners for this model. We will describe these numerical results in more detail in Subsection 6.5.2.

In order to determine the convergence of V-cycles iterations in FAS algorithms and Newton iterations (outer iterations) in the Newton-Krylov-AMG algorithm, we used the residual tolerance $\text{Tol} = 1e-08$ in all cases. For the Newton-Krylov-AMG algorithm, inner tolerance values are varied to limit the convergence of GMRES iterations (inner iterations). We consider the best parameters for both of these algorithms with the weighted Red-Black Gauss-Seidel smoother; which is applied with varying values of the free parameter ω.

To define the CHHS model we choose $\gamma = 2$, $\epsilon = 0.1$ in all cases. There is one Equation (6.1) in this system which is time-dependent and we solve it with a fixed time step $\triangle t = 0.001$. In our

experiments in this chapter the domain is $X_1 = -3.2$, $X_2 = 3.2$, $Y_1 = -3.2$, $Y_2 = 3.2$. In all experiments we used only Dirichlet boundary conditions. The grid dimension are $my = 2^{L_{MAX}} + 1$ and $mx = my$ in directions x and y, where L_{MAX} denotes the maximum grid level. The number of unknown grid points is defined as $nu = nx \cdot ny$, where $nx = mx - 2$ and $ny = my - 2$. The grid size is defined as $dx = (X_2 - X_1)/(mx - 1)$, and $dy = (Y_2 - Y_1)/(my - 1)$. The number of equations is $neq = 3 \cdot nu$.

In the following Table 6.5 the grid level, the grid dimension, unknowns grid points nu and the number of equations $neq = 3 \times nu$ that we have used to solve the Cahn-Hilliard-Hele-Shaw system of equations are stated for reference.

Table 6.5: The grid level, the grid size, unknowns grid points nu and the number of equations $neq = 3 \times nu$ in the discrete Cahn-Hilliard-Hele-Shaw system of equations.

Grid	Grid Size	Unknowns Grid Points	N. of Equations
3	9×9	7×7	147
4	17×17	15×15	675
5	33×33	31×31	2883
6	65×65	63×63	11907
7	129×129	127×127	48387
8	257×257	255×255	195075
9	513×513	511×511	783363

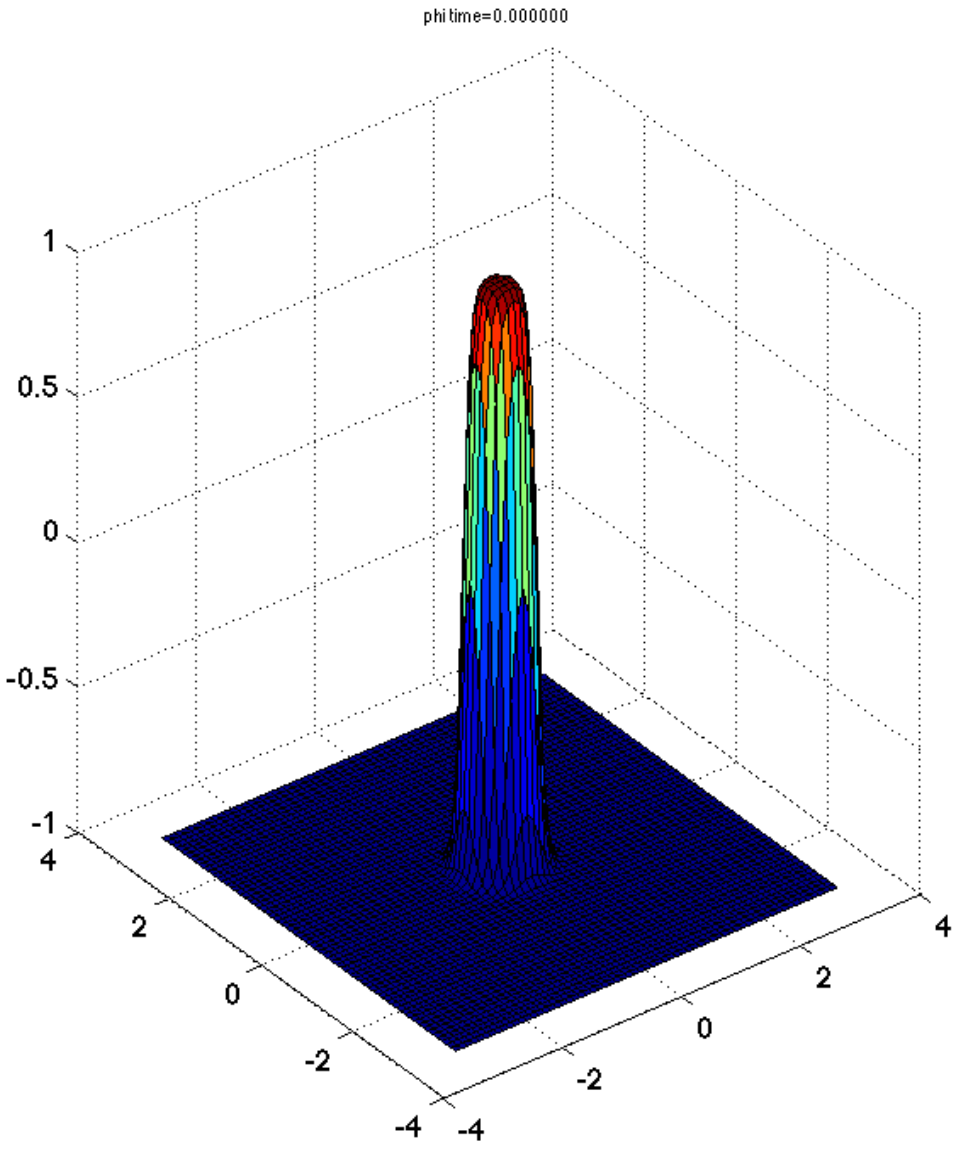


Figure 6.19: The evolution of the numerical solution the Cahn-Hilliard-Hele-Shaw system of equations for ϕ with Grid level= 65^2 in 2D at time T=0.

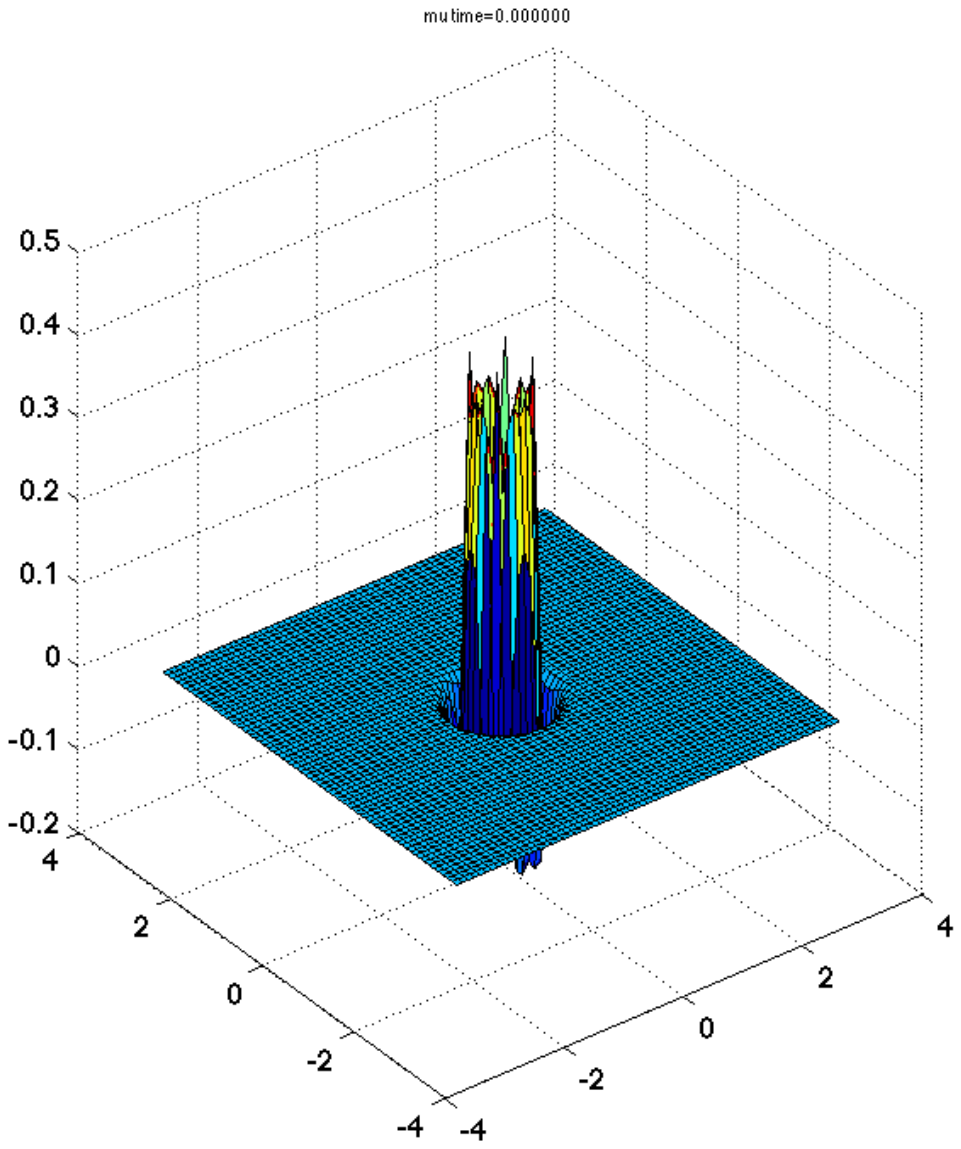


Figure 6.20: The evolution of the numerical solution the Cahn-Hilliard-Hele-Shaw system of equations for μ with Grid level= 65^2 in 2D at time T=0.

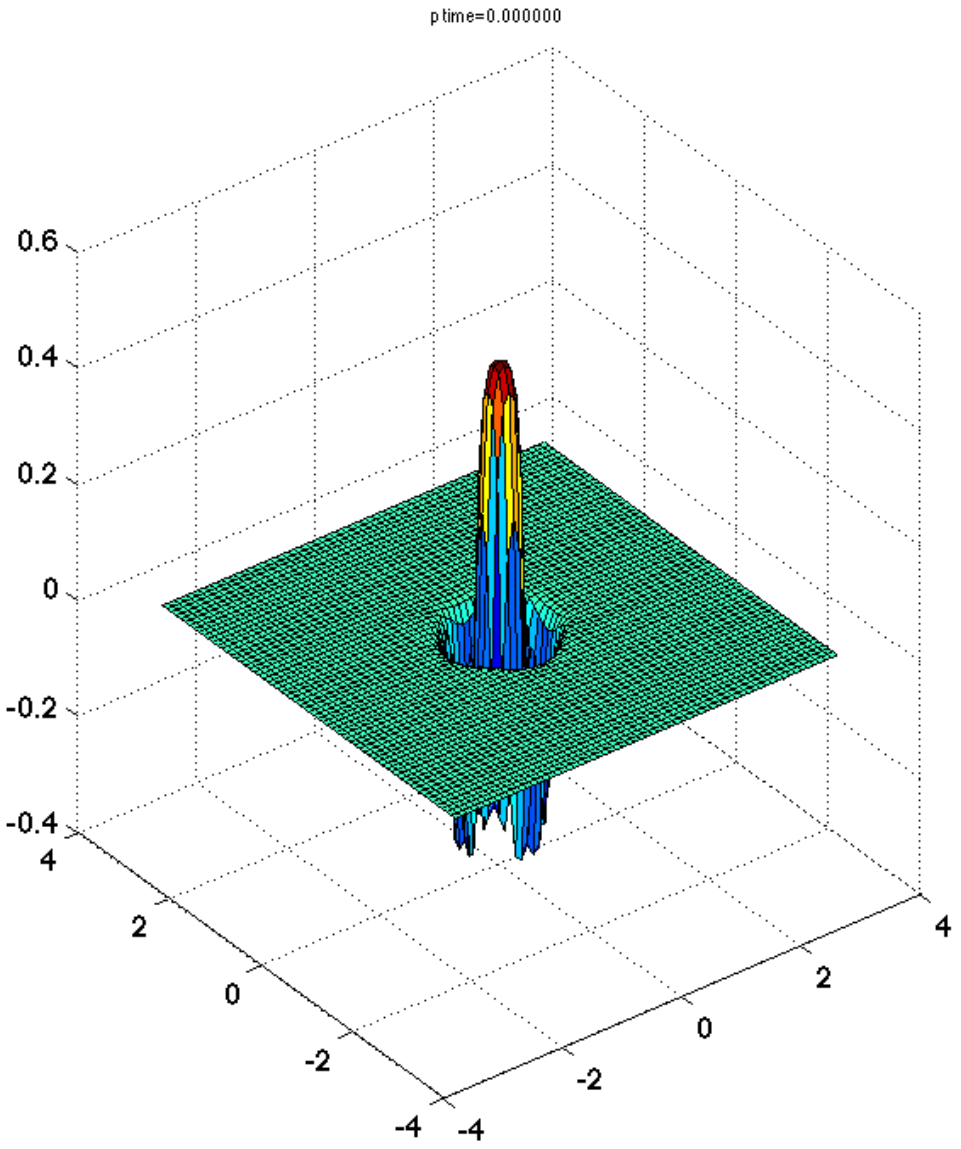


Figure 6.21: The evolution of the numerical solution the Cahn-Hilliard-Hele-Shaw system of equations for p with Grid level= 65^2 in 2D at time T=0.

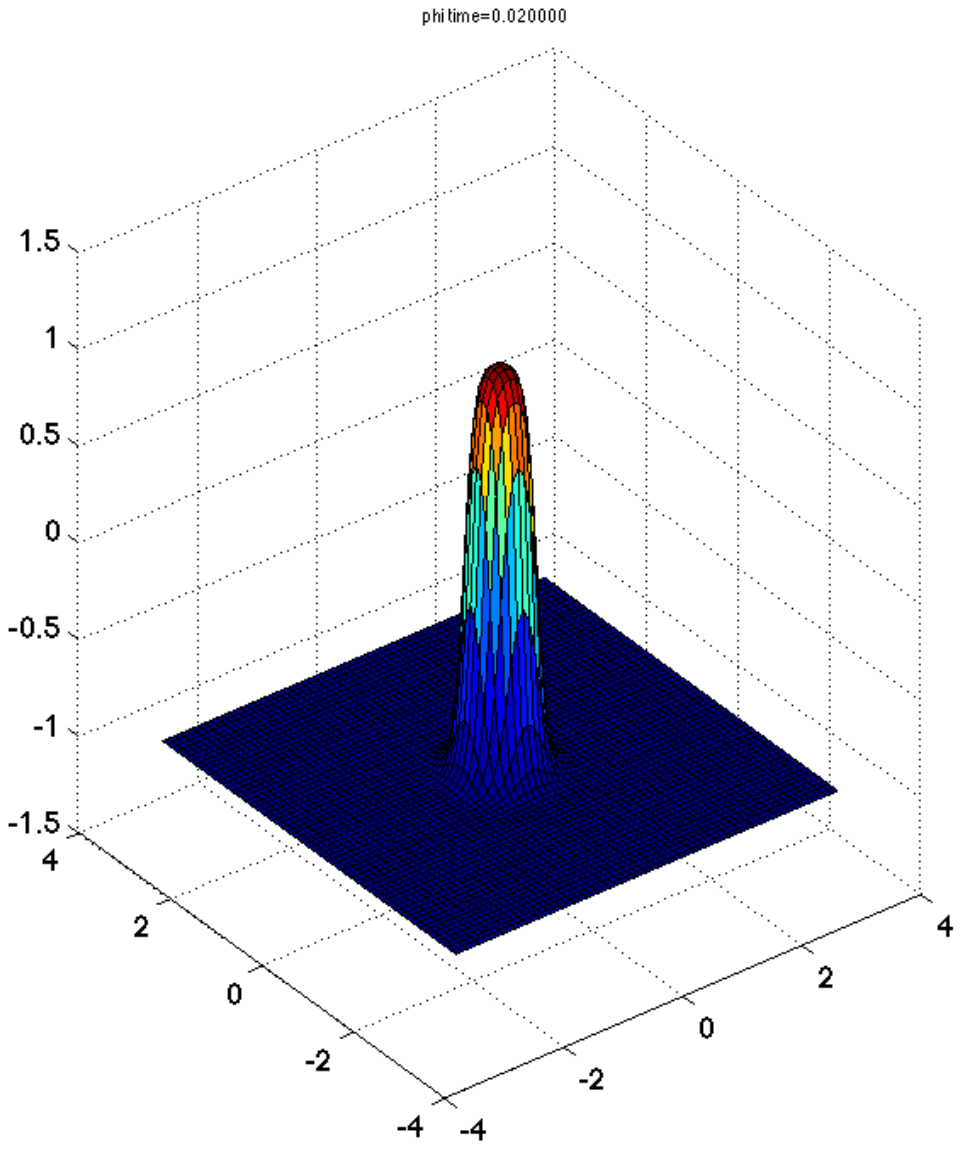


Figure 6.22: The evolution of the numerical solution the Cahn-Hilliard-Hele-Shaw system of equations for ϕ with Grid level= 65^2 in 2D at time T=0.02.

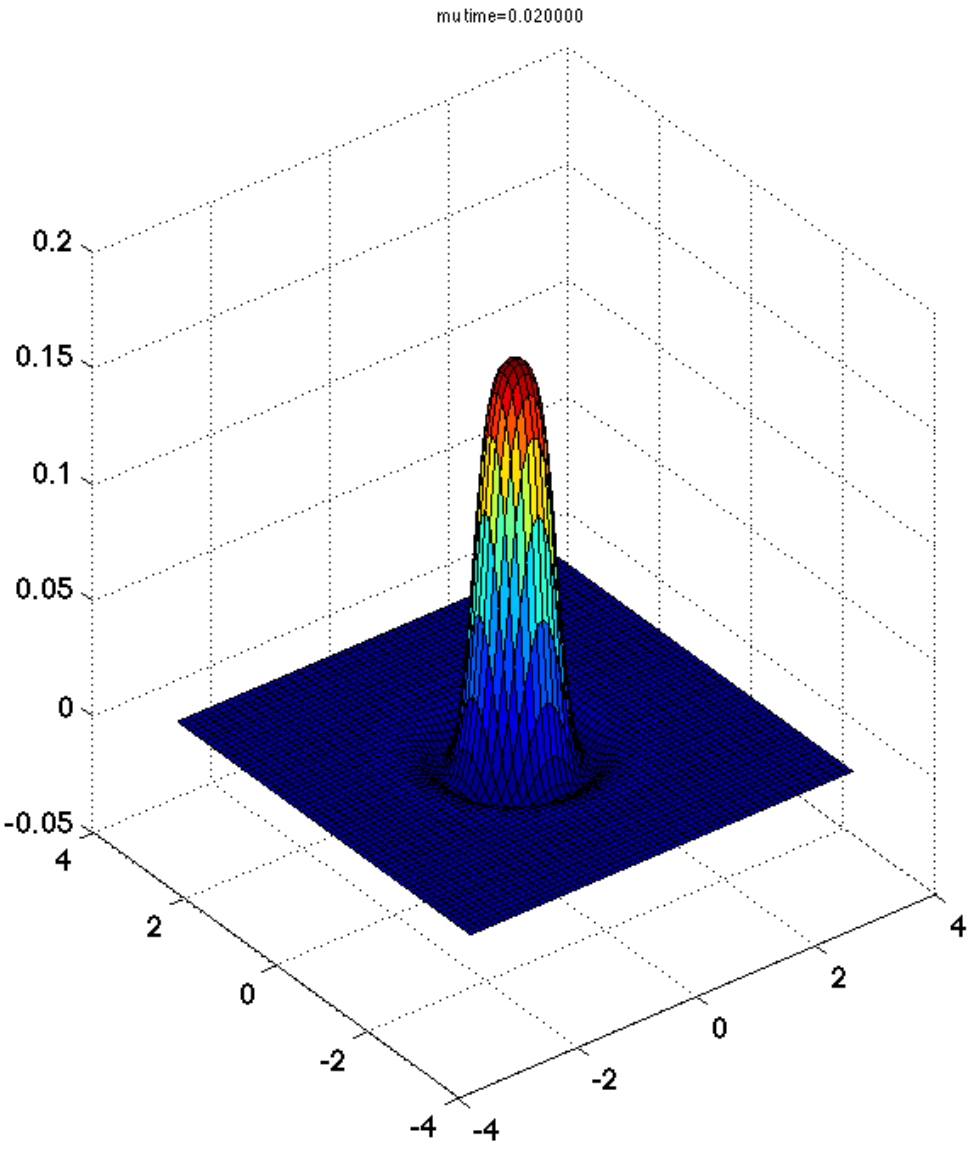


Figure 6.23: The evolution of the numerical solution the Cahn-Hilliard-Hele-Shaw system of equations for μ with Grid level= 65^2 in 2D at time T=0.02.

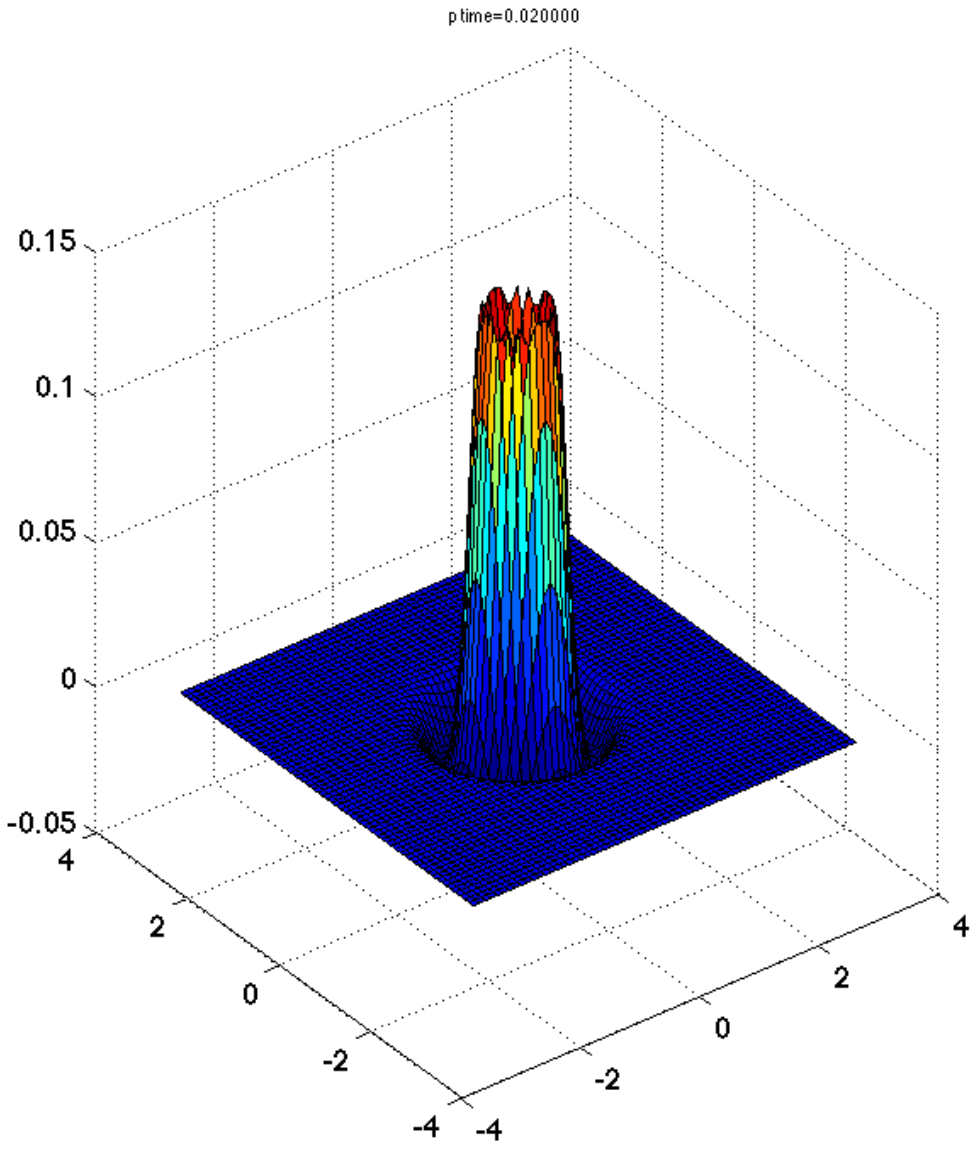
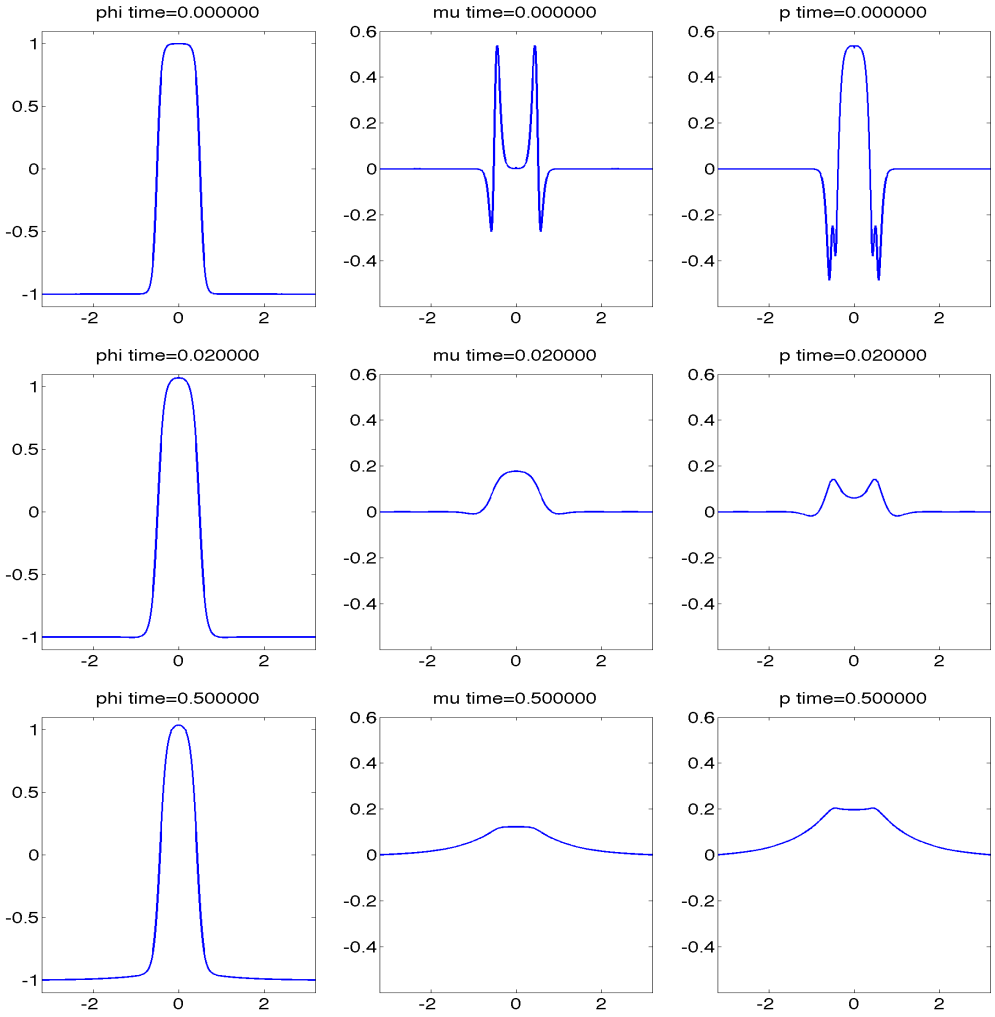


Figure 6.24: The evolution of the numerical solution the Cahn-Hilliard-Hele-Shaw system of equations for p with Grid level= 65^2 in 2D at time T=0.02.

Figures from Figure 6.19 to Figure 6.24, display the evolution of the numerical solution the Cahn- Hilliard-Hele-Shaw system of equations of the initial and the end solution of the CHHS system in 2D for ϕ, μ and p respectively.

The numerical results presented in Figure 6.25 show the evolution of all variables ϕ, μ and p with time for the solution of the Cahn-Hilliard-Hele-Shaw system of equations for grid levels 9. This shows a 1D slice through the centre line of the 2D domain and $(y = 0)$.

Figure 6.25: The evolution of the numerical solutions of the variables ϕ, μ and p for the Cahn-Hilliard-Hele-Shaw system of equations for Grid 9 in 1D at time T=0, T=0.02 and T=0.5.



In a similar way as for the thin film flow model, we will generate numerical results for the CHHS model using different parameters choices so as to compare the best values for each solver. In the following subsections, we explain the different choices of free parameters for the two nonlinear multilevel algorithms, and we will display their numerical results there.

6.5.1 Numerical Results using FAS

For the numerical solution of the time-dependent thin film flow system with the FAS algorithm, in Chapter 5, we observed that the best choice of the parameters for the FAS algorithm was the nonlinear Red-Black Gauss-Seidel smoother with parameter $\omega = 1$, $(pre, post)_{smooth} = (1, 1)$ and $Coarse\ Grid = 5$ with the relative tolerance Tol $= 1e - 8$. Therefore, we will start our experiments in this section with these selections of the free parameters for the FAS algorithm, for solving the Cahn-Hilliard-Hele-Shaw system of equations. Then, we will vary the parameters to investigate what is the best choice of parameters for this system.

FAS with Red Black Gauss-Seidel smoother and $(pre, post)_{smooth} = (1, 1)$, $\omega = 1$ and $Coarse\ Grid = 5$

Table 6.6 presents the performance of the FAS algorithm of the solution of the CHHS model with the Red-Black-G-S smoother, $(pre, post)_{smooth} = (1, 1)$, $\omega = 1$ and $Coarse\ Grid = 5$. Moreover, when the problem size is grown by a factor of 4 the total time is also grown by a factor of 4 which means that linear time complexity of $O(N)$. However, as one can see in Table 6.6 with each row, the total time grows less than factor four, despite, the fact the number of unknowns increases by a factor of four, this may be because that when we solve with $Coarse\ Grid = 5$ this represents quite a lot of work. Hence the proportion of total work on the coarse grid is a big proportion and so we only approach the asymptotic scaling of a factor of four when the fine grid is very fine.

Table 6.6: FAS performance for solving the CHHS model with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol $= 1e - 8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
6	5	1	1	7	8	8.00	34.5098
7	5	1	1	8	8	8.00	65.2943
8	5	1	1	8	8	8.00	191.0100
9	5	1	1	8	9	8.15	696.3082

FAS with Red Black Gauss-Seidel smoother and varying damping factor ω

We now consider the nonlinear Red-Black Gauss-Seidel smoother with varying value of the free parameter ω. From Table 6.7 to Table 6.12, we use varying $\omega = 0.8$, 0.9, 1, 1.1, 1.2 and 1.5, with the aim of determining the optimal value for the nonlinear Red-Black Gauss-Seidel smoother with a $(1, 1)$ pre- and post-smoother and the (coarsest) grid level $Coarse\ Grid = 3$. Note that the optimal choice of ω is not expected to depend upon the choice of grid. As in the previous

chapter we observed that the best choice for the parameter ω for the Red-Black Gauss-Seidel smoother is $\omega = 1$. Therefore, we use $\omega = 1$ in the FAS algorithm with the Red-Black Gauss-Seidel smoother with a $(1, 1)$ pre- and post-smoother as the best choices, as shown in Table 6.13.

Table 6.7: FAS performance for solving the CHHS model with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps $= 20$ with fixed time step size $\triangle t = 0.001$, with nonlinear Red Black G-S with $\omega = 0.8$ and the relative tolerance for FAS is Tol $= 1e - 8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	9	10	9.75	8.2364
6	3	1	1	10	10	10.0	15.9341
7	3	1	1	10	10	10.0	53.7230
8	3	1	1	10	10	10.0	208.4905
9	3	1	1	10	10	10.0	844.5687

Table 6.8: FAS performance for solving the CHHS model with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps $= 20$ with fixed time step size $\triangle t = 0.001$, with nonlinear Red Black G-S with $\omega = 0.9$ and the relative tolerance for FAS is Tol $= 1e - 8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	8	8	8.00	6.0530
6	3	1	1	9	10	9.20	14.8112
7	3	1	1	10	10	10.0	54.1207
8	3	1	1	10	10	10.0	206.6188
9	3	1	1	10	10	10.0	823.2040

Table 6.9: FAS performance for solving the CHHS model with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	6	7	6.25	6.7972
6	3	1	1	8	8	8.00	13.4090
7	3	1	1	8	9	8.95	48.3214
8	3	1	1	9	10	9.95	205.4950
9	3	1	1	9	10	9.95	816.8447

Table 6.10: FAS performance for solving the CHHS model with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, with nonlinear Red Black G-S with $\omega = 1.1$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	6	6	6.00	4.7087
6	3	1	1	9	10	9.75	16.0541
7	3	1	1	8	9	8.90	48.9073
8	3	1	1	8	10	9.85	207.8731
9	3	1	1	8	10	9.95	849.3583

Table 6.11: FAS performance for solving the CHHS model with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, with nonlinear Red Black G-S with $\omega = 1.2$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	7	8	7.70	6.0455
6	3	1	1	10	10	10.0	16.4973
7	3	1	1	9	10	9.95	55.0479
8	3	1	1	9	10	9.95	211.2513
9	3	1	1	9	10	9.95	831.5742

Table 6.12: FAS performance for solving the CHHS model with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, with nonlinear Red Black G-S with $\omega = 1.5$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	3	1	1	10	10	10	7.5102
6	3	1	1	10	10	10	16.4433
7	3	1	1	10	10	10	55.7032
8	3	1	1	10	10	10	212.5388
9	3	1	1	10	10	10	841.1717

Whilst numerical results in Tables 6.7, 6.8, 6.9, 6.10, 6.11 and 6.12 suggest that for the best and optimal total execution time of the FAS algorithm for the solution of the CHHS system of equations is $\omega = 1$, if we change this value we still observe close to optimal numerical results. That is, we have an almost fixed number of V-cycles and the time is scaling almost linearly in all cases, presenting evidence that the FAS algorithm is acting close to optimally.

FAS with Red Black Gauss-Seidel smoother and $\omega = 1$ and with varying *Coarse grid level*

Results so far show a coarse level of $Coarse\ Grid = 5$ is superior to $Coarse\ Grid = 3$. In Table 6.13, we consider $Coarse\ Grid = 4$ using the same parameters. By examining Table 6.6, Table 6.9 and Table 6.13 and we observe that the best coarse grid level for the FAS algorithm to solve this problem is $Coarse\ Grid = 4$ (see Table 6.13).

Table 6.13: FAS performance for solving the CHHS model with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	4	1	1	6	6	6.00	7.4894
6	4	1	1	7	8	7.50	15.8361
7	4	1	1	8	8	8.00	47.3338
8	4	1	1	8	8	8.00	171.1253
9	4	1	1	8	9	8.15	691.6424

FAS with Red Black Gauss-Seidel smoother and varying $(pre, post)_{smooth}$

We applied grid level $Coarse\ Grid = 4$, in Table 6.14 and Table 6.15, using $(pre, post)_{smooth} = (2, 1)$ and $(pre, post)_{smooth} = (2, 2)$, with $\omega = 1$ and $G = 4$. As we can see in Table 6.14 and Table 6.15, even though the number of V-cycles decreased the computational cost increased. Consequently, we observed that the best choice of $(pre, post)_{smooth}$ is $(1, 1)$ with $\omega = 1$ and $Coarse\ Grid = 4$ in Table 6.13.

Table 6.14: FAS performance for solving the CHHS model with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol $= 1e - 8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	4	2	1	5	5	5.00	6.8284
6	4	2	1	6	6	6.00	16.0370
7	4	2	1	6	6	6.00	51.7765
8	4	2	1	7	7	7.00	226.2201
9	4	2	1	7	7	7.00	866.6310

Table 6.15: FAS performance for solving the CHHS model with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol $= 1e - 8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	4	2	2	4	5	4.75	7.1577
6	4	2	2	5	6	5.85	19.2701
7	4	2	2	6	7	6.40	71.6780
8	4	2	2	6	7	6.55	280.1953
9	4	2	2	6	7	6.75	1.1123e+03

Summary

The best results in terms of the total execution time that we have obtained are presented in Table 6.13. From Table 6.7 to Table 6.11 we show a sequence of experiments where we used FAS with the Red-Black Gauss-Seidel smoother, $(pre, post)_{smooth} = (1, 1)$ and relative tolerance Tol $= 1e-8$ with the coarsest $Coarse\ Grid = 3$ and differing values for the parameter $\omega = 0.8$, 0.9, 1, 1.1, 1.2 and 1.5. These experimental results suggest that our implementation is close to optimal for the CHHS model. In Table 6.14 and Table 6.15, we changed the number

of pre- and post- smooth with using Red-Black Gauss-Seidel with $\omega = 1$, Tol=$1e-8$ and the coarsest level of grid $Coarse\ Grid = 4$. We observed that more applications of the smoother always raises the execution time, despite decreasing the number of overall cycles. The test in Table 6.13 shows that FAS with the nonlinear Red-Black Gauss-Seidel smoother with parameter $\omega = 1$, $(pre, post)_{smooth} = (1, 1)$, $Coarse\ Grid = 4$ is the best selection of the free parameters, based on all tests that we have achieved for this model.

6.5.2 Numerical Results using Newton-Krylov-AMG

In this subsection, we investigate and test the numerical solutions presented by applying the Newton-Krylov solver with our proposed preconditioners to solve the CHHS system in 2D. The numerical results that follow include several choices for the values of the parameters for the various preconditioners, in order to determine the best preconditioning strategy based upon our proposed block structure.

Newton-Krylov with P_7, associated with J_2

Since we will consider the alternative ordering for the system of equations that leads to the Jacobian given by J_2 in Equation (6.53), we start our experiments by examining the P_7 preconditioner. The preconditioner P_7 in Equation (6.69) is a block diagonal preconditioner which contains an approximation of the Schur Complement in Equation (6.66) for the original matrix J_2, and this is solved exactly using a direct solver (backslash in MATLAB). We display the performance of Newton-Krylov with P_7 preconditioner with $Tol = 1e-3$ for the GMRES iterations in Table 6.16. As is to be expected, this is not an optimal preconditioner (in terms of time complexity), because we solve the diagonal blocks using direct linear algebra. Moreover, for each fine grid level, it has an almost fixed number of Newton iterations and an almost fixed number of GMRES iterations which means that this preconditioner could be the being for a preconditioning strategy if this direct solver can be eliminated.

Table 6.16: Solving the CHHS model using Newton-Krylov with P_7 which is block diagonal preconditioner, using (backslash) direct solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the running times (in seconds), over 20 steps with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-3$ and $Restart = 150$ and Newton $Tol = 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	4	4.00	23	66	52.94	21.3791
6	4	5	4.05	24	86	65.98	128.1302
7	4	5	4.25	24	87	67.86	551.0928
8	4	5	4.85	24	88	70.59	2.9480e+03

Newton-Krylov with P_8, associated with J_2

The preconditioner P_8 in Equation (6.70) is an upper block triangular preconditioner which is based upon our approximation of the Schur Complement in Equation (6.66). As in the previous example the diagonal blocks are solved using a direct solver (backslash in MATLAB). In Table 6.17 we present the performance of Newton-Krylov with P_8 preconditioner with $Tol = 1e-3$ for the GMRES iterations. As expected, this preconditioner is not an optimal (in terms of the time executions), since we solve the diagonal blocks using a direct linear algebra. However, it has almost a constant number of Newton iterations and almost constant number of GMRES iterations which again indicates that it may be the basis for a better preconditioner. We, therefore, consider two possible modifications.

Table 6.17: Solving the CHHS model using Newton-Krylov with P_8 which is block upper triangle preconditioner, using (backslash) direct solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the running times (in seconds), over 20 steps with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-3$ and $Restart = 150$ and Newton $Tol = 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	4	4.00	13	35	27.55	10.3939
6	4	4	4.00	10	43	32.20	60.0039
7	4	5	4.25	10	44	34.44	313.9960
8	5	5	5.00	9	45	34.92	1.5416e+03

Newton-Krylov-AMG with P_{8a}, associated with J_2

The preconditioner P_{8a} in Equation (6.71) is an upper block triangular preconditioner which is solved using a direct solver (backslash) for the first diagonal block $(1,1)$, and using one V-cycle of the algebraic multigrid method for the other two blocks on the diagonal. Table 6.18, illustrates the performance of Newton-Krylov solver with preconditioner P_{8a} using the GMRES tolerance $Tol = 1e-3$ and Newton $Tol = 1e-8$. It can be seen from this table that the numerical results achieved using this preconditioner are close to optimal than for P_8 preconditioner. This is because the number of Newton iterations is constant, and GMRES iterations appear almost fixed. The use of the direct solver (backslash) in the first diagonal block $(1,1)$ is likely to present perfect scaling, however.

Table 6.18: Solving the CHHS model using Newton-Krylov with P_{8a} preconditioner, using (backslash) direct solver for the first block in the diagonal with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-3$ and $Restart = 150$ and Newton $Tol = 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	4	4	20	40	33.43	6.7713
6	4	4	4	16	47	38.85	30.3869
7	5	5	5	17	51	42.44	171.1324
8	5	5	5	16	51	42.10	674.4229

Newton-Krylov-AMG with P_{8b}, associated with J_2 with varying $(pre, post)_{smooth}$

We now consider the P_{8b} preconditioner in Equation (6.72), obtained by replacing the three blocks on the diagonal of P_8 preconditioner with one AMG V-cycle. We used for this preconditioner one AMG V-cycle with Gauss-Seidel smoothing for each diagonal block, and consider the effect of varying the pre-and post-smoothing within the AMG solver, using $(pre, post)_{smooth} = (1, 1)$ and $(pre, post)_{smooth} = (2, 2)$.

When we apply the P_{8b} preconditioner, the software implementation that we use for the $AMG(\sigma + K_\epsilon)$ returns a warning at the coarsening stage for the $(1, 1)$ block. This warning states that it was not possible to create a full grid hierarchy for the Jacobian system. This means that the AMG coarsening has failed to achieve the level of coarsening that is targetted, and we will discuss this issue briefly in Subsection 6.5.3. Nevertheless, even with this warning, the running time of the P_{8b} preconditioner is close to optimal. The numerical results of the CHHS system by using the P_{8b} preconditioner are reported in Table 6.20. Tables from 6.19 to 6.20 display a set of tests where we apply different values of $(pre, post)_{smooth}$ with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 which is using the fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-3$ and $Restart = 150$. In these tables, as we can clearly see that the number of Newton iterations are constant and the number of GMRES iterations are almost constant and that the P_{8b} preconditioner is almost optimal in terms of time complexity. Furthermore, these tables show that the better choice for the AMG smoothing strategy is $(pre, post)_{smooth} = (1, 1)$. (Note that the default choice in the AMG software that we have used is $(pre, post)_{smooth} = (2, 2)$).

Table 6.19: Solving the CHHS model using Newton-Krylov with P_{8b} preconditioner which is block upper triangle preconditioned using AMG solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e{-}3$ and $Restart = 150$ with $(pre, post)_{smooth} = (1, 1)$ and Newton $Tol = 1e - 8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	4	4	19	39	32.68	4.4336
6	4	4	4	17	45	37.66	21.9373
7	5	5	5	18	48	40.19	93.0952
8	5	5	5	18	48	40.38	374.9731
9	5	5	5	18	48	40.35	1.8154e+03

Table 6.20: Solving the CHHS model using Newton-Krylov with P_{8b} preconditioner which is block upper triangle preconditioned using AMG solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e{-}3$ and $Restart = 150$ with $(pre, post)_{smooth} = (2, 2)$ and Newton $Tol = 1e - 8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	4	4	19	36	30.81	6.8427
6	4	4	4	17	44	36.56	22.5150
7	5	5	5	18	46	39.11	98.2018
8	5	5	5	18	46	39.07	392.9243
9	5	5	5	18	46	38.99	2.0275e+03

Newton-Krylov-AMG with P_{8b}, associated with J_2 with varying GMRES Tol

Based on the observation that the best of the preconditioners considered so far is the P_{8b} preconditioner, Tables 6.19, 6.21 and 6.22 show a set for experiments for the P_{8b} preconditioner, when we varied the convergence tolerance for GMRES between $Tol = 1e - 3$, $Tol = 1e - 4$ and $Tol = 1e - 6$. In these tests we use the AMG solver with $(pre, post)_{smooth} = (1, 1)$ and again show the system with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 (using the fixed time step size $\triangle t = 0.001$). As we can clearly see in these tables, the number of Newton iterations are almost constant and the number of GMRES iterations are almost constant, and the P_{8b} preconditioner is almost optimal in terms of time complexity in all these cases. However, Table 6.21 shows the best performance of the Newton-Krylov with our preconditioner P_{8b} occurs with $Tol = 1e - 4$ for the GMRES iterations.

Table 6.21: Solving the CHHS model using Newton-Krylov with P_{8b} preconditioner which is block upper triangle preconditioned using AMG solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps $= 20$ with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e{-}4$ and $Restart = 150$ with $(pre, post)_{smooth} = (1, 1)$ and Newton $Tol = 1e - 8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	3	3.00	29	45	40.22	4.0845
6	3	4	3.15	24	54	45.25	20.5059
7	4	4	4.00	25	58	48.51	90.3437
8	4	4	4.00	25	58	48.60	356.1623
9	4	5	4.05	25	58	48.69	1.7426e+03

Table 6.22: Solving the CHHS model using Newton-Krylov with P_{8b} preconditioner which is block upper triangle preconditioned using AMG solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps $= 20$ with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e{-}6$ and $Restart = 150$ with $(pre, post)_{smooth} = (1, 1)$ and Newton $Tol = 1e - 8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	2	3	2.35	45	58	52.42	4.0867
6	3	3	3.00	43	70	59.80	25.7710
7	3	4	3.05	41	74	61.86	89.3078
8	3	4	3.05	41	75	61.85	350.9432
9	3	4	3.15	41	75	62.15	1.8449e+03

Newton-Krylov with P_9, associated with J_2

In this subsection, we consider the different preconditioner associated with J_2 which is P_9. We consider this (and P_{10} in the next subsection) because we have a minor issue in the previous preconditioner, P_{8b}, regarding the difficulty of coarsening the $(1, 1)$ block effectively. In order to avoid this issue, we suggest approximating this preconditioner by replacing this first block only in the diagonal by the identity matrix, which gives the P_9 preconditioner.

From Table 6.23, we can see that the GMRES iterations are slightly increased when the grid size is increasing. Since we are seeking to bound the iteration count as the mesh is refined we can deduce that this is not an ideal preconditioner. Therefore, there is no need to go to further approximation for this preconditioner.

Table 6.23: Solving the CHHS model using Newton-Krylov with P_9 which is block upper triangle preconditioner, using (backslash) direct solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the running times (in seconds), over 20 steps with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-3$ and $Restart = 150$ with $(pre, post)_{smooth} = (2, 2)$ and Newton $Tol = 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	4	3.95	26	50	41.76	12.7220
6	4	4	4.00	27	57	47.68	75.5696
7	4	5	4.10	27	57	49.06	297.8608
8	4	5	4.25	40	84	71.16	2.0583e+03

Newton-Krylov with P_{10}, associated with J_2

An alternative to the previous approximation of the P_8 preconditioner is now considered by examining the P_{10} preconditioner. In order to avoid the issue in the P_{8b} preconditioner, we propose approximating this preconditioner by removing the reaction term in the first block $(1, 1)$, which we hypothesize may have caused the problem with the AMG coarsening. This leads to let another potential preconditioner, which is the P_{10} preconditioner in Equation (6.74).

In Table 6.24, we show the performance of the Newton-Krylov solver with preconditioner P_{10} with fixed $\triangle t = 0.001$, the number of time steps are $= 20$, and GMRES with maximum iteration $Maxit = 300$, $Restart = 150$, and Tol $= 1e-3$. The benefit of this preconditioner is that it has an almost constant number of Newton iterations and an almost constant number of GMRES iterations. However, it is still not optimal in time since we solve the diagonal blocks by using direct linear algebra. This is now addressed through consideration of P_{10a} and P_{10b}.

Table 6.24: Solving the CHHS model using Newton-Krylov with P_{10} which is block upper triangle preconditioner, using (backslash) direct solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the running times (in seconds), over 20 steps with time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-3$ and $Restart = 150$ and Newton $Tol = 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	5	4.30	17	91	68.94	28.1251
6	5	5	5.00	14	102	78.36	191.6535
7	5	6	5.05	14	105	78.72	763.8447
8	5	6	5.10	14	103	78.75	3.5078e+03

Newton-Krylov-AMG with P_{10a}, associated with J_2

In the P_{10a} preconditioner in Equation (6.75), we have done further approximation to the P_{10} preconditioner, in which we used the (backslash) direct solver for the first block in the diagonal and we replaced the two other blocks in the diagonal with one-AMG V-cycle. In Table 6.25, we show the performance of Newton-Krylov with preconditioner P_{10a} with fixed $\triangle t = 0.001$, the number of time steps are $= 20$, and GMRES with maximum iteration $Maxit = 300$, $Restart = 150$ and $Tol = 1e-3$.

This preconditioner is shown to be relatively slow because we solve the $(1,1)$ diagonal block using direct linear algebra. However, this preconditioner has an almost constant number of Newton iterations and a bounded number of GMRES iterations, which means that it may have the potential for optimal behaviour if we can make further improvements.

Table 6.25: Solving the CHHS model using Newton-Krylov with P_{10a} preconditioner, using (backslash) direct solver for the first block in the diagonal and we replaced the two other blocks in the diagonal with one-AMG V-cycle with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps $= 20$ with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-3$ and $Restart = 150$ and Newton $Tol = 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	5	4.20	25	94	76.32	13.6387
6	5	5	5.00	24	109	87.59	91.4400
7	5	6	5.05	24	115	90.21	388.1750
8	5	6	5.15	24	116	91.06	1.5881e+03

Newton-Krylov-AMG with P_{10b}, associated with J_2 with varying $(pre, post)_{smooth}$

In the P_{10b} preconditioner in Equation (6.76), we approximated the first block of P_{10a} using AMG. Table 6.26 and Table 6.27 display the result of the P_{10b} preconditioner with the AMG solver undertaken using $(pre, post)_{smooth} = (1,1)$ and $(pre, post)_{smooth} = (2,2)$ respectively. Note that they appear to be performing very well up to the final mesh, at which point the execution time grows dramatically. This will be discussed further in Subsection 6.5.3.

Table 6.26: Solving the CHHS model using Newton-Krylov with P_{10b} preconditioner which is block upper triangle preconditioned using AMG solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps $= 20$ with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e{-}3$ and $Restart = 150$ also, with $(pre, post)_{smooth} = (1, 1)$ and Newton $Tol = 1e - 8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	4	4.00	39	75	65.61	9.1506
6	4	5	4.80	35	88	75.93	50.9454
7	5	5	5.00	36	91	77.09	181.1328
8	5	6	5.05	37	92	77.28	715.5243
9	5	6	5.20	36	91	77.07	4.3585e+03

Table 6.27: Solving the CHHS model using Newton-Krylov with P_{10b} preconditioner which is block upper triangle preconditioned using AMG solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps $= 20$ with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e{-}3$ and $Restart = 150$ also, with $(pre, post)_{smooth} = (2, 2)$ and Newton $Tol = 1e - 8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	4	4	4.00	38	74	64.37	9.2810
6	4	5	4.80	34	86	73.68	54.0540
7	5	5	5.00	35	90	75.57	210.0908
8	5	6	5.05	35	89	75.31	754.0217
9	5	6	5.20	35	89	75.16	4.4798e+03

Newton-Krylov-AMG with P_{10b}, associated with J_2 with varying GMRES Tol

In Table 6.26, Table 6.28 and Table 6.29, we show a series of tests where we varied the tolerance of GMRES between Tol $= 1e - 03$, Tol $= 1e - 04$ and Tol $= 1e - 06$, where the tolerance of Newton iterations is Tol $= 1e - 08$ and $(pre, post)_{smooth} = (1, 1)$ for the P_{10b} preconditioner. In Table 6.28, we display that the P_{10b} preconditioner with GMRES tolerance $Tol = 1e - 4$ is the best choice. Nevertheless, despite having fixed Newton iterations and almost fixed GMRES iterations the execution time still gives superlinearity when moving from grid 8 to grid 9. Consequentially, this preconditioner is still sub-optimal.

Table 6.28: Solving the CHHS model using Newton-Krylov with P_{10b} preconditioner which is block upper triangle preconditioned using AMG solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps $= 20$ with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e{-}4$ and $Restart = 150$ also, with $(pre, post)_{smooth} = (1, 1)$ and Newton $Tol = 1e - 8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	4	3.40	58	87	75.74	9.7501
6	4	4	4.00	59	103	89.94	51.8372
7	4	5	4.05	60	105	91.15	175.0671
8	4	5	4.10	59	106	91.20	686.3724
9	4	5	4.20	60	106	91.07	3.8034e+03

Table 6.29: Solving the CHHS model using Newton-Krylov with P_{10b} preconditioner which is block upper triangle preconditioned using AMG solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps $= 20$ with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e{-}6$ and $Restart = 300$ also, with $(pre, post)_{smooth} = (1, 1)$ and Newton $Tol = 1e - 8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	3	3.00	82	104	94.51	8.5607
6	3	4	3.05	86	125	108.88	54.0815
7	3	4	3.05	86	128	110.63	169.8623
8	3	4	3.20	86	132	111.68	683.8858
9	3	4	3.30	87	131	111.95	5.7580e+03

Summary

In this subsection, we performed numerical tests to optimize the parameter selection for the Newton-Krylov algorithm with a variety of preconditioners based upon block factorizations and their simplification. From the numerical results displayed in Table 6.16 and Table 6.17 for the P_7 and P_8 preconditioners, we note that the approximations to the Schur Complement, that we propose appear to yield close to optimal iteration counts. However, the use of the direct solver (backslash) means that it is impractical to solve very large problems due to its less effective use of memory and superlinear time growth. Nevertheless, this inspired us to consider further improvements. Numerical results using the P_{8b} preconditioner are given in Tables 6.19, 6.20, 6.21 and 6.22. We found that the best performance of the Newton-Krylov algorithm with P_{8b} preconditioner is provided by Table 6.21, where we used the GMRES tolerance is Tol $= 1e - 4$ and $(pre, post)_{smooth} = (1, 1)$. Each of these cases requires a fixed number of Newton iterations and scale almost linearly in running time, showing that they are close to being optimal precon-

ditioners.

In Tables 6.26, 6.28 and 6.29, we show results of the performance of the Newton-Krylov solver with our P_{10b} preconditioner with different tolerance for GMRES which are $Tol = 1e-3$, $Tol = 1e-4$ and $Tol = 1e-6$ respectively. In order to compare these results, we observed that the best performance of the Newton-Krylov algorithm for solving the CHHS system of equations with the P_{10b} preconditioner is given in Table 6.28, where the GMRES tolerance is Tol $= 1e-4$, $(pre, post)_{smooth} = (1,1)$ and the Newton tolerance is $Tol = 1e-8$. In this table, we found the numerical result has an almost constant number of Newton and GMRES iterations, however, we do not see the desired scaling of linear running time.

In summary, from all these numerical results, we can observe that the best choice of new preconditioner for solving the CHHS system of equations is P_{8b}. This leads to a fixed number of Newton and GMRES iterations as well as an almost linear total execution time. In the next subsection, we discuss and compare the numerical results of the numerical solution of the CHHS system of equations by using the two nonlinear multilevel algorithms to achieve the best numerical solution for the CHHS problem in two dimensions.

6.5.3 Discussion and Comparison

In this chapter, we demonstrated the optimal performance of the FAS algorithm as well as the almost optimal preconditioners that we developed for the Newton-Krylov-AMG algorithm. In this subsection, we compare the FAS and Newton-Krylov-AMG algorithms for the numerical solution of the CHHS problem.

As we can see in Subsection 6.5.1 in all the numerical results for the FAS algorithm we have presented, this algorithm performed optimally, or very nearly optimally, even if we do not make the very best choices of the free parameters. The best numerical results in terms of the total execution time and the number of V-cycles are presented in Table 6.13. We observed that the best value of the parameter ω is $\omega = 1$ with the best smoother Red Black G-S, the best value of the pre- and post-smooth $(pre, post) = (1,1)$ and the best coarse grid size is $G = 4$. For simplicity, we present here the best numerical result of the FAS algorithm again as in Table 6.30.

Table 6.30: FAS performance for solving the CHHS model with with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, with nonlinear Red Black G-S with $\omega = 1$ and the relative tolerance for FAS is Tol = $1e-8$

Input				Output			
Grid Size		Smoother		FAS Solver			Time (sec)
Fine	*Coarse*	*Pre*	*Post*	*Min*	*Max*	*Average*	-
5	4	1	1	6	6	6.00	7.4894
6	4	1	1	7	8	7.50	15.8361
7	4	1	1	8	8	8.00	47.3338
8	4	1	1	8	8	8.00	171.1253
9	4	1	1	8	9	8.15	691.6424

Our experiments presented in subsection 6.5.1 show that when we increase the grid size, the number of V-cycles remains almost fixed which indicates that the performance is independent of the problem size. Furthermore, the total execution time of this algorithm grows by approximately a factor of 4 as the problem size increases by a factor of 4, which implies that our FAS algorithm is optimal, with a linear time complexity of approximately $O(N)$.

The second part of this discussion considers the numerical performance of the Newton-Krylov-AMG algorithm, as we described in detail in Subsection 6.5.2. For our newly proposed preconditioner, we observed that both the P_{8b} and P_{10b} preconditioners are almost optimal and both are qualitatively good preconditioners. However, from our experiments, we observe that the P_{8b} preconditioner is better than the P_{10b} preconditioner. The best performance of the Newton-Krylov-AMG algorithm is for the P_{8b} preconditioner with GMRES tolerance $Tol = 1e-4$ presented in Table 6.21. We present this table again here for simplicity as shown in Table 6.31.

Table 6.31: Solving the CHHS model using Newton-Krylov with P_{8b} preconditioner which is block upper triangle preconditioned using AMG solver with $\gamma = 2.0$, $\varepsilon = 0.1$ and the number of time steps = 20 with fixed time step size $\triangle t = 0.001$, GMRES with maximum iteration $Maxit = 300$, $Tol = 1e-4$ and $Restart = 150$ with $(pre, post)_{smooth} = (1, 1)$ and Newton $Tol = 1e-8$.

Input	Output						
Grid size	Newton Solver			GMRES Solver			Time
Fine	*Min*	*Max*	*Average*	*Min*	*Max*	*Average*	-
5	3	3	3.00	29	45	40.22	4.0845
6	3	4	3.15	24	54	45.25	20.5059
7	4	4	4.00	25	58	48.51	90.3437
8	4	4	4.00	25	58	48.60	356.1623
9	4	5	4.05	25	58	48.69	1.7426e+03

In Table 6.31, the performance of the Newton-Krylov-AMG with the P_{8b} preconditioner is ob-

served to be nearly optimal. It has an almost fixed number of Newton iterations and GMRES iterations. We use AMG as a part of this preconditioner. The AMG coarsening does not compete quite as desired for every block. In particular, we find that the coarsening of the J_{11} diagonal block in the Jacobian matrix sometimes fails to reach the target number of levels on a warning is issued. The numerical results in Table 6.31 show the total time is close to optimal, but not fully optimal because the AMG is less effective due to it not coarsening as we expected. Nevertheless, it is close to optimal and good preconditioner.

Further insight into our P_{8b} preconditioner for the CHHS system can be obtained by considering the structure of the J_{11} diagonal block in more detail. The AMG cannot coarsen the J_{11} block fully. If we consider the block $J_{11} = \frac{\partial F_\mu}{\partial \phi}$, where $[F_\mu(U)]_{i,j}$ is the discrete equation written as,

$$[F_\mu(U)]_{i,j} = -\mu_{i,j}^{n+1} + (\phi_{i,j}^{n+1})^3 - \phi_{i,j}^{n+1}$$

$$-\epsilon^2 \left[\frac{1}{\triangle x^2}(\phi_{i+1,j}^{n+1} - 2\phi_{i,j}^{n+1} + \phi_{i-1,j}^{n+1}) + \frac{1}{\triangle y^2}(\phi_{i,j+j}^{n+1} - 2\phi_{i,j}^{n+1} + \phi_{i,j-1}^{n+1}) \right], \tag{6.77}$$

then, from Equation (6.33) we have,

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j}} = -1 + 3\,(\phi_{i,j})^2 + 2\,(\epsilon)^2 \left[(\frac{1}{\Delta x^2}) + (\frac{1}{\Delta y^2}) \right]. \tag{6.78}$$

In the J_{11} block Jacobian the off-diagonal terms are the Laplacian constant coefficients in Equations (6.31), (6.32), (6.34) and (6.35). On the diagonal of the Jacobian block, we can clearly see from Equation (6.78), there are two main terms: the nonlinear term (or reaction term) and the linear term (or diffusion term). When we apply the block preconditioner, we solve the linear system to find the vector δ_ϕ (the first part of the unknowns),

$$J_{11}\delta_\phi = -F_\mu. \tag{6.79}$$

AMG forms a sequence of coarsen approximations to the J_{11} block. This requires that the diagonal entries do not become negative during this process, which is guaranteed for SPD matrices. In this case of J_{11} block we can examine the diagonal (6.78), and see that the sign of this term depends on the current value of $\phi_{(i,j)}$ and the Δx, Δy. In particular, on a sufficiently fine grid (when Δx and Δy are small) the diagonal is always positive. Moreover, on a very coarse grid the diagonal may become negative when $\phi_{(i,j)}$ is close to zero. Hence we cannot guarantee the positive-diagonal property of this matrix and in practice for very coarse grids. This might be a reason why the AMG coarsening can break down. Therefore, to avoid this problem, we suggest ignoring the nonlinear term (the reaction term) in the J_{11} diagonal block producing the P_{10b} preconditioner.

In the P_{10b} preconditioner the AMG coarsening completes, but the preconditioner is a further

approximation since we have removed the source term. Therefore, in practice, P_{10b} is less effective. The numerical results of the P_{10b} preconditioner show that the growth in the total time is slightly sub-optimal (i.e. worse than linear). Nevertheless, it is still close to optimal and overall a good preconditioner in terms of bounding the Newton and GMRES iteration count. The number of iterations needed when using the P_{10b} preconditioner in the Newton-Krylov-AMG was significantly larger than for the P_{8b} preconditioner. In practice, both of these preconditioners have advantages, even though each preconditioner is not quite perfectly optimal in terms of the total execution time.

For comparison, in Table 6.21 we presented the numerical results for the Newton-Krylov-AMG algorithm with the P_{8b} preconditioner and in Table 6.26 we presented the numerical results for the Newton-Krylov-AMG algorithm with the P_{10b} preconditioner. We can see that for the P_{10b} preconditioner more GMRES iterations are required than for the P_{8b} preconditioner, however, interestingly, in the P_{10b} preconditioner, we observed that number of GMRES iterations are almost fixed when the grid size is increased.

We conclude this discussion by the clear conclusion that the better algorithm for solving the CHHS model is our optimal FAS solver. This can be attributed in part to the fact that we have not been able to design a truly perfectly optimal preconditioner in this case, compared to the thin film flow case where the comparison was reversed. Moreover, these numerical results presented in this chapter allow us to confidently say that the FAS algorithm is better than our preconditioned Newton-Krylov algorithm in order to solve the CHHS system.

6.6 Summary

We have presented numerical results to demonstrate that the two nonlinear multilevel algorithms yield almost optimal efficiency for the CHHS problem, however, the best performance is achieved by the FAS algorithm for solving this problem. This is contrast to the thin film model considered in Chapter 5, where we showed that the Newton-Krylov algorithm was the most efficient. In that case, we were able to limit the number of GMRES iterations required at each Newton iteration. In the case of the CHHS system it was possible to design a preconditioner in a similar way but this required further approximations that made it less efficient overall. In this case, the more robust behaviour of the FAS algorithm provided a more efficient solution strategy. We conclude that the numerical results for the nonlinear nonsymmetric problem depend largely on the problem itself, which means that there is not likely to be one approach that is going to be the best, but it is problem dependent.

Chapter 7

Conclusion and Future Research

In this chapter, we summarise the main outcomes of this thesis, highlight the core conclusion and suggest future avenues of research.

7.1 Conclusion

Nonlinear multilevel schemes are established as fast solvers for nonlinear PDEs of Elliptic and Parabolic type. In order to obtain the robust and efficient solution of nonlinear PDE systems, we considered in this thesis three nonlinear multilevel schemes: the Full Approximation Scheme (FAS), the Newton-Multigrid (Newton-MG) algorithm and the Newton-Krylov algorithm with a new preconditioner that we have developed, based on the use of Algebraic Multigrid (AMG).

This research considered the efficient numerical solution of two representative nonlinear systems: the thin film flow model; and, the Cahn-Hilliard-Hele-Shaw (CHHS) model. In both cases, we discretised with FDM in space and an implicit scheme in time. At each time step, the resulting discrete nonlinear algebraic system must be solved efficiently. We have therefore compared and contrasted the three nonlinear multilevel schemes to solve the discrete system for the thin film flow model. Moreover, we have also compared and contrasted the FAS and Newton-Krylov-AMG algorithms when solving the discrete system for the CHHS model. These schemes are all optimal or near optimal (i.e. their computational cost increases close to linearly with the number of degrees of freedom).

We presented extensive numerical results for the numerical solution of the thin film flow in the steady-state and time-dependent problems that demonstrate the optimality of these three approaches, as well as demonstrating, through comparative numerical results, that the best

numerical approach of the three is the Newton-Krylov algorithm with our new block preconditioner based on the use of AMG.

For the discrete CHHS system, we focused on the FAS and Newton-Krylov algorithms. We present extensive results for the time-dependent case. We designed a new block preconditioner for the Newton-Krylov algorithm. In this case, we could not produce a completely optimal algorithm but results demonstrated close to optimal behaviour.

Conversely here, the numerical solution of the CHHS model implies that FAS is superior to the Newton-Krylov-AMG algorithm in terms of execution time. An advantage of the FAS iteration over the Newton iteration here is in the large Jacobian matrix which does not require to be stored, and hence the memory requirements are smaller. Therefore, for the very large system sizes, the FAS iteration may be achievable whereas the Newton-Krylov iteration is not. However, a novel preconditioner is implemented and developed in Chapters 5 and 6 for two different nonlinear systems which permit straightforward comparisons of the computational time required in the Newton-Krylov-AMG method for each model.

In conclusion, the research carried out and documented in this thesis gives a good basis for the use of multilevel solution algorithms for the efficient and accurate numerical solution of complex nonlinear PDE systems. The numerical experiments that we have performed in this thesis imply that the best choice of numerical algorithm for the nonlinear PDE system is problem dependent, which suggests that it is not possible to develop a single approach that would outperform all other strategies for any given nonlinear system.

7.2 Future work

We have presented these nonlinear multilevel numerical solution methods and compared their computational performance for solving two separate nonlinear systems of equations. Nonlinear systems can be found in many aspects of mathematics, science and engineering and it would be interesting to study further nonlinear systems and solve them with the algorithms developed here. In this section, we will suggest some possible avenues for future research.

- It is likely that our nonlinear multilevel algorithms can be extended to other nonlinear PDE systems that can be expressed in Parabolic form, for example, the multi-phase flow (in tumour growth [25] or in porous media [134]).
- It would be interesting to examine how the use of a different stencil for FDM, or an FEM approximation, would affect the numerical algorithms presented here.
- One could develop and implement the application of a single AMG V-cycle and use it in our preconditioner as part of the Newton-Krylov algorithm rather than use the Harwell

Subroutine Library (HSL). This may give more insight into the Newton-Krylov algorithm for the CHHS system that was solved in this work.

– A theoretical analysis of the nonlinear multilevel schemes that we have considered in this thesis can be made in order to provide insight into how the nonlinear multilevel schemes can be anticipated to perform in practice. Brabazon [17] presented such an analysis for single nonlinear PDEs.

– One limitation of our current approach is the memory usage. Further work could explore more memory efficient implementations of our algorithm. It is possible, for example, to extend the CHHS model to 3D and therefore adjust our currently accurate and efficient nonlinear multilevel algorithms in order to make this model more efficient in memory to be practical for actual computation in 3D.

Appendices

Appendix A

The Jacobian Matrix for the Thin Film Flow Model

In Appendix A, we generate the entries of the matrix in Equation (5.18) with applied Dirichlet boundary conditions. The internal points of this matrix are the same as the analytical Jacobian matrix that we have presented in Chapter 5 in Section 5.4.2. However, we illustrate here the modified matrix entries near boundaries.

From Equations (5.15) and (5.16) we can simply derive the Jacobian terms for the boundary conditions. These terms can be used to build the analytical Jacobian efficiently and in the same sparse format for the thin film flow model as before in Chapter 5.

A.1 F_p Equation

We differentiate Equation (5.16) with respect to $h_{i+1,j}$, $h_{i,j}$ and $h_{i,j+1}$ to obtain the left bottom points for J_{11} as follows:

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i+1,j}} = \frac{6}{(\Delta x)^2}, \tag{A.1}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j}} = -2\,(\frac{6}{(\Delta x)^2} + \frac{6}{(\Delta y)^2}), \tag{A.2}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j+1}} = \frac{6}{(\Delta y)^2}, \tag{A.3}$$

where $p_{i-1,j} = p_{i,j-1} = 0$ and $h_{i-1,j} = h_{i,j-1} = 1$ from the Dirichlet boundary conditions and $i = 1$, $j = 1$.

We differentiate Equation (5.16) with respect to $h_{i+1,j}$, $h_{i-1,j}$, $h_{i,j}$ and $h_{i,j+1}$ to obtain the

center bottom points for J_{11} as follows:

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i+1,j}} = \frac{6}{(\Delta x)^2}, \tag{A.4}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i-1,j}} = \frac{6}{(\Delta x)^2}, \tag{A.5}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j}} = -2(\frac{6}{(\Delta x)^2} + \frac{6}{(\Delta y)^2}), \tag{A.6}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j+1}} = \frac{6}{(\Delta y)^2}, \tag{A.7}$$

where $p_{i,j-1} = 0$ and $h_{i,j-1} = 1$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = 1$.

We differentiate Equation (5.16) with respect to $h_{i+1,j}$, $h_{i-1,j}$, $h_{i,j}$ and $h_{i,j+1}$ to obtain the right bottom points for J_{11} as follows:

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i-1,j}} = \frac{6}{(\Delta x)^2}, \tag{A.8}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j}} = -2(\frac{6}{(\Delta x)^2} + \frac{6}{(\Delta y)^2}), \tag{A.9}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j+1}} = \frac{6}{(\Delta y)^2}, \tag{A.10}$$

where $p_{i+1,j} = p_{i,j-1} = 0$ and $h_{i+1,j} = h_{i,j-1} = 1$ from the Dirichlet boundary conditions and $i = N$, $j = 1$.

We differentiate Equation (5.16) with respect to $h_{i-1,j}$, $h_{i,j}$, $h_{i,j-1}$ and $h_{i,j+1}$ to obtain the centre right points for J_{11} as follows:

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i-1,j}} = \frac{6}{(\Delta x)^2}, \tag{A.11}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j}} = -2(\frac{6}{(\Delta x)^2} + \frac{6}{(\Delta y)^2}), \tag{A.12}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j+1}} = \frac{6}{(\Delta y)^2}, \tag{A.13}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j-1}} = \frac{6}{(\Delta y)^2}, \tag{A.14}$$

where $p_{i+1,j} = 0$ and $h_{i+1,j} = 1$ from the Dirichlet boundary conditions and $i = N$, $j = 2, ..., M-$

1.

We differentiate Equation (5.16) with respect to $h_{i+1,j}$, $h_{i-1,j}$, $h_{i,j}$ and $h_{i,j+1}$ to obtain the top right points for J_{11} as follows:

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i-1,j}} = \frac{6}{(\Delta x)^2}, \tag{A.15}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j}} = -2(\frac{6}{(\Delta x)^2} + \frac{6}{(\Delta y)^2}), \tag{A.16}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j-1}} = \frac{6}{(\Delta y)^2}, \tag{A.17}$$

where $p_{i+1,j} = p_{i,j+1} = 0$ and $h_{i+1,j} = h_{i,j+1} = 1$ from the Dirichlet boundary conditions and $i = N$, $j = M$.

We differentiate Equation (5.16) with respect to $h_{i-1,j}$, $h_{i,j}$, $h_{i,j-1}$ and $h_{i,j+1}$ to obtain the top centre points for J_{11} as follows:

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i+1,j}} = \frac{6}{(\Delta x)^2}, \tag{A.18}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i-1,j}} = \frac{6}{(\Delta x)^2}, \tag{A.19}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j}} = -2(\frac{6}{(\Delta x)^2} + \frac{6}{(\Delta y)^2}), \tag{A.20}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j-1}} = \frac{6}{(\Delta y)^2}, \tag{A.21}$$

where $p_{i,j+1} = 0$ and $h_{i,j+1} = 1$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = M$.

We differentiate Equation (5.16) with respect to $h_{i+1,j}$, $h_{i-1,j}$, $h_{i,j}$ and $h_{i,j+1}$ to obtain the top left points for J_{11} as follows:

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i+1,j}} = \frac{6}{(\Delta x)^2}, \tag{A.22}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j}} = -2(\frac{6}{(\Delta x)^2} + \frac{6}{(\Delta y)^2}), \tag{A.23}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j-1}} = \frac{6}{(\Delta y)^2}, \tag{A.24}$$

where $p_{i-1,j} = p_{i,j+1} = 0$ and $h_{i-1,j} = h_{i,j+1} = 1$ from the Dirichlet boundary conditions and $i = 1$, $j = M$.

We differentiate Equation (5.16) with respect to $h_{i+1,j}$, $h_{i,j}$, $h_{i,j-1}$ and $h_{i,j+1}$ to obtain the centre left points for J_{11} as follows:

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i+1,j}} = \frac{6}{(\Delta x)^2}, \tag{A.25}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j}} = -2(\frac{6}{(\Delta x)^2} + \frac{6}{(\Delta y)^2}), \tag{A.26}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j+1}} = \frac{6}{(\Delta y)^2}, \tag{A.27}$$

$$\frac{\partial F_{P_{i,j}}}{\partial h_{i,j-1}} = \frac{6}{(\Delta y)^2}, \tag{A.28}$$

where $p_{i-1,j} = 0$ and $h_{i-1,j} = 1$ from the Dirichlet boundary conditions and $i = 1$, $j = 2, ..., M-1$.

A.2 F_h Equation

We differentiate Equation (5.15) with respect to $h_{i+1,j}$, $h_{i,j}$ and $h_{i,j+1}$ to obtain the left bottom points for J_{21} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i+1,j}} = \frac{1}{2(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^2\left(\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta x}\right)-2\right)\right], \tag{A.29}$$

$$\begin{aligned}\frac{\partial F_{h_{i,j}}}{\partial h_{i,j}} = &\frac{1}{2(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^2\left(\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta x}\right)-2\right)-\left(\frac{h_{i,j}+1}{2}\right)^2\left(\left(\frac{p_{i,j}}{\Delta x}\right)-2\right)\right] \\ &+\frac{1}{2(\Delta y)}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^2\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta y}\right)-\left(\frac{h_{i,j}+1}{2}\right)^2\left(\frac{p_{i,j}}{\Delta y}\right)\right],\end{aligned} \tag{A.30}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j+1}} = \frac{1}{2(\Delta y)}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^2\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta y}\right)\right], \tag{A.31}$$

where $p_{i-1,j} = p_{i,j-1} = 0$ and $h_{i-1,j} = h_{i,j-1} = 1$ from the Dirichlet boundary conditions and $i = 1$, $j = 1$.

We differentiate Equation (5.15) with respect to $h_{i+1,j}$, $h_{i-1,j}$, $h_{i,j}$ and $h_{i,j+1}$ to obtain the center bottom points for J_{21} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i+1,j}} = \frac{1}{2(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^2\left(\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta x}\right)-2\right)\right], \tag{A.32}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i-1,j}} = \frac{-1}{2(\Delta x)}\left[\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^2\left(\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta x}\right)-2\right)\right], \tag{A.33}$$

$$\begin{aligned}\frac{\partial F_{h_{i,j}}}{\partial h_{i,j}} = \frac{1}{2(\Delta x)}&\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^2\left(\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta x}\right)-2\right)-\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^2\left(\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta x}\right)-2\right)\right]\\ &+\frac{1}{2(\Delta y)}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^2\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta y}\right)-\left(\frac{h_{i,j}+1}{2}\right)^2\left(\frac{p_{i,j}}{\Delta y}\right)\right],\end{aligned} \tag{A.34}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j+1}} = \frac{1}{2(\Delta y)}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^2\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta y}\right)\right], \tag{A.35}$$

where $p_{i,j-1}=0$ and $h_{i,j-1}=1$ from the Dirichlet boundary conditions and $i=2,...,N-1$, $j=1$.

We differentiate Equation (5.15) with respect to $h_{i-1,j}$, $h_{i,j}$ and $h_{i,j+1}$ to obtain the right bottom points for J_{21} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i-1,j}} = \frac{-1}{2(\Delta x)}\left[\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^2\left(\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta x}\right)-2\right)\right], \tag{A.36}$$

$$\begin{aligned}\frac{\partial F_{h_{i,j}}}{\partial h_{i,j}} = \frac{1}{2(\Delta x)}&\left[\left(\frac{1+h_{i,j}}{2}\right)^2\left(\left(\frac{-p_{i,j}}{\Delta x}\right)-2\right)-\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^2\left(\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta x}\right)-2\right)\right]\\ &+\frac{1}{2(\Delta y)}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^2\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta y}\right)-\left(\frac{h_{i,j}+1}{2}\right)^2\left(\frac{p_{i,j}}{\Delta y}\right)\right],\end{aligned} \tag{A.37}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j+1}} = \frac{1}{2(\Delta y)}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^2\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta y}\right)\right], \tag{A.38}$$

where $p_{i+1,j}=p_{i,j-1}=0$ and $h_{i+1,j}=h_{i,j-1}=1$ from the Dirichlet boundary conditions and $i=N$, $j=1$.

We differentiate Equation (5.15) with respect to $h_{i-1,j}$, $h_{i,j}$ and $h_{i,j+1}$, $h_{i,j-1}$ to obtain the center right points for J_{21} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i-1,j}} = \frac{-1}{2(\Delta x)}\left[\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^2\left(\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta x}\right)-2\right)\right], \tag{A.39}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j}} = \frac{1}{2(\Delta x)}\left[\left(\frac{1+h_{i,j}}{2}\right)^2\left(\left(\frac{-p_{i,j}}{\Delta x}\right)-2\right)-\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^2\left(\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta x}\right)-2\right)\right]$$

$$+\frac{1}{2(\Delta y)}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^2\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta y}\right)-\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^2\left(\frac{p_{i,j}-p_{i,j-1}}{\Delta y}\right)\right], \tag{A.40}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j+1}} = \frac{1}{2(\Delta y)}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^2\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta y}\right)\right], \tag{A.41}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j-1}} = \frac{-1}{2(\Delta y)}\left[\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^2\left(\frac{p_{i,j}-p_{i,j-1}}{\Delta y}\right)\right], \tag{A.42}$$

where $p_{i+1,j} = 0$ and $h_{i+1,j} = 1$ from the Dirichlet boundary conditions and $i = N$, $j = 2, ..., M-1$.

We differentiate Equation (5.15) with respect to $h_{i-1,j}$, $h_{i,j}$ and $h_{i,j-1}$ to obtain the top right points for J_{21} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i-1,j}} = \frac{-1}{2(\Delta x)}\left[\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^2\left(\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta x}\right)-2\right)\right], \tag{A.43}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j}} = \frac{1}{2(\Delta x)}\left[\left(\frac{1+h_{i,j}}{2}\right)^2\left(\left(\frac{-p_{i,j}}{\Delta x}\right)-2\right)-\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^2\left(\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta x}\right)-2\right)\right]$$

$$+\frac{1}{2(\Delta y)}\left[\left(\frac{1+h_{i,j}}{2}\right)^2\left(\frac{-p_{i,j}}{\Delta y}\right)-\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^2\left(\frac{p_{i,j}-p_{i,j-1}}{\Delta y}\right)\right], \tag{A.44}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j-1}} = \frac{-1}{2(\Delta y)}\left[\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^2\left(\frac{p_{i,j}-p_{i,j-1}}{\Delta y}\right)\right], \tag{A.45}$$

where $p_{i+1,j} = p_{i,j+1} = 0$ and $h_{i+1,j} = h_{i,j+1} = 1$ from the Dirichlet boundary conditions and $i = N$, $j = M$.

We differentiate Equation (5.15) with respect to $h_{i+1,j}$, $h_{i-1,j}$, $h_{i,j}$ and $h_{i,j-1}$ to obtain the center top points for J_{21} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i+1,j}} = \frac{1}{2(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^2\left(\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta x}\right)-2\right)\right], \tag{A.46}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i-1,j}} = \frac{-1}{2(\Delta x)}\left[\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^2\left(\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta x}\right)-2\right)\right], \tag{A.47}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j}} = \frac{1}{2(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^2\left(\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta x}\right)-2\right)-\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^2\left(\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta x}\right)-2\right)\right]$$

$$+\frac{1}{2(\Delta y)}\left[\left(\frac{1+h_{i,j}}{2}\right)^2\left(\frac{-p_{i,j}}{\Delta y}\right)-\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^2\left(\frac{p_{i,j}-p_{i,j-1}}{\Delta y}\right)\right], \tag{A.48}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j-1}} = \frac{-1}{2(\Delta y)}\left[\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^2\left(\frac{p_{i,j}-p_{i,j-1}}{\Delta y}\right)\right], \tag{A.49}$$

where $p_{i,j+1} = 0$ and $h_{i,j+1} = 1$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = M$.

We differentiate Equation (5.15) with respect to $h_{i+1,j}$, $h_{i-1,j}$, $h_{i,j}$ and $h_{i,j-1}$ to obtain the top bottom points for J_{21} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i+1,j}} = \frac{1}{2(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^2\left(\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta x}\right)-2\right)\right], \tag{A.50}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j}} = \frac{1}{2(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^2\left(\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta x}\right)-2\right)-\left(\frac{h_{i,j}+1}{2}\right)^2\left(\left(\frac{p_{i,j}}{\Delta x}\right)-2\right)\right]$$

$$+\frac{1}{2(\Delta y)}\left[\left(\frac{1+h_{i,j}}{2}\right)^2\left(\frac{-p_{i,j}}{\Delta y}\right)-\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^2\left(\frac{p_{i,j}-p_{i,j-1}}{\Delta y}\right)\right], \tag{A.51}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j-1}} = \frac{-1}{2(\Delta y)}\left[\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^2\left(\frac{p_{i,j}-p_{i,j-1}}{\Delta y}\right)\right], \tag{A.52}$$

where $p_{i-1,j} = p_{i,j+1} = 0$ and $h_{i-1,j} = h_{i,j+1} = 1$ from the Dirichlet boundary conditions and $i = 1$, $j = M$.

We differentiate Equation (5.15) with respect to $h_{i+1,j}$, $h_{i-1,j}$, $h_{i,j}$ and $h_{i,j-1}$ to obtain the center left points for J_{21} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i+1,j}} = \frac{1}{2(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^2\left(\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta x}\right)-2\right)\right], \tag{A.53}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j}} = \frac{1}{2(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^2\left(\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta x}\right)-2\right)-\left(\frac{h_{i,j}+1}{2}\right)^2\left(\left(\frac{p_{i,j}}{\Delta x}\right)-2\right)\right]$$

$$+\frac{1}{2(\Delta y)}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^2\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta y}\right)-\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^2\left(\frac{p_{i,j}-p_{i,j-1}}{\Delta y}\right)\right], \tag{A.54}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j+1}} = \frac{1}{2(\Delta y)}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^2\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta y}\right)\right], \tag{A.55}$$

$$\frac{\partial F_{h_{i,j}}}{\partial h_{i,j-1}} = \frac{-1}{2(\Delta y)}\left[\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^2\left(\frac{p_{i,j}-p_{i,j-1}}{\Delta y}\right)\right], \tag{A.56}$$

where $p_{i-1,j} = 0$ and $h_{i-1,j} = 1$ from the Dirichlet boundary conditions and $i = 1$, $j = 2, ..., M-1$.

We differentiate Equation (5.15) with respect to $p_{i+1,j}$, $p_{i,j}$ and $p_{i,j+1}$ to obtain the left bottom points for J_{22} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i+1,j}} = \frac{1}{3\,(\Delta x)^2}\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^3, \tag{A.57}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j}} = \frac{1}{3\,(\Delta x)^2}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^3+\left(\frac{h_{i,j}+1}{2}\right)^3\right]+\frac{1}{3\,(\Delta y)^2}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^3+\left(\frac{h_{i,j}+1}{2}\right)^3\right], \tag{A.58}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j+1}} = \frac{1}{3\,(\Delta y)^2}\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^3, \tag{A.59}$$

where $p_{i-1,j} = p_{i,j-1} = 0$ and $h_{i-1,j} = h_{i,j-1} = 1$ from the Dirichlet boundary conditions and $i = 1$, $j = 1$.

We differentiate Equation (5.15) with respect to $p_{i+1,j}$, $p_{i,j}$, $p_{i,j-1}$ and $p_{i,j+1}$ to obtain the centre right points for J_{22} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i-1,j}} = \frac{1}{3\,(\Delta x)^2}\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3, \tag{A.60}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j}} = \frac{1}{3\,(\Delta x)^2}\left[\left(\frac{1+h_{i,j}}{2}\right)^3 + \left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3\right] + \frac{1}{3\,(\Delta y)^2}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^3 + \left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^3\right], \tag{A.61}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j+1}} = \frac{1}{3\,(\Delta y)^2}\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^3, \tag{A.62}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j-1}} = \frac{1}{3\,(\Delta y)^2}\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^3, \tag{A.63}$$

where $p_{i+1,j} = 0$ and $h_{i+1,j} = 1$ from the Dirichlet boundary conditions and $i = N$, $j = 2, ..., M-1$.

We differentiate Equation (5.15) with respect to $p_{i+1,j}$, $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j+1}$ to obtain the center bottom points for J_{22} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i+1,j}} = \frac{1}{3\,(\Delta x)^2}\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^3, \tag{A.64}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i-1,j}} = \frac{1}{3\,(\Delta x)^2}\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3, \tag{A.65}$$

$$\begin{aligned}\frac{\partial F_{h_{i,j}}}{\partial p_{i,j}} = {} & \frac{1}{3\,(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^3\frac{1}{(\Delta x)} + \left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3\frac{1}{(\Delta x)}\right] \\ & + \frac{1}{3\,(\Delta y)}\left[\left(\frac{h_{i,j}+h_{i,j+1}}{2}\right)^3\frac{1}{(\Delta y)} + \left(\frac{h_{i,j}+1}{2}\right)^3\frac{1}{(\Delta y)}\right],\end{aligned} \tag{A.66}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j+1}} = \frac{1}{3\,(\Delta y)^2}\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^3, \tag{A.67}$$

where $p_{i,j-1} = 0$ and $h_{i,j-1} = 1$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = 1$.

We differentiate Equation (5.15) with respect to $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j+1}$ to obtain the right bottom points for J_{22} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i-1,j}} = \frac{1}{3\,(\Delta x)^2}\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3, \tag{A.68}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j}} = \frac{1}{3\,(\Delta x)}\left[\left(\frac{1+h_{i,j}}{2}\right)^3\frac{1}{(\Delta x)} + \left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3\frac{1}{(\Delta x)}\right]$$

$$+\frac{1}{3\left(\Delta y\right)}\left[\left(\frac{h_{i,j}+h_{i,j+1}}{2}\right)^3\frac{1}{(\Delta y)}+\left(\frac{h_{i,j}+1}{2}\right)^3\frac{1}{(\Delta y)}\right], \tag{A.69}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j+1}}=\frac{1}{3\left(\Delta y\right)^2}\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^3, \tag{A.70}$$

where $p_{i+1,j}=p_{i,j-1}=0$ and $h_{i+1,j}=h_{i,j-1}=1$ from the Dirichlet boundary conditions and $i=N$, $j=1$.

We differentiate Equation (5.15) with respect to $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j+1}$ to obtain the top right points for J_{22} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i-1,j}}=\frac{1}{3\left(\Delta x\right)^2}\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3, \tag{A.71}$$

$$\begin{aligned}\frac{\partial F_{h_{i,j}}}{\partial p_{i,j}}=&\frac{1}{3\left(\Delta x\right)}\left[\left(\frac{1+h_{i,j}}{2}\right)^3\frac{1}{(\Delta x)}+\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3\frac{1}{(\Delta x)}\right]\\&+\frac{1}{3\left(\Delta y\right)}\left[\left(\frac{1+h_{i,j}}{2}\right)^3\frac{1}{(\Delta y)}+\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^3\frac{1}{(\Delta y)}\right],\end{aligned} \tag{A.72}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j-1}}=\frac{1}{3\left(\Delta y\right)^2}\left(\frac{h_{i,j-1}+h_{i,j}}{2}\right)^3, \tag{A.73}$$

where $p_{i+1,j}=p_{i,j+1}=0$ and $h_{i+1,j}=h_{i,j+1}=1$ from the Dirichlet boundary conditions and $i=N$, $j=M$.

We differentiate Equation (5.15) with respect to $p_{i+1,j}$, $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j-1}$ to obtain the center bottom points for J_{22} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i+1,j}}=\frac{1}{3\left(\Delta x\right)^2}\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^3, \tag{A.74}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i-1,j}}=\frac{1}{3\left(\Delta x\right)^2}\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3, \tag{A.75}$$

$$\begin{aligned}\frac{\partial F_{h_{i,j}}}{\partial p_{i,j}}=&\frac{1}{3\left(\Delta x\right)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^3\frac{1}{(\Delta x)}+\left(\frac{h_{i,j}+h_{i-1,j}}{2}\right)^3\frac{1}{(\Delta x)}\right]\\&+\frac{1}{3\left(\Delta y\right)}\left[\left(\frac{h_{i,j}+1}{2}\right)^3\frac{1}{(\Delta y)}+\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^3\frac{1}{(\Delta y)}\right],\end{aligned} \tag{A.76}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j-1}}=\frac{1}{3\left(\Delta y\right)^2}\left(\frac{h_{i,j-1}+h_{i,j}}{2}\right)^3, \tag{A.77}$$

where $p_{i,j+1} = 0$ and $h_{i,j+1} = 1$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = M$.

We differentiate Equation (5.15) with respect to $p_{i+1,j}$, $p_{i,j}$ and $p_{i,j-1}$ to obtain the top right points for J_{22} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i+1,j}} = \frac{1}{3\,(\Delta x)^2}\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^3, \tag{A.78}$$

$$\begin{aligned}\frac{\partial F_{h_{i,j}}}{\partial p_{i,j}} = &\frac{1}{3\,(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^3\frac{1}{(\Delta x)}+\left(\frac{h_{i,j}+1}{2}\right)^3\frac{1}{(\Delta x)}\right]\\ &+\frac{1}{3\,(\Delta y)}\left[\left(\frac{1+h_{i,j}}{2}\right)^3\frac{1}{(\Delta y)}+\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^3\frac{1}{(\Delta y)}\right],\end{aligned} \tag{A.79}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j-1}} = \frac{1}{3\,(\Delta y)^2}\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^3, \tag{A.80}$$

where $p_{i-1,j} = p_{i,j+1} = 0$ and $h_{i-1,j} = h_{i,j+1} = 1$ from the Dirichlet boundary conditions and $i = 1$, $j = M$.

We differentiate Equation (5.15) with respect to $p_{i+1,j}$, $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j-1}$ to obtain the center left points for J_{22} as follows:

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i+1,j}} = \frac{1}{3\,(\Delta x)^2}\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^3, \tag{A.81}$$

$$\begin{aligned}\frac{\partial F_{h_{i,j}}}{\partial p_{i,j}} = &\frac{1}{3\,(\Delta x)}\left[\left(\frac{h_{i+1,j}+h_{i,j}}{2}\right)^3\frac{1}{(\Delta x)}+\left(\frac{h_{i,j}+1}{2}\right)^3\frac{1}{(\Delta x)}\right]\\ &+\frac{1}{3\,(\Delta y)}\left[\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^3\frac{1}{(\Delta y)}+\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^3\frac{1}{(\Delta y)}\right],\end{aligned} \tag{A.82}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j+1}} = \frac{1}{3\,(\Delta y)^2}\left(\frac{h_{i,j+1}+h_{i,j}}{2}\right)^3, \tag{A.83}$$

$$\frac{\partial F_{h_{i,j}}}{\partial p_{i,j-1}} = \frac{1}{3\,(\Delta y)^2}\left(\frac{h_{i,j}+h_{i,j-1}}{2}\right)^3, \tag{A.84}$$

where $p_{i-1,j} = 0$ and $h_{i-1,j} = 1$ from the Dirichlet boundary conditions and $i = 1$, $j = 2, ..., M-1$.

Appendix B

The Jacobian Matrix for the CHHS Model

In Appendix B, we present the entries of the matrix in Equation (6.10) with applied Dirichlet boundary conditions. The entries at internal points of this matrix are the same as the analytical Jacobian matrix that we have presented in Chapter 6 in Section 6.2.1.

From Equations (6.6), (6.7) and (6.8) we can simply derive the Jacobian terms for the boundary conditions. These terms can be used to build the analytical Jacobian efficiently and in the same sparse format for the CHHS model as before.

B.1 F_ϕ Equation

We differentiate Equation (6.6) with respect to $\phi_{i+1,j}$, $\phi_{i,j}$ and $\phi_{i,j+1}$ to obtain the non-zero entries of J_{11} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[\gamma \left(\phi_{i+1,j} + \phi_{i,j} \right) \left(\mu_{i+1,j} - \mu_{i,j} \right) + (\frac{p_{i+1,j} - p_{i,j}}{2}) \right], \tag{B.1}$$

$$\begin{aligned} \frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j}} = & -(\frac{1}{\Delta t}) + (\frac{1}{\Delta x^2}) \left[\gamma \left(\phi_{i+1,j} + \phi_{i,j} \right) \left(\mu_{i+1,j} - \mu_{i,j} \right) + (\frac{p_{i+1,j} - p_{i,j}}{2}) \right] \\ & -(\frac{1}{\Delta x^2}) \left[\gamma \left(\phi_{i,j} + \phi_{i-1,j} \right) \left(\mu_{i,j} - \mu_{i-1,j} \right) + (\frac{p_{i,j} - p_{i-1,j}}{2}) \right] \\ & +(\frac{1}{\Delta y^2}) \left[\gamma \left(\phi_{i,j+1} + \phi_{i,j} \right) \left(\mu_{i,j+1} - \mu_{i,j} \right) + (\frac{p_{i,j+1} - p_{i,j}}{2}) \right] \\ & -(\frac{1}{\Delta y^2}) \left[\gamma \left(\phi_{i,j} + \phi_{i,j-1} \right) \left(\mu_{i,j} - \mu_{i,j-1} \right) + (\frac{p_{i,j} - p_{i,j-1}}{2}) \right], \end{aligned} \tag{B.2}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[\gamma \left(\phi_{i,j+1} + \phi_{i,j} \right) \left(\mu_{i,j+1} - \mu_{i,j} \right) + (\frac{p_{i,j+1} - p_{i,j}}{2}) \right], \tag{B.3}$$

where $\phi_{i-1,j} = \phi_{i,j-1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = 1$.

We differentiate Equation (6.6) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$ and $\phi_{i,j+1}$ to obtain the non-zero entries of J_{11} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[\gamma \left(\phi_{i+1,j} + \phi_{i,j} \right) \left(\mu_{i+1,j} - \mu_{i,j} \right) + (\frac{p_{i+1,j} - p_{i,j}}{2}) \right], \tag{B.4}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{1}{\Delta x^2}) \left[\gamma \left(\phi_{i,j} + \phi_{i-1,j} \right) \left(\mu_{i,j} - \mu_{i-1,j} \right) + (\frac{p_{i,j} - p_{i-1,j}}{2}) \right], \tag{B.5}$$

$$\begin{aligned} \frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j}} = & -(\frac{1}{\Delta t}) + (\frac{1}{\Delta x^2}) \left[\gamma \left(\phi_{i+1,j} + \phi_{i,j} \right) \left(\mu_{i+1,j} - \mu_{i,j} \right) + (\frac{p_{i+1,j} - p_{i,j}}{2}) \right] \\ & -(\frac{1}{\Delta x^2}) \left[\gamma \left(\phi_{i,j} + \phi_{i-1,j} \right) \left(\mu_{i,j} - \mu_{i-1,j} \right) + (\frac{p_{i,j} - p_{i-1,j}}{2}) \right] \\ & +(\frac{1}{\Delta y^2}) \left[\gamma \left(\phi_{i,j+1} + \phi_{i,j} \right) \left(\mu_{i,j+1} - \mu_{i,j} \right) + (\frac{p_{i,j+1} - p_{i,j}}{2}) \right] \\ & -(\frac{1}{\Delta y^2}) \left[\gamma \left(\phi_{i,j} + \phi_{i,j-1} \right) \left(\mu_{i,j} - \mu_{i,j-1} \right) + (\frac{p_{i,j} - p_{i,j-1}}{2}) \right], \end{aligned} \tag{B.6}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[\gamma \left(\phi_{i,j+1} + \phi_{i,j} \right) \left(\mu_{i,j+1} - \mu_{i,j} \right) + (\frac{p_{i,j+1} - p_{i,j}}{2}) \right], \tag{B.7}$$

where $\phi_{i,j-1} = -1$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = 1$.

We differentiate Equation (6.6) with respect to $\phi_{i-1,j}$, $\phi_{i,j}$ and $\phi_{i,j+1}$ to obtain the non-zero entries of J_{11} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{1}{\Delta x^2}) \left[\gamma \left(\phi_{i,j} + \phi_{i-1,j} \right) \left(\mu_{i,j} - \mu_{i-1,j} \right) + (\frac{p_{i,j} - p_{i-1,j}}{2}) \right], \tag{B.8}$$

$$\begin{aligned} \frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j}} = & -(\frac{1}{\Delta t}) + (\frac{1}{\Delta x^2}) \left[\gamma \left(\phi_{i+1,j} + \phi_{i,j} \right) \left(\mu_{i+1,j} - \mu_{i,j} \right) + (\frac{p_{i+1,j} - p_{i,j}}{2}) \right] \\ & -(\frac{1}{\Delta x^2}) \left[\gamma \left(\phi_{i,j} + \phi_{i-1,j} \right) \left(\mu_{i,j} - \mu_{i-1,j} \right) + (\frac{p_{i,j} - p_{i-1,j}}{2}) \right] \\ & +(\frac{1}{\Delta y^2}) \left[\gamma \left(\phi_{i,j+1} + \phi_{i,j} \right) \left(\mu_{i,j+1} - \mu_{i,j} \right) + (\frac{p_{i,j+1} - p_{i,j}}{2}) \right] \end{aligned}$$

$$-(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j}+\phi_{i,j-1}\right)\left(\mu_{i,j}-\mu_{i,j-1}\right)+(\frac{p_{i,j}-p_{i,j-1}}{2})\right], \tag{B.9}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j+1}}=(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j+1}+\phi_{i,j}\right)\left(\mu_{i,j+1}-\mu_{i,j}\right)+(\frac{p_{i,j+1}-p_{i,j}}{2})\right], \tag{B.10}$$

where $\phi_{i+1,j}=\phi_{i,j-1}=-1$, $\mu_{i+1,j}=p_{i+1,j}=0$ and $\mu_{i,j-1}=p_{i,j-1}=0$ from the Dirichlet boundary conditions and $i=N$, $j=1$.

We differentiate Equation (6.6) with respect to $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{11} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i-1,j}}=-(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i,j}+\phi_{i-1,j}\right)\left(\mu_{i,j}-\mu_{i-1,j}\right)+(\frac{p_{i,j}-p_{i-1,j}}{2})\right], \tag{B.11}$$

$$\begin{aligned}\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j}}=&-(\frac{1}{\Delta t})+(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i+1,j}+\phi_{i,j}\right)\left(\mu_{i+1,j}-\mu_{i,j}\right)+(\frac{p_{i+1,j}-p_{i,j}}{2})\right]\\&-(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i,j}+\phi_{i-1,j}\right)\left(\mu_{i,j}-\mu_{i-1,j}\right)+(\frac{p_{i,j}-p_{i-1,j}}{2})\right]\\&+(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j+1}+\phi_{i,j}\right)\left(\mu_{i,j+1}-\mu_{i,j}\right)+(\frac{p_{i,j+1}-p_{i,j}}{2})\right]\\&-(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j}+\phi_{i,j-1}\right)\left(\mu_{i,j}-\mu_{i,j-1}\right)+(\frac{p_{i,j}-p_{i,j-1}}{2})\right],\end{aligned} \tag{B.12}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j+1}}=(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j+1}+\phi_{i,j}\right)\left(\mu_{i,j+1}-\mu_{i,j}\right)+(\frac{p_{i,j+1}-p_{i,j}}{2})\right], \tag{B.13}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j-1}}=-(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j}+\phi_{i,j-1}\right)\left(\mu_{i,j}-\mu_{i,j-1}\right)+(\frac{p_{i,j}-p_{i,j-1}}{2})\right], \tag{B.14}$$

where $\phi_{i+1,j}-1$ and $\mu_{i+1,j}=p_{i+1,j}=0$ from the Dirichlet boundary conditions and $i=N$, $j=2,...,M-1$.

We differentiate Equation (6.6) with respect to $\phi_{i-1,j}$, $\phi_{i,j}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{11} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i-1,j}}=-(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i,j}+\phi_{i-1,j}\right)\left(\mu_{i,j}-\mu_{i-1,j}\right)+(\frac{p_{i,j}-p_{i-1,j}}{2})\right], \tag{B.15}$$

$$\begin{aligned}\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j}}=&-(\frac{1}{\Delta t})+(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i+1,j}+\phi_{i,j}\right)\left(\mu_{i+1,j}-\mu_{i,j}\right)+(\frac{p_{i+1,j}-p_{i,j}}{2})\right]\\&-(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i,j}+\phi_{i-1,j}\right)\left(\mu_{i,j}-\mu_{i-1,j}\right)+(\frac{p_{i,j}-p_{i-1,j}}{2})\right]\end{aligned}$$

$$+(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j+1}+\phi_{i,j}\right)\left(\mu_{i,j+1}-\mu_{i,j}\right)+(\frac{p_{i,j+1}-p_{i,j}}{2})\right]$$
$$-(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j}+\phi_{i,j-1}\right)\left(\mu_{i,j}-\mu_{i,j-1}\right)+(\frac{p_{i,j}-p_{i,j-1}}{2})\right], \tag{B.16}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j-1}}=-(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j}+\phi_{i,j-1}\right)\left(\mu_{i,j}-\mu_{i,j-1}\right)+(\frac{p_{i,j}-p_{i,j-1}}{2})\right], \tag{B.17}$$

where $\phi_{i+1,j}=\phi_{i,j+1}=-1$, $\mu_{i+1,j}=p_{i+1,j}=0$ and $\mu_{i,j+1}=p_{i,j+1}=0$ from the Dirichlet boundary conditions and $i=N$, $j=M$.

We differentiate Equation (6.6) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{11} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i+1,j}}=(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i+1,j}+\phi_{i,j}\right)\left(\mu_{i+1,j}-\mu_{i,j}\right)+(\frac{p_{i+1,j}-p_{i,j}}{2})\right], \tag{B.18}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i-1,j}}=-(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i,j}+\phi_{i-1,j}\right)\left(\mu_{i,j}-\mu_{i-1,j}\right)+(\frac{p_{i,j}-p_{i-1,j}}{2})\right], \tag{B.19}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j}}=-(\frac{1}{\Delta t})+(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i+1,j}+\phi_{i,j}\right)\left(\mu_{i+1,j}-\mu_{i,j}\right)+(\frac{p_{i+1,j}-p_{i,j}}{2})\right]$$
$$-(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i,j}+\phi_{i-1,j}\right)\left(\mu_{i,j}-\mu_{i-1,j}\right)+(\frac{p_{i,j}-p_{i-1,j}}{2})\right]$$
$$+(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j+1}+\phi_{i,j}\right)\left(\mu_{i,j+1}-\mu_{i,j}\right)+(\frac{p_{i,j+1}-p_{i,j}}{2})\right]$$
$$-(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j}+\phi_{i,j-1}\right)\left(\mu_{i,j}-\mu_{i,j-1}\right)+(\frac{p_{i,j}-p_{i,j-1}}{2})\right], \tag{B.20}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j-1}}=-(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j}+\phi_{i,j-1}\right)\left(\mu_{i,j}-\mu_{i,j-1}\right)+(\frac{p_{i,j}-p_{i,j-1}}{2})\right], \tag{B.21}$$

where $\phi_{i,j+1}=-1$ and $\mu_{i,j+1}=p_{i,j+1}=0$ from the Dirichlet boundary conditions and $i=2,...,N-1$, $j=M$.

We differentiate Equation (6.6) with respect to $\phi_{i+1,j}, \phi_{i,j}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{11} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i+1,j}}=(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i+1,j}+\phi_{i,j}\right)\left(\mu_{i+1,j}-\mu_{i,j}\right)+(\frac{p_{i+1,j}-p_{i,j}}{2})\right], \tag{B.22}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j}}=-(\frac{1}{\Delta t})+(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i+1,j}+\phi_{i,j}\right)\left(\mu_{i+1,j}-\mu_{i,j}\right)+(\frac{p_{i+1,j}-p_{i,j}}{2})\right]$$

$$-(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i,j}+\phi_{i-1,j}\right)\left(\mu_{i,j}-\mu_{i-1,j}\right)+(\frac{p_{i,j}-p_{i-1,j}}{2})\right]$$
$$+(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j+1}+\phi_{i,j}\right)\left(\mu_{i,j+1}-\mu_{i,j}\right)+(\frac{p_{i,j+1}-p_{i,j}}{2})\right]$$
$$-(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j}+\phi_{i,j-1}\right)\left(\mu_{i,j}-\mu_{i,j-1}\right)+(\frac{p_{i,j}-p_{i,j-1}}{2})\right], \tag{B.23}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j}+\phi_{i,j-1}\right)\left(\mu_{i,j}-\mu_{i,j-1}\right)+(\frac{p_{i,j}-p_{i,j-1}}{2})\right], \tag{B.24}$$

where $\phi_{i-1,j} = \phi_{i,j+1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = M$.

We differentiate Equation (6.6) with respect to $\phi_{i+1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{11} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i+1,j}} = (\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i+1,j}+\phi_{i,j}\right)\left(\mu_{i+1,j}-\mu_{i,j}\right)+(\frac{p_{i+1,j}-p_{i,j}}{2})\right], \tag{B.25}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j}} = -(\frac{1}{\Delta t})+(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i+1,j}+\phi_{i,j}\right)\left(\mu_{i+1,j}-\mu_{i,j}\right)+(\frac{p_{i+1,j}-p_{i,j}}{2})\right]$$
$$-(\frac{1}{\Delta x^2})\left[\gamma\left(\phi_{i,j}+\phi_{i-1,j}\right)\left(\mu_{i,j}-\mu_{i-1,j}\right)+(\frac{p_{i,j}-p_{i-1,j}}{2})\right]$$
$$+(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j+1}+\phi_{i,j}\right)\left(\mu_{i,j+1}-\mu_{i,j}\right)+(\frac{p_{i,j+1}-p_{i,j}}{2})\right]$$
$$-(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j}+\phi_{i,j-1}\right)\left(\mu_{i,j}-\mu_{i,j-1}\right)+(\frac{p_{i,j}-p_{i,j-1}}{2})\right], \tag{B.26}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j+1}} = (\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j+1}+\phi_{i,j}\right)\left(\mu_{i,j+1}-\mu_{i,j}\right)+(\frac{p_{i,j+1}-p_{i,j}}{2})\right], \tag{B.27}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{1}{\Delta y^2})\left[\gamma\left(\phi_{i,j}+\phi_{i,j-1}\right)\left(\mu_{i,j}-\mu_{i,j-1}\right)+(\frac{p_{i,j}-p_{i,j-1}}{2})\right], \tag{B.28}$$

where $\phi_{i-1,j} = -1$ and $\mu_{i-1,j} = p_{i-1,j} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = 2, ..., M-1$.

We differentiate Equation (6.6) with respect to $\mu_{i+1,j}$, $\mu_{i,j}$ and $\mu_{i,j+1}$ to obtain the non-zero entries of J_{12} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{1}{\Delta x^2})\left[(1+\gamma\left((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2\right))\right], \tag{B.29}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j}} = -(\frac{1}{\Delta x^2}) \left[(1+\gamma\,((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i-1,j}}{2})^2)) \right]$$
$$-(\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i,j-1}}{2})^2)) \right], \tag{B.30}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2)) \right], \tag{B.31}$$

where $\phi_{i-1,j} = \phi_{i,j-1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = 1$.

We differentiate Equation (6.6) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$ and $\mu_{i,j+1}$ to obtain the non-zero entries of J_{12} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[(1+\gamma\,((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2)) \right], \tag{B.32}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{1}{\Delta x^2}) \left[(1+\gamma\,((\frac{\phi_{i,j}+\phi_{i-1,j}}{2})^2)) \right], \tag{B.33}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j}} = -(\frac{1}{\Delta x^2}) \left[(1+\gamma\,((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i-1,j}}{2})^2)) \right]$$
$$-(\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i,j-1}}{2})^2)) \right], \tag{B.34}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2)) \right], \tag{B.35}$$

where $\phi_{i,j-1} = -1$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = 1$.

We differentiate Equation (6.6) with respect to $\mu_{i-1,j}$, $\mu_{i,j}$ and $\mu_{i,j+1}$ to obtain the non-zero entries of J_{12} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{1}{\Delta x^2}) \left[(1+\gamma\,((\frac{\phi_{i,j}+\phi_{i-1,j}}{2})^2)) \right], \tag{B.36}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j}} = -(\frac{1}{\Delta x^2}) \left[(1+\gamma\,((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i-1,j}}{2})^2)) \right]$$
$$-(\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i,j-1}}{2})^2)) \right], \tag{B.37}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2)) \right], \tag{B.38}$$

where $\phi_{i+1,j} = \phi_{i,j-1} = -1$, $\mu_{i+1,j} = p_{i+1,j} = 0$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = 1$.

We differentiate Equation (6.6) with respect to $\mu_{i-1,j}$, $\mu_{i,j}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{12} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{1}{\Delta x^2}) \left[(1 + \gamma ((\frac{\phi_{i,j} + \phi_{i-1,j}}{2})^2)) \right], \tag{B.39}$$

$$\begin{aligned} \frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j}} = &-(\frac{1}{\Delta x^2}) \left[(1 + \gamma ((\frac{\phi_{i+1,j} + \phi_{i,j}}{2})^2)) + (1 + \gamma ((\frac{\phi_{i,j} + \phi_{i-1,j}}{2})^2)) \right] \\ &-(\frac{1}{\Delta y^2}) \left[(1 + \gamma ((\frac{\phi_{i,j+1} + \phi_{i,j}}{2})^2)) + (1 + \gamma ((\frac{\phi_{i,j} + \phi_{i,j-1}}{2})^2)) \right], \end{aligned} \tag{B.40}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[(1 + \gamma ((\frac{\phi_{i,j+1} + \phi_{i,j}}{2})^2)) \right], \tag{B.41}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{1}{\Delta y^2}) \left[(1 + \gamma ((\frac{\phi_{i,j} + \phi_{i,j-1}}{2})^2)) \right], \tag{B.42}$$

where $\phi_{i+1,j} = -1$ and $\mu_{i+1,j} = p_{i+1,j} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = 2, ..., M-1$.

We differentiate Equation (6.6) with respect to $\mu_{i-1,j}$, $\mu_{i,j}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{12} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{1}{\Delta x^2}) \left[(1 + \gamma ((\frac{\phi_{i,j} + \phi_{i-1,j}}{2})^2)) \right], \tag{B.43}$$

$$\begin{aligned} \frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j}} = &-(\frac{1}{\Delta x^2}) \left[(1 + \gamma ((\frac{\phi_{i+1,j} + \phi_{i,j}}{2})^2)) + (1 + \gamma ((\frac{\phi_{i,j} + \phi_{i-1,j}}{2})^2)) \right] \\ &-(\frac{1}{\Delta y^2}) \left[(1 + \gamma ((\frac{\phi_{i,j+1} + \phi_{i,j}}{2})^2)) + (1 + \gamma ((\frac{\phi_{i,j} + \phi_{i,j-1}}{2})^2)) \right], \end{aligned} \tag{B.44}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{1}{\Delta y^2}) \left[(1 + \gamma ((\frac{\phi_{i,j} + \phi_{i,j-1}}{2})^2)) \right], \tag{B.45}$$

where $\phi_{i+1,j} = \phi_{i,j+1} = -1$, $\mu_{i+1,j} = p_{i+1,j} = 0$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = M$.

We differentiate Equation (6.6) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{12} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[(1 + \gamma ((\frac{\phi_{i+1,j} + \phi_{i,j}}{2})^2)) \right], \tag{B.46}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{1}{\Delta x^2}) \left[(1 + \gamma ((\frac{\phi_{i,j} + \phi_{i-1,j}}{2})^2)) \right], \tag{B.47}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j}} = -(\frac{1}{\Delta x^2}) \left[(1+\gamma\,((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i-1,j}}{2})^2)) \right]$$
$$-(\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i,j-1}}{2})^2)) \right], \tag{B.48}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j}+\phi_{i,j-1}}{2})^2)) \right], \tag{B.49}$$

where $\phi_{i,j+1} = -1$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = M$.

We differentiate Equation (6.6) with respect to $\mu_{i+1,j}$, $\mu_{i,j}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{12} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[(1+\gamma\,((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2)) \right], \tag{B.50}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j}} = -(\frac{1}{\Delta x^2}) \left[(1+\gamma\,((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i-1,j}}{2})^2)) \right]$$
$$-(\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i,j-1}}{2})^2)) \right], \tag{B.51}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j}+\phi_{i,j-1}}{2})^2)) \right], \tag{B.52}$$

where $\phi_{i-1,j} = \phi_{i,j+1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = M$.

We differentiate Equation (6.6) with respect to $\mu_{i+1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{12} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[(1+\gamma\,((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2)) \right], \tag{B.53}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j}} = -(\frac{1}{\Delta x^2}) \left[(1+\gamma\,((\frac{\phi_{i+1,j}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i-1,j}}{2})^2)) \right]$$
$$-(\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2)) + (1+\gamma\,((\frac{\phi_{i,j}+\phi_{i,j-1}}{2})^2)) \right], \tag{B.54}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j+1}+\phi_{i,j}}{2})^2)) \right], \tag{B.55}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{1}{\Delta y^2}) \left[(1+\gamma\,((\frac{\phi_{i,j}+\phi_{i,j-1}}{2})^2)) \right], \tag{B.56}$$

where $\phi_{i-1,j} = -1$ and $\mu_{i-1,j} = p_{i-1,j} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = 2, ..., M-1$.

We differentiate Equation (6.6) with respect to $p_{i+1,j}$, $p_{i,j}$ and $p_{i,j+1}$ to obtain the non-zero entries of J_{13} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} \right], \tag{B.57}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j}} = -(\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right]$$
$$-(\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{B.58}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} \right], \tag{B.59}$$

where $\phi_{i-1,j} = \phi_{i,j-1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = 1$.

We differentiate Equation (6.6) with respect to $p_{i+1,j}$, $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j+1}$ to obtain the non-zero entries of J_{13} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} \right], \tag{B.60}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right], \tag{B.61}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j}} = -(\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right]$$
$$-(\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{B.62}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} \right], \tag{B.63}$$

where $\phi_{i,j-1} = -1$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = 1$.

We differentiate Equation (6.6) with respect to $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j+1}$ to obtain the non-zero entries of J_{13} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right], \tag{B.64}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j}} = -(\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right]$$
$$-(\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{B.65}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} \right], \tag{B.66}$$

where $\phi_{i+1,j} = \phi_{i,j-1} = -1$, $\mu_{i+1,j} = p_{i+1,j} = 0$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = 1$.

We differentiate Equation (6.6) with respect to $p_{i-1,j}$, $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{13} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right], \tag{B.67}$$

$$\begin{aligned} \frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j}} = &-(\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right] \\ &-(\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \end{aligned} \tag{B.68}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} \right], \tag{B.69}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{B.70}$$

where $\phi_{i+1,j} = -1$ and $\mu_{i+1,j} = p_{i+1,j} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = 2, ..., M-1$.

We differentiate Equation (6.6) with respect to $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{13} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right], \tag{B.71}$$

$$\begin{aligned} \frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j}} = &-(\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right] \\ &-(\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \end{aligned} \tag{B.72}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{B.73}$$

where $\phi_{i+1,j} = \phi_{i,j+1} = -1$, $\mu_{i+1,j} = p_{i+1,j} = 0$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = M$.

We differentiate Equation (6.6) with respect to $p_{i+1,j}$, $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{13} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} \right], \tag{B.74}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right], \tag{B.75}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j}} = -(\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right] -(\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{B.76}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{B.77}$$

where $\phi_{i,j+1} = -1$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = M$.

We differentiate Equation (6.6) with respect to $p_{i+1,j}$, $p_{i,j}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{13} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} \right], \tag{B.78}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j}} = -(\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right] -(\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{B.79}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{B.80}$$

where $\phi_{i-1,j} = \phi_{i,j+1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = M$.

We differentiate Equation (6.6) with respect to $p_{i+1,j}$, $p_{i-1,j}$, $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{13} as follows:

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} \right], \tag{B.81}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j}} = -(\frac{1}{\Delta x^2}) \left[\frac{\phi_{i+1,j} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i-1,j}}{2} \right] -(\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} + \frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{B.82}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j+1} + \phi_{i,j}}{2} \right], \tag{B.83}$$

$$\frac{\partial F_{\phi_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}) \left[\frac{\phi_{i,j} + \phi_{i,j-1}}{2} \right], \tag{B.84}$$

where $\phi_{i-1,j} = -1$ and $\mu_{i-1,j} = p_{i-1,j} = 0$ from the Dirichlet boundary conditions and $i = 1$,

$j = 2, ..., M-1$.

B.2 F_μ Equation

We differentiate Equation (6.7) with respect to $\phi_{i+1,j}$, $\phi_{i,j}$ and $\phi_{i,j+1}$ to obtain the non-zero entries of J_{21} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i+1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{B.85}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j}} = -1 + 3\,(\phi_{i,j})^2 + 2\,(\epsilon)^2 \left[(\frac{1}{\Delta x^2}) + (\frac{1}{\Delta y^2})\right], \tag{B.86}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j+1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{B.87}$$

where $\phi_{i-1,j} = \phi_{i,j-1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = 1$.

We differentiate Equation (6.7) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$ and $\phi_{i,j+1}$ to obtain the non-zero entries of J_{21} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i+1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{B.88}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{B.89}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j}} = -1 + 3\,(\phi_{i,j})^2 + 2\,(\epsilon)^2 \left[(\frac{1}{\Delta x^2}) + (\frac{1}{\Delta y^2})\right], \tag{B.90}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j+1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{B.91}$$

where $\phi_{i,j-1} = -1$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = 1$.

We differentiate Equation (6.7) with respect to $\phi_{i-1,j}$, $\phi_{i,j}$ and $\phi_{i,j+1}$ to obtain the non-zero entries of J_{21} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{B.92}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j}} = -1 + 3\,(\phi_{i,j})^2 + 2\,(\epsilon)^2 \left[(\frac{1}{\Delta x^2}) + (\frac{1}{\Delta y^2})\right], \tag{B.93}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j+1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{B.94}$$

where $\phi_{i+1,j} = \phi_{i,j-1} = -1$, $\mu_{i+1,j} = p_{i+1,j} = 0$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet

boundary conditions and $i = N$, $j = 1$.

We differentiate Equation (6.7) with respect to $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{21} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{B.95}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j}} = -1 + 3\,(\phi_{i,j})^2 + 2\,(\epsilon)^2 \left[(\frac{1}{\Delta x^2}) + (\frac{1}{\Delta y^2})\right], \tag{B.96}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j+1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{B.97}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{B.98}$$

where $\phi_{i+1,j} = -1$ and $\mu_{i+1,j} = p_{i+1,j} = 0$from the Dirichlet boundary conditions and $i = N$, $j = 2, ..., M-1$.

We differentiate Equation (6.7) with respect to $\phi_{i-1,j}$, $\phi_{i,j}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{21} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{B.99}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j}} = -1 + 3\,(\phi_{i,j})^2 + 2\,(\epsilon)^2 \left[(\frac{1}{\Delta x^2}) + (\frac{1}{\Delta y^2})\right], \tag{B.100}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{B.101}$$

where $\phi_{i+1,j} = \phi_{i,j+1} = -1$, $\mu_{i+1,j} = p_{i+1,j} = 0$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = M$.

We differentiate Equation (6.7) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{21} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i+1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{B.102}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{B.103}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j}} = -1 + 3\,(\phi_{i,j})^2 + 2\,(\epsilon)^2 \left[(\frac{1}{\Delta x^2}) + (\frac{1}{\Delta y^2})\right], \tag{B.104}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{B.105}$$

where $\phi_{i,j+1} = -1$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = M$.

We differentiate Equation (6.7) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{21} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i+1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{B.106}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j}} = -1 + 3\,(\phi_{i,j})^2 + 2\,(\epsilon)^2 \left[(\frac{1}{\Delta x^2}) + (\frac{1}{\Delta y^2})\right], \tag{B.107}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{B.108}$$

where $\phi_{i-1,j} = \phi_{i,j+1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = M$.

We differentiate Equation (6.7) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{21} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i+1,j}} = -(\frac{(\epsilon)^2}{\Delta x^2}), \tag{B.109}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j}} = -1 + 3\,(\phi_{i,j})^2 + 2\,(\epsilon)^2 \left[(\frac{1}{\Delta x^2}) + (\frac{1}{\Delta y^2})\right], \tag{B.110}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j+1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{B.111}$$

$$\frac{\partial F_{\mu_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{(\epsilon)^2}{\Delta y^2}), \tag{B.112}$$

where $\phi_{i-1,j} = -1$ and $\mu_{i-1,j} = p_{i-1,j} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = 2, ..., M-1$.

We differentiate Equation (6.7) with respect to $\mu_{i+1,j}$, $\mu_{i,j}$ and $\mu_{i,j+1}$ to obtain the non-zero entries of J_{22} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \mu_{i,j}} = -1, \tag{B.113}$$

where $\phi_{i-1,j} = \phi_{i,j-1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = 1$.

We differentiate Equation (6.7) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$ and $\mu_{i,j+1}$ to obtain the

non-zero entries of J_{22} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \mu_{i,j}} = -1, \tag{B.114}$$

where $\phi_{i,j-1} = -1$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = 1$.

We differentiate Equation (6.7) with respect to $\mu_{i-1,j}$, $\mu_{i,j}$ and $\mu_{i,j+1}$ to obtain the non-zero entries of J_{22} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \mu_{i,j}} = -1, \tag{B.115}$$

where $\phi_{i+1,j} = \phi_{i,j-1} = -1$, $\mu_{i+1,j} = p_{i+1,j} = 0$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = 1$.

We differentiate Equation (6.7) with respect to $\mu_{i-1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{22} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \mu_{i,j}} = -1, \tag{B.116}$$

where $\phi_{i+1,j} = -1$ and $\mu_{i+1,j} = p_{i+1,j} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = 2, ..., M-1$.

We differentiate Equation (6.7) with respect to $\mu_{i-1,j}$, $\mu_{i,j}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{22} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \mu_{i,j}} = -1, \tag{B.117}$$

where $\phi_{i+1,j} = \phi_{i,j+1} = -1$, $\mu_{i+1,j} = p_{i+1,j} = 0$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = M$.

We differentiate Equation (6.7) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{22} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \mu_{i,j}} = -1, \tag{B.118}$$

where $\phi_{i,j+1} = -1$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = M$.

We differentiate Equation (6.7) with respect to $\mu_{i+1,j}$, $\mu_{i,j}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{22} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \mu_{i,j}} = -1, \tag{B.119}$$

where $\phi_{i-1,j} = \phi_{i,j+1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = M$.

We differentiate Equation (6.7) with respect to $\mu_{i+1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{22} as follows:

$$\frac{\partial F_{\mu_{i,j}}}{\partial \mu_{i,j}} = -1, \tag{B.120}$$

where $\phi_{i-1,j} = -1$ and $\mu_{i-1,j} = p_{i-1,j} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = 2, ..., M-1$.

B.3 F_p Equation

We differentiate Equation (6.8) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{31} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i+1,j}} = (\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i+1,j} - \mu_{i,j}), \tag{B.121}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j}} &= (\frac{\gamma}{\Delta x^2})(\frac{1}{2})\,[(\mu_{i+1,j} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i-1,j})] \\ &+(\frac{\gamma}{\Delta y^2})(\frac{1}{2})\,[(\mu_{i,j+1} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i,j-1})]\,,\end{aligned} \tag{B.122}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j+1}} = (\frac{\gamma}{\Delta y^2})\,(\frac{1}{2})\,(\mu_{i,j+1} - \mu_{i,j}), \tag{B.123}$$

where $i = 1$, $j = 1$.

We differentiate Equation (6.8) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{31} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i+1,j}} = (\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i+1,j} - \mu_{i,j}), \tag{B.124}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i,j} - \mu_{i-1,j}), \tag{B.125}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j}} &= (\frac{\gamma}{\Delta x^2})(\frac{1}{2})\,[(\mu_{i+1,j} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i-1,j})] \\ &+(\frac{\gamma}{\Delta y^2})(\frac{1}{2})\,[(\mu_{i,j+1} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i,j-1})]\,,\end{aligned} \tag{B.126}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j+1}} = (\frac{\gamma}{\Delta y^2})\,(\frac{1}{2})\,(\mu_{i,j+1} - \mu_{i,j}), \tag{B.127}$$

where $i = 2, ..., N-1$, $j = 1$.

We differentiate Equation (6.8) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{31} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i,j} - \mu_{i-1,j}), \tag{B.128}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j}} &= (\frac{\gamma}{\Delta x^2})(\frac{1}{2})\,[(\mu_{i+1,j} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i-1,j})] \\ &+(\frac{\gamma}{\Delta y^2})(\frac{1}{2})\,[(\mu_{i,j+1} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i,j-1})]\,,\end{aligned} \tag{B.129}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j+1}} = (\frac{\gamma}{\Delta y^2})\,(\frac{1}{2})\,(\mu_{i,j+1} - \mu_{i,j}), \tag{B.130}$$

where $i = N$, $j = 1$.

We differentiate Equation (6.8) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{31} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i,j} - \mu_{i-1,j}), \tag{B.131}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j}} &= (\frac{\gamma}{\Delta x^2})(\frac{1}{2})\,[(\mu_{i+1,j} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i-1,j})] \\ &+(\frac{\gamma}{\Delta y^2})(\frac{1}{2})\,[(\mu_{i,j+1} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i,j-1})]\,,\end{aligned} \tag{B.132}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j+1}} = (\frac{\gamma}{\Delta y^2})\,(\frac{1}{2})\,(\mu_{i,j+1} - \mu_{i,j}), \tag{B.133}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{\gamma}{\Delta y^2})\,(\frac{1}{2})\,(\mu_{i,j} - \mu_{i,j-1}), \tag{B.134}$$

where $i = N$, $j = 2, ..., M-1$.

We differentiate Equation (6.8) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{31} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i,j} - \mu_{i-1,j}), \tag{B.135}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j}} &= (\frac{\gamma}{\Delta x^2})(\frac{1}{2})\,[(\mu_{i+1,j} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i-1,j})] \\ &+(\frac{\gamma}{\Delta y^2})(\frac{1}{2})\,[(\mu_{i,j+1} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i,j-1})]\,,\end{aligned} \tag{B.136}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{\gamma}{\Delta y^2})\,(\frac{1}{2})\,(\mu_{i,j} - \mu_{i,j-1}), \tag{B.137}$$

where $i = N$, $j = M$.

We differentiate Equation (6.8) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{31} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i+1,j}} = (\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i+1,j} - \mu_{i,j}), \tag{B.138}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i-1,j}} = -(\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i,j} - \mu_{i-1,j}), \tag{B.139}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j}} = (\frac{\gamma}{\Delta x^2})(\frac{1}{2})\,[(\mu_{i+1,j} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i-1,j})]$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j+1}} = (\frac{\gamma}{\Delta y^2})\,(\frac{1}{2})\,(\mu_{i,j+1} - \mu_{i,j}), \tag{B.140}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{\gamma}{\Delta y^2})\,(\frac{1}{2})\,(\mu_{i,j} - \mu_{i,j-1}), \tag{B.141}$$

where $i = 2, ..., N-1$, $j = M$.

We differentiate Equation (6.8) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{31} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i+1,j}} = (\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i+1,j} - \mu_{i,j}), \tag{B.142}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j}} = (\frac{\gamma}{\Delta x^2})(\frac{1}{2})\,[(\mu_{i+1,j} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i-1,j})]$$

$$+(\frac{\gamma}{\Delta y^2})(\frac{1}{2})\,[(\mu_{i,j+1} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i,j-1})]\,, \tag{B.143}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{\gamma}{\Delta y^2})\,(\frac{1}{2})\,(\mu_{i,j} - \mu_{i,j-1}), \tag{B.144}$$

where $i = 1$, $j = M$.

We differentiate Equation (6.8) with respect to $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j}$, $\phi_{i,j+1}$ and $\phi_{i,j-1}$ to obtain the non-zero entries of J_{31} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i+1,j}} = (\frac{\gamma}{\Delta x^2})\,(\frac{1}{2})\,(\mu_{i+1,j} - \mu_{i,j}), \tag{B.145}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j}} = (\frac{\gamma}{\Delta x^2})(\frac{1}{2})\,[(\mu_{i+1,j} - \mu_{i,j}) - (\mu_{i,j} - \mu_{i-1,j})]$$

$$+(\frac{\gamma}{\Delta y^2})(\frac{1}{2})\left[(\mu_{i,j+1}-\mu_{i,j})-(\mu_{i,j}-\mu_{i,j-1})\right], \tag{B.146}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j+1}} = (\frac{\gamma}{\Delta y^2})(\frac{1}{2})(\mu_{i,j+1}-\mu_{i,j}), \tag{B.147}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \phi_{i,j-1}} = -(\frac{\gamma}{\Delta y^2})(\frac{1}{2})(\mu_{i,j}-\mu_{i,j-1}), \tag{B.148}$$

where $i = 1$, $j = 2, ..., M-1$.

We differentiate Equation (6.8) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{32} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{\gamma}{\Delta x^2})\frac{(\phi_{i+1,j}+\phi_{i,j})}{2}, \tag{B.149}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j}} &= -(\frac{\gamma}{\Delta x^2})\left[\frac{(\phi_{i+1,j}+\phi_{i,j})}{2}+\frac{(\phi_{i,j}+\phi_{i-1,j})}{2}\right] \\ &\quad -(\frac{\gamma}{\Delta y^2})\left[\frac{(\phi_{i,j+1}+\phi_{i,j})}{2}+\frac{(\phi_{i,j}+\phi_{i,j-1})}{2}\right],\end{aligned} \tag{B.150}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{\gamma}{\Delta y^2})\frac{(\phi_{i,j+1}+\phi_{i,j})}{2}, \tag{B.151}$$

where $i = 1$, $j = 1$.

We differentiate Equation (6.8) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{32} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{\gamma}{\Delta x^2})\frac{(\phi_{i+1,j}+\phi_{i,j})}{2}, \tag{B.152}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{\gamma}{\Delta x^2})\frac{(\phi_{i,j}+\phi_{i-1,j})}{2}, \tag{B.153}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j}} &= -(\frac{\gamma}{\Delta x^2})\left[\frac{(\phi_{i+1,j}+\phi_{i,j})}{2}+\frac{(\phi_{i,j}+\phi_{i-1,j})}{2}\right] \\ &\quad -(\frac{\gamma}{\Delta y^2})\left[\frac{(\phi_{i,j+1}+\phi_{i,j})}{2}+\frac{(\phi_{i,j}+\phi_{i,j-1})}{2}\right],\end{aligned} \tag{B.154}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{\gamma}{\Delta y^2})\frac{(\phi_{i,j+1}+\phi_{i,j})}{2}, \tag{B.155}$$

where $i = 2, ..., N-1$, $j = 1$.

We differentiate Equation (6.8) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain

the non-zero entries of J_{32} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{\gamma}{\Delta x^2}) \frac{(\phi_{i,j} + \phi_{i-1,j})}{2}, \tag{B.156}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j}} = &-(\frac{\gamma}{\Delta x^2}) \left[\frac{(\phi_{i+1,j} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i-1,j})}{2}\right] \\ &-(\frac{\gamma}{\Delta y^2}) \left[\frac{(\phi_{i,j+1} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i,j-1})}{2}\right],\end{aligned} \tag{B.157}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{\gamma}{\Delta y^2}) \frac{(\phi_{i,j+1} + \phi_{i,j})}{2}, \tag{B.158}$$

where $i = N$, $j = 1$.

We differentiate Equation (6.8) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{32} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{\gamma}{\Delta x^2}) \frac{(\phi_{i,j} + \phi_{i-1,j})}{2}, \tag{B.159}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j}} = &-(\frac{\gamma}{\Delta x^2}) \left[\frac{(\phi_{i+1,j} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i-1,j})}{2}\right] \\ &-(\frac{\gamma}{\Delta y^2}) \left[\frac{(\phi_{i,j+1} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i,j-1})}{2}\right],\end{aligned} \tag{B.160}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{\gamma}{\Delta y^2}) \frac{(\phi_{i,j+1} + \phi_{i,j})}{2}, \tag{B.161}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{\gamma}{\Delta y^2}) \frac{(\phi_{i,j} + \phi_{i,j-1})}{2}, \tag{B.162}$$

where $i = N$, $j = 2, ..., M-1$.

We differentiate Equation (6.8) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{32} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{\gamma}{\Delta x^2}) \frac{(\phi_{i,j} + \phi_{i-1,j})}{2}, \tag{B.163}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j}} = &-(\frac{\gamma}{\Delta x^2}) \left[\frac{(\phi_{i+1,j} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i-1,j})}{2}\right] \\ &-(\frac{\gamma}{\Delta y^2}) \left[\frac{(\phi_{i,j+1} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i,j-1})}{2}\right],\end{aligned} \tag{B.164}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{\gamma}{\Delta y^2}) \frac{(\phi_{i,j} + \phi_{i,j-1})}{2}, \tag{B.165}$$

where $i = N$, $j = M$.

We differentiate Equation (6.8) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{32} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{\gamma}{\Delta x^2}) \frac{(\phi_{i+1,j} + \phi_{i,j})}{2}, \tag{B.166}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i-1,j}} = (\frac{\gamma}{\Delta x^2}) \frac{(\phi_{i,j} + \phi_{i-1,j})}{2}, \tag{B.167}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j}} = &-(\frac{\gamma}{\Delta x^2}) \left[\frac{(\phi_{i+1,j} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i-1,j})}{2}\right] \\ &-(\frac{\gamma}{\Delta y^2}) \left[\frac{(\phi_{i,j+1} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i,j-1})}{2}\right],\end{aligned} \tag{B.168}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{\gamma}{\Delta y^2}) \frac{(\phi_{i,j} + \phi_{i,j-1})}{2}, \tag{B.169}$$

where $i = 2, ..., N-1$, $j = M$.

We differentiate Equation (6.8) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{32} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{\gamma}{\Delta x^2}) \frac{(\phi_{i+1,j} + \phi_{i,j})}{2}, \tag{B.170}$$

$$\begin{aligned}\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j}} = &-(\frac{\gamma}{\Delta x^2}) \left[\frac{(\phi_{i+1,j} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i-1,j})}{2}\right] \\ &-(\frac{\gamma}{\Delta y^2}) \left[\frac{(\phi_{i,j+1} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i,j-1})}{2}\right],\end{aligned} \tag{B.171}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{\gamma}{\Delta y^2}) \frac{(\phi_{i,j} + \phi_{i,j-1})}{2}, \tag{B.172}$$

where $i = 1$, $j = M$.

We differentiate Equation (6.8) with respect to $\mu_{i+1,j}$, $\mu_{i-1,j}$, $\mu_{i,j}$, $\mu_{i,j+1}$ and $\mu_{i,j-1}$ to obtain the non-zero entries of J_{32} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i+1,j}} = (\frac{\gamma}{\Delta x^2}) \frac{(\phi_{i+1,j} + \phi_{i,j})}{2}, \tag{B.173}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j}} = -(\frac{\gamma}{\Delta x^2}) \left[\frac{(\phi_{i+1,j} + \phi_{i,j})}{2} + \frac{(\phi_{i,j} + \phi_{i-1,j})}{2}\right]$$

$$-(\frac{\gamma}{\Delta y^2})\left[\frac{(\phi_{i,j+1}+\phi_{i,j})}{2}+\frac{(\phi_{i,j}+\phi_{i,j-1})}{2}\right], \tag{B.174}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j+1}} = (\frac{\gamma}{\Delta y^2})\frac{(\phi_{i,j+1}+\phi_{i,j})}{2}, \tag{B.175}$$

$$\frac{\partial F_{p_{i,j}}}{\partial \mu_{i,j-1}} = (\frac{\gamma}{\Delta y^2})\frac{(\phi_{i,j}+\phi_{i,j-1})}{2}, \tag{B.176}$$

where $i = 1$, $j = 2, ..., M-1$.
We differentiate Equation (6.8) with respect to $p_{i+1,j}$, $p_{i,j}$ and $p_{i,j+1}$ to obtain the non-zero entries of J_{33} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}), \tag{B.177}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j}} = -(2)\,(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}), \tag{B.178}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}), \tag{B.179}$$

where $\phi_{i-1,j} = \phi_{i,j-1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = 1$.

We differentiate Equation (6.8) with respect to $p_{i+1,j}$, $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j+1}$ to obtain the non-zero entries of J_{33} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}), \tag{B.180}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}), \tag{B.181}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j}} = -(2)\,(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}), \tag{B.182}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}), \tag{B.183}$$

where $\phi_{i,j-1} = -1$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = 1$.

We differentiate Equation (6.8) with respect to $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j+1}$ to obtain the non-zero entries of J_{33} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}), \tag{B.184}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j}} = -(2)\,(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}), \tag{B.185}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}), \tag{B.186}$$

where $\phi_{i+1,j} = \phi_{i,j-1} = -1$, $\mu_{i+1,j} = p_{i+1,j} = 0$ and $\mu_{i,j-1} = p_{i,j-1} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = 1$.

We differentiate Equation (6.8) with respect to $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j+1}$ to obtain the non-zero entries of J_{33} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}), \tag{B.187}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j}} = -(2)\,(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}), \tag{B.188}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}), \tag{B.189}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}), \tag{B.190}$$

where $\phi_{i+1,j} = -1$ and $\mu_{i+1,j} = p_{i+1,j} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = 2, ..., M-1$.

We differentiate Equation (6.8) with respect to $p_{i-1,j}$, $p_{i,j}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{33} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}), \tag{B.191}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j}} = -(2)\,(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}), \tag{B.192}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}), \tag{B.193}$$

where $\phi_{i+1,j} = \phi_{i,j+1} = -1$, $\mu_{i+1,j} = p_{i+1,j} = 0$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = N$, $j = M$.

We differentiate Equation (6.8) with respect to $p_{i+1,j}, p_{i-1,j}$, $p_{i,j}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{33} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}), \tag{B.194}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i-1,j}} = (\frac{1}{\Delta x^2}), \tag{B.195}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j}} = -(2)\,(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}), \tag{B.196}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}), \tag{B.197}$$

where $\phi_{i,j+1} = -1$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = 2, ..., N-1$, $j = M$.

We differentiate Equation (6.8) with respect to $p_{i+1,j}$, $p_{i,j}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{33} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}), \tag{B.198}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j}} = -(2)\,(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}), \tag{B.199}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}), \tag{B.200}$$

where $\phi_{i-1,j} = \phi_{i,j+1} = -1$, $\mu_{i-1,j} = p_{i-1,j} = 0$ and $\mu_{i,j+1} = p_{i,j+1} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = M$.

We differentiate Equation (6.8) with respect to $p_{i+1,j}$, $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j-1}$ to obtain the non-zero entries of J_{33} as follows:

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i+1,j}} = (\frac{1}{\Delta x^2}), \tag{B.201}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j}} = -(2)\,(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}), \tag{B.202}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j+1}} = (\frac{1}{\Delta y^2}), \tag{B.203}$$

$$\frac{\partial F_{p_{i,j}}}{\partial p_{i,j-1}} = (\frac{1}{\Delta y^2}), \tag{B.204}$$

where $\phi_{i-1,j} = -1$ and $\mu_{i-1,j} = p_{i-1,j} = 0$ from the Dirichlet boundary conditions and $i = 1$, $j = 2, ..., M-1$.

Bibliography

[1] A. H. S. Alrehaili. *Efficient Iterative Solution Algorithms For Numerical Models of Multiphase Flow*. PhD thesis, University of Leeds, 2018.

[2] W. F. Ames. *Numerical methods for partial differential equations*. Academic Press, 1977.

[3] P. Amestoy, I. Duff, and J. LExcellent. MUMPS multifrontal massively parallel solver. Technical report, Tech. rep., Version 2.0. Technical Report TR/PA/98/02. CERFACS, 42 Ave G. Coriolis, 31057 Toulouse Cedex, France, 1998.

[4] D. M. Anderson, G. B. McFadden, and A. A. Wheeler. Diffuse-interface methods in fluid mechanics. *Annual Review of Fluid Mechanics*, 30(1):139–165, 1998.

[5] U. M. Ascher and L. R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. SIAM, 1998.

[6] K. Atkinson, W. Han, and D. E. Stewart. *Numerical solution of ordinary differential equations*, volume 108. John Wiley & Sons, 2011.

[7] R. E. Bank and D. J. Rose. Global approximate Newton methods. *Numerische Mathematik*, 37(2):279–295, 1981.

[8] 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*, volume 43. SIAM, 1994.

[9] S. Bellavia, D. Bertaccini, and B. Morini. Nonsymmetric preconditioner updates in Newton–Krylov methods for nonlinear systems. *SIAM Journal on Scientific Computing*, 33(5):2595–2619, 2011.

[10] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[11] M. S. Berger. *Nonlinearity and functional analysis: lectures on nonlinear problems in mathematical analysis*, volume 74. Academic Press, 1977.

[12] A. L. Bertozzi. The mathematics of moving contact lines in thin liquid films. *Notices of the AMS*, 45(6):689–697, 1998.

[13] A. L. Bertozzi and M. P. Brenner. Linear stability and transient growth in driven contact lines. *Physics of Fluids*, 9(3):530–539, 1997.

[14] L. Blackford, J. Choi, A. Cleary, E. DAzevedo, J. Jemmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. Whaley. Scalapack: A linear algebra library for message-passing computers. In *PPSC*. SIAM, 1997.

[15] J. Boyle, M. Mihajlovic, and J. Scott. HSL MI20: an efficient AMG preconditioner. Technical report, Department of Computational and Applied Mathematics, Rutherford Appleton Laboratory, 2007.

[16] K. Brabazon, M. Hubbard, and P. Jimack. Nonlinear multigrid methods for second order differential operators with nonlinear diffusion coefficient. *Computers & Mathematics with Applications*, 68(12):1619–1634, 2014.

[17] K. J. Brabazon. *Multigrid methods for nonlinear second order partial differential operators*. PhD thesis, University of Leeds, 2014.

[18] D. Braess. *Finite elements: Theory, fast solvers, and applications in solid mechanics*. Cambridge University Press, 2007.

[19] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390, 1977.

[20] A. Brandt. Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(1-4):23–56, 1986.

[21] A. Brandt, S. McCormick, and J. Ruge. *Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations*. Report, Inst. for Computational Studies, Fort Collins, Colorado, 1983.

[22] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. *Sparsity and its Applications*, 257, 1985.

[23] R. J. Braun. Dynamics of the tear film. *Annual Review of Fluid Mechanics*, 44:267–297, 2012.

[24] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical solution of initial-value problems in differential-algebraic equations*, volume 14. SIAM, 1996.

[25] C. Breward, H. Byrne, and C. Lewis. The role of cell-cell interactions in a two-phase model for avascular tumour growth. *Journal of Mathematical Biology*, 45(2):125–152, 2002.

[26] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge. Algebraic multigrid based on element interpolation (AMGe). *SIAM Journal on Scientific Computing*, 22(5):1570–1592, 2001.

[27] M. Brezina, A. Doostan, T. Manteuffel, S. McCormick, and J. Ruge. Smoothed aggregation algebraic multigrid for stochastic PDE problems with layered materials. *Numerical Linear Algebra with Applications*, 21(2):239–255, 2014.

[28] W. L. Briggs, S. F. McCormick, et al. *A multigrid tutorial*, volume 72. SIAM, 2000.

[29] J. Brown, B. Smith, and A. Ahmadia. Achieving textbook multigrid efficiency for hydrostatic ice sheet flow. *SIAM Journal on Scientific Computing*, 35(2):B359–B375, 2013.

[30] P. N. Brown and Y. Saad. Convergence theory of nonlinear Newton–Krylov algorithms. *SIAM Journal on Optimization*, 4(2):297–330, 1994.

[31] J. L. Bull, C. A. Reickert, S. Tredici, E. Komori, E. L. Frank, D. O. Brant, J. B. Grotberg, and R. B. Hirschl. Flow limitation in liquid-filled lungs: Effects of liquid properties. *Journal of Biomechanical Engineering*, 127(4):630–636, 2005.

[32] R. L. Burden and D. J. Faires. *Numerical Analysis*. PWS Publishing Co., Boston, MA, USA, 1989.

[33] J. C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[34] G. Caginalp. Stefan and Hele-Shaw type models as asymptotic limits of the phase-field equations. *Physical Review A*, 39(11):5887, 1989.

[35] J. W. Cahn. On spinodal decomposition. *Acta Metallurgica*, 9(9):795–801, 1961.

[36] J. W. Cahn and J. E. Hilliard. Free energy of a nonuniform system. i. interfacial free energy. *The Journal of Chemical Physics*, 28(2):258–267, 1958.

[37] C. Canuto, A. Quarteroni, M. Y. Hussaini, and T. A. Zang. *Fundamentals of Fluid Dynamics*. Springer, 2007.

[38] L.-Q. Chen. Phase-field models for microstructure evolution. *Annual Review of Materials Research*, 32(1):113–140, 2002.

[39] J. Cui. Multigrid methods for two-dimensional Maxwell's equations on graded meshes. *Journal of Computational and Applied Mathematics*, 255:231–247, 2014.

[40] P.-G. De Gennes. Wetting: statics and dynamics. *Reviews of Modern Physics*, 57(3):827, 1985.

[41] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[42] P. Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*, volume 35. Springer Science & Business Media, 2011.

[43] P. Deuflhard. A short history of Newton's method. *Documenta Mathematica, Optimization stories*, pages 25–30, 2012.

[44] F. Dobrian and A. Pothen. Oblio: a sparse direct solver library for serial and parallel computations. Technical report, Old Dominion University, 2000.

[45] P. DuChateau and D. Zachmann. *Applied partial differential equations*. Courier Corporation, 2012.

[46] S. C. Eisenstat and H. F. Walker. Globally convergent inexact Newton methods. *SIAM Journal on Optimization*, 4(2):393–422, 1994.

[47] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics & Scientific Computation, 2014.

[48] Y. A. Erlangga and R. Nabben. Algebraic multilevel Krylov methods. *SIAM Journal on Scientific Computing*, 31(5):3417–3437, 2009.

[49] R. W. Freund, G. H. Golub, and N. M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1992.

[50] H. B. Frieboes, F. Jin, Y.-L. Chuang, S. M. Wise, J. S. Lowengrub, and V. Cristini. Three-dimensional multispecies nonlinear tumor growth II: tumor invasion and angiogenesis. *Journal of Theoretical Biology*, 264(4):1254–1278, 2010.

[51] P. Gaskell, P. Jimack, M. Sellier, and H. Thompson. Efficient and accurate time adaptive multigrid simulations of droplet spreading. *International Journal for Numerical Methods in Fluids*, 45(11):1161–1186, 2004.

[52] P. Gaskell, P. Jimack, M. Sellier, and H. Thompson. Flow of evaporating, gravity-driven thin liquid films over topography. *Physics of Fluids*, 18(1):013601, 2006.

[53] P. Gaskell, P. Jimack, M. Sellier, H. Thompson, and M. Wilson. Gravity-driven flow of continuous thin liquid films on non-porous substrates with topography. *Journal of Fluid Mechanics*, 509:253–280, 2004.

[54] A. Gillman. *Fast direct solvers for elliptic partial differential equations*. PhD thesis, University of Colorado, 2011.

[55] G. H. Golub and C. F. Van Loan. *Matrix computations. 4rd eds.* John Hopkins University Press, 2012.

[56] S. Gottlieb, C.-W. Shu, and E. Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Review*, 43(1):89–112, 2001.

[57] A. Greenbaum, V. Pták, and Z. E. K. St rakoš. Any nonincreasing convergence curve is possible for GMRES. *SIAM Journal on Matrix Analysis and Applications*, 17(3):465–469, 1996.

[58] D. F. Griffiths and D. J. Higham. *Numerical Methods for Ordinary Differential Equations: Initial Value Problems.* Springer Science & Business Media, 2010.

[59] G. Grinstein and G. Mazenko. *Directions in condensed matter physics: memorial volume in honor of Shang-keng Ma*, volume 1. World Scientific, 1986.

[60] L. Guo, H. Huang, D. R. Gaston, C. J. Permann, D. Andrs, G. D. Redden, C. Lu, D. T. Fox, and Y. Fujita. A parallel, fully coupled, fully implicit solution to reactive transport in porous media using the preconditioned Jacobian-free Newton-Krylov method. *Advances in water resources*, 53:101–108, 2013.

[61] W. Hackbusch. *Iterative solution of large sparse systems of equations*, volume 40. Springer, 1994.

[62] W. Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

[63] E. Hairer, S. Norsett, and G. Wanner. Solving ordinary differential equations I: Nonstiff problems. *SIAM Review*, 32(3):485–486, 1990.

[64] D. Halpern, O. Jensen, and J. Grotberg. A theoretical study of surfactant and liquid delivery into the lung. *Journal of Applied Physiology*, 85(1):333–352, 1998.

[65] M. T. Heath. *Scientific Computing: An Introductory Survey.* McGraw-Hill, 1997.

[66] M. Heil. An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 193(1-2):1–23, 2004.

[67] D. Hermann, M. Frank, K. Busch, and P. Wolfle. Photonic band structure computations. *Optics Express*, 8(3):167–172, 2001.

[68] E. Hermans, M. S. Bhamla, P. Kao, G. G. Fuller, and J. Vermant. Lung surfactants and different contributions to thin film stability. *Soft Matter*, 11(41):8048–8057, 2015.

[69] M. R. Hestenes and E. Stiefel. *Methods of conjugate gradients for solving linear systems*, volume 49. Journal of the National Bureau Standards Washington, DC, 1952.

[70] HSL. A collection of fortran codes for large-scale scientific computation. *See http://www.hsl.rl.ac.uk*, 2007.

[71] W. Hundsdorfer and J. G. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*, volume 33. Springer Science & Business Media, 2013.

[72] A. Iserles. *A first course in the numerical analysis of differential equations*. Number 44. Cambridge University Press, 2009.

[73] G. Jouvet and C. Gräser. An adaptive Newton multigrid method for a model of marine ice sheets. *Journal of Computational Physics*, 252:419–437, 2013.

[74] G. Juncu, A. Nicola, and C. Popa. Nonlinear multigrid methods for numerical solution of the variably saturated flow equation in two space dimensions. *Transport in Porous Media*, 91(1):35–47, 2012.

[75] D. Juric and G. Tryggvason. A front-tracking method for dendritic solidification. *Journal of Computational Physics*, 123(1):127–148, 1996.

[76] S. Kalliadasis, C. Bielarz, and G. Homsy. Steady free-surface thin film flows over topography. *Physics of Fluids*, 12(8):1889–1898, 2000.

[77] C. T. Kelley. Iterative methods for linear and nonlinear equations. *Raleigh N. C.: North Carolina State University*, 1995.

[78] C. T. Kelley. *Solving nonlinear equations with Newton's method*, volume 1. SIAM, 2003.

[79] D. Kelly, D. S. Gago, O. Zienkiewicz, and I. Babuska. A posteriori error analysis and adaptive processes in the finite element method: Part I—error analysis. *International Journal for Numerical Methods in Engineering*, 19(11):1593–1619, 1983.

[80] Y.-T. Kim, N. Goldenfeld, and J. Dantzig. Computation of dendritic microstructures using a level set method. *Physical Review E*, 62(2):2471, 2000.

[81] P. Knabner and L. Angermann. Numerical methods for elliptic and parabolic partial differential equations. Translation from the german. *Texts in Applied Mathematics*, 44, 2003.

[82] D. A. Knoll and D. E. Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[83] D. A. Knoll and W. Rider. A multigrid preconditioned Newton–Krylov method. *SIAM Journal on Scientific Computing*, 21(2):691–710, 1999.

[84] R. Kobayashi. Modeling and numerical simulations of dendritic crystal growth. *Physica D: Nonlinear Phenomena*, 63(3-4):410–423, 1993.

[85] J. D. Lambert. *Numerical methods for ordinary differential systems: the initial value problem*. Wiley Chichester, 1991.

[86] C. Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, 1950.

[87] L. Lapidus and G. F. Pinder. *Numerical solution of partial differential equations in science and engineering*. John Wiley & Sons, 2011.

[88] H.-G. Lee, J. Lowengrub, and J. Goodman. Modeling pinchoff and reconnection in a Hele-Shaw cell. I. the models and their calibration. *Physics of Fluids*, 14(2):492–513, 2002.

[89] H.-G. Lee, J. Lowengrub, and J. Goodman. Modeling pinchoff and reconnection in a Hele-Shaw cell. II. analysis and simulation in the nonlinear regime. *Physics of Fluids*, 14(2):514–545, 2002.

[90] R. J. LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge University Press, 2002.

[91] X. S. Li, J. W. Demmel, J. R. Gilbert, L. Grigori, M. Shao, and I. Yamazaki. SuperLU users guide. *Lawrence Berkeley National Laboratory Tech. Report, LBNL-44289*, 1999.

[92] P. T. Lin. Improving multigrid performance for unstructured mesh drift–diffusion simulations on 147,000 cores. *International Journal for Numerical Methods in Engineering*, 91(9):971–989, 2012.

[93] S. P. MacLachlan and L. N. Olson. Theoretical bounds for algebraic multigrid performance: review and analysis. *Numerical Linear Algebra with Applications*, 21(2):194–220, 2014.

[94] C. Marchi, L. Araki, A. Alves, R. Suero, S. Gonçalves, and M. Pinto. Repeated Richardson extrapolation applied to the two-dimensional Laplace equation using triangular and square grids. *Applied Mathematical Modelling*, 37(7):4661–4675, 2013.

[95] S. F. McCormick. *Multigrid methods*. SIAM, 1987.

[96] C. T. Miller, C. N. Dawson, M. W. Farthing, T. Y. Hou, J. Huang, C. E. Kees, C. Kelley, and H. P. Langtangen. Numerical simulation of water resources problems: Models, methods, and trends. *Advances in Water Resources*, 51:405–437, 2013.

[97] M. F. Murphy, G. H. Golub, and A. J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6):1969–1972, 2000.

[98] S. G. Nash. Newton-type minimization via the Lanczos method. *SIAM Journal on Numerical Analysis*, 21(4):770–788, 1984.

[99] S. Orchard. On surface levelling in viscous liquids and gels. *Applied Scientific Research, Section A*, 11(4-6):451–464, 1963.

[100] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*, volume 30. SIAM, 1970.

[101] F. H. Pereira, S. L. L. Verardi, and S. I. Nabeta. A fast algebraic multigrid preconditioned conjugate gradient solver. *Applied Mathematics and Computation*, 179(1):344–351, 2006.

[102] M. Pernice and H. F. Walker. NITSOL: A Newton iterative solver for nonlinear systems. *SIAM Journal on Scientific Computing*, 19(1):302–318, 1998.

[103] L. M. Peurrung and D. B. Graves. Spin coating over topography. *IEEE Transactions on Semiconductor Manufacturing*, 6(1):72–76, 1993.

[104] Y. Pinchover and J. Rubinstein. *An introduction to partial differential equations*. Cambridge University Press, 2005.

[105] M. A. Pinsky. *Partial differential equations and boundary-value problems with applications*, volume 15. American Mathematical Soc., 2011.

[106] S. S. Rao. *The finite element method in engineering*. Elsevier, 2010.

[107] J. N. Reddy. *An introduction to the finite element method*, volume 2. McGraw-Hill New York, 1993.

[108] K. Rektorys. *Solving ordinary and partial boundary value problems in science and engineering*, volume 3. CRC Press, 1998.

[109] J. Rosam, P. Jimack, and A. Mullis. An adaptive, fully implicit multigrid phase-field model for the quantitative simulation of non-isothermal binary alloy solidification. *Acta Materialia*, 56(17):4559–4569, 2008.

[110] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of Computation*, 37(155):105–126, 1981.

[111] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[112] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[113] Y. Saad and H. A. Van Der Vorst. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1):1–33, 2000.

[114] L. Schwartz and D. Weidner. Modeling of coating flows on curved surfaces. *Journal of Engineering Mathematics*, 29(1):91–103, 1995.

[115] M. Sellier. *The numerical simulation of thin film flow over heterogeneous substrates*. PhD thesis, University of Leeds, 2003.

[116] V. P. Sergiievskyi, W. Hackbusch, and M. V. Fedorov. Multigrid solver for the reference interaction site model of molecular liquids theory. *Journal of Computational Chemistry*, 32(9):1982–1992, 2011.

[117] G. Sewell. *The numerical solution of ordinary and partial differential equations*, volume 75. John Wiley & Sons, 2005.

[118] L. F. Shampine. *Numerical solution of ordinary differential equations*, volume 4. CRC Press, 1994.

[119] J. R. Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994. Carnegie-Mellon University. Department of Computer Science.

[120] A. Shinozaki and Y. Oono. Spinodal decomposition in a Hele-Shaw cell. *Physical Review A*, 45(4):R2161, 1992.

[121] G. D. Smith. *Numerical solution of partial differential equations: finite difference methods*. Oxford University Press, 1985.

[122] R. V. Southwell. *Relaxation Methods in Theoretical Physics: A Continuation of the Treatise, Relaxation Methods in Engineering Science*, volume 2. The Clarendon Press, 1946.

[123] I. P. Stanimirovic and M. B. Tasic. Performance comparison of storage formats for sparse matrices. *Ser. Mathematics and Informatics*, 24(1):39–51, 2009.

[124] G. Strang and G. J. Fix. *An analysis of the finite element method*, volume 212. Prentice-Hall Englewood Cliffs, NJ, 1973.

[125] W. A. Strauss. Partial differential equations: An introduction. *New York*, 1992.

[126] K. Stüben. Algebraic multigrid (amg): experiences and comparisons. *Applied Mathematics and Computation*, 13(3-4):419–451, 1983.

[127] K. Stuben. Algebraic multigrid (AMG): an introduction with applications. *Multigrid*, 2000.

[128] K. Stuben. An introduction to algebraic multigrid. *Multigrid*, pages 413–532, 2001.

[129] E. Süli and D. F. Mayers. *An introduction to numerical analysis*. Cambridge University Press, 2003.

[130] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, Orlando, 2001.

[131] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996.

[132] P. S. Vassilevski and U. M. Yang. Reducing communication in algebraic multigrid using additive variants. *Numerical Linear Algebra with Applications*, 21(2):275–296, 2014.

[133] H. Vesteeg and W. Malalasekera. An introduction to computational fluid dynamics. *Essex: Longman Scientific & Technical*, 1995.

[134] C. Wang and P. Cheng. A multiphase mixture model for multiphase, multicomponent transport in capillary porous media I. model development. *International Journal of Heat and Mass Transfer*, 39(17):3607–3618, 1996.

[135] A. Wathen. Preconditioning and fast solvers for incompressible flow. *Technical report, Oxford University Computing Laboratory*, 2004.

[136] A. J. Wathen. Preconditioning. *Acta Numerica*, 24:329–376, 2015.

[137] P. Wesseling. *Introduction To Multigrid Methods.* DTIC Document, 1995.

[138] P. Wesseling and C. W. Oosterlee. Geometric multigrid with applications to computational fluid dynamics. *Journal of Computational and Applied Mathematics*, 128(1):311–334, 2001.

[139] S. Wise. Unconditionally stable finite difference, nonlinear multigrid simulation of the Cahn-Hilliard-Hele-Shaw system of equations. *Journal of Scientific Computing*, 44(1):38–68, 2010.

[140] S. M. Wise, J. S. Lowengrub, and V. Cristini. An adaptive multigrid algorithm for simulating solid tumor growth using mixture models. *Mathematical and Computer Modelling*, 53(1-2):1–20, 2011.

[141] S. M. Wise, J. S. Lowengrub, H. B. Frieboes, and V. Cristini. Three-dimensional multispecies nonlinear tumor growth I: model and numerical method. *Journal of Theoretical Biology*, 253(3):524–543, 2008.

[142] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34(4):581–613, 1992.

[143] J. Xu. The single-grid multilevel method and its applications. *Inverse Problems & Imaging*, 7(3), 2013.

[144] W. Xu and T. F. Coleman. Solving nonlinear equations with the Newton–Krylov method based on automatic differentiation. *Optimization Methods and Software*, 29(1):88–101, 2014.

[145] F. W. Yang. *Multigrid solution methods for nonlinear time-dependent systems.* PhD thesis, University of Leeds, 2014.

[146] O. Zeinkiewicz, R. Taylor, and J. Zhu. *The finite element method: its basis and fundamentals.* Elsevier Butterworth-Heinemann, 2005.