Efficient Iterative Solution Algorithms For Numerical Models of Multiphase Flow

by

Ahlam Hamdan S Alrehaili

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

UNIVERSITY OF LEEDS

The University of Leeds
School of Computing
April 2018

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 foremost, thank you Allah (Almighty God) for giving me the strength, knowledge, well-being and ability to undertake some of the most important work and experience in my life.

I dedicate this thesis to my parents Mr Hamdan and Mrs Mutyiah, whose dreams for me have resulted in this achievement and with for their love, support, motivation, prayers and care. They have always supported my education and have been a great motivation to accomplish this degree. I would also thank them for their patience and wait for me until I finish my PhD when I was studying abroad.

I am forever indebted to my supervisors, Dr. Mark Walkley and Prof. Peter Jimack, for their invaluable support, patience, motivation, feedback and encouraging comments throughout the duration of my PhD. I would also thank them for generously sharing their time and knowledges and also they have worked to instil great confidence in both myself and my work.

Very special thanks to Dr. Matthew Hubbard for sharing his source codes, advices, and suggestions during the planning and development of this research work.

To my husband, Sameer and my children Mayar, Mohammed and Muadh for their love, patience and support during my study. To my brothers and sisters in Saudi Arabia; Nabeel, Majed, Majdi, Basma, Reem, Shaima, Abdulmajeed, Bushra and Mashael, who were always beside me to support me. I appreciate all that you did for me.

Thanks also go to my friends in School of Computing; Amirah Alharbi, Areej Alshutayri, Entisar Abolkasim, Luluh Aldhubayi, and Mashael Aljohany.

I owe special thanks to Taibah University, Medina, for granting me a scholarship to pursue my studies in the UK.

Finally, my special thanks go to people in the School of Computing, I have always been proud of being one of its members.

Abstract

This thesis is concerned with the development and application of optimally efficient numerical methods for the simulation of vascular tumour growth, based upon the multiphase fluid model introduced by Hubbard and Byrne [57]. This multiphase model involves the flow and interaction of four different, but coupled, phases which are each treated as incompressible fluids.

Following a short review of models for tumour growth we describe in detail the model of Hubbard and Byrne [57], and introduce the discretization schemes used. This involves a finite volume scheme to approximate mass conservation and conforming finite element schemes to approximate momentum conservation and a reaction-diffusion equation for the background nutrient concentration. The momentum conservation system is represented as a Stokes-like flow of each phase, with source terms that reflect the phase interactions. It will be demonstrated that the solution of these coupled momentum equations, approximated using a Taylor-Hood finite element method in two dimensions, is the most computationally intensive component of the solution algorithm. The nonlinear system arising from the nutrient equation is the second most computationally expensive component.

The solvers presented in this work for the discretized systems are based on preconditioned Krylov methods. Algebraic multigrid (AMG) preconditioner and a novel block preconditioner are used with Krylov methods for solving the linear systems arising from the nutrient equation at each Newton step and from the momentum equation, respectively. In each case these are shown to be very efficient algorithms: when the preconditioning strategies are applied to practical problems, the CPU time and memory are demonstrated to scale almost linearly with the problem size.

Finally, the basic multiphase tumour model is extended to consider drug delivery and the inclusion of additional phases. To solve this extended model our preconditioning strategy is extended to cases with more than four phases. This is again demonstrated to perform optimally.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

General Introduction

This chapter is divided to three sections. Some background to the thesis is introduced in Section 1.1. This is followed by Section 1.2, which summarizes the main achievements and contributions of the thesis. Finally, the overview of the whole thesis is presented in Section 1.3.

1.1 Subject of the Thesis

Multiphase flows consist of more than one phase, or component: i.e mixtures of liquids, gases, and/or solids. There are many examples of multiphase flows in our environment such as rain drops or fog in air, oil extraction through a porous medium [51] and others. Also, many biological and medical flows are multiphase, such as blood flow with bubbles or solid suspensions, for example [25].

In this thesis, we focus primarily on the study of a particular multiphase flow model for tumour growth. Most of the studies in this area have used between two and four phases in 2D and 3D, often treating all of the phases under consideration as fluids (e.g [57]), or mixtures between fluids and solids (e.g [99]). As with any other flow or mechanics problems, there are many different ways to model multiphase flow [25]. It may be modelled experimentally in the laboratory by using appropriate instrumentation. Also, modelling theoretically (mathematically), using mathematical equations and simplifying them to produce exact or asymptotic solutions. Thirdly, modelling computationally (numerically) by using computers to address the complexity of the flow by approximating the solution of the governing mathematical models. Our study focuses on computational models which are based on continuous mathematical models.

The most common mathematical formulations of multiphase flow require a system of partial differential equations (PDEs), which are generally both time dependent and nonlinear. There

are several formulations of governing conservation equations for multiphase flow, depending on the application field (see for example in porous media [111] or in tumour growth [26, 57]). In multiphase flow models, effective conservation of mass and momentum are developed for each phase (without mixing phases). These include the terms of interaction, modelling the exchange of mass and momentum between the phases. The conservation of mass is often referred to as the continuity equation. Each of these phases is considered to have a separate volume fraction and velocity, and the sum of theses volume fractions is unity [25, 57].

The mathematical model that we study in this thesis is presented by Hubbard and Byrne [57], and extends the work of Breward, Byrne and Lewis [26,27]. It is a multiphase model of vascular tumour growth in two dimensions and includes four phases: healthy cells, tumour cells, blood vessels and extracellular material, with the assumption that each phase behaves like a viscous fluid.

During the last few decades, research related to tumour growth has increased dramatically and mathematical tumour modelling has made a significant contribution to this. These mathematical models complement biological experiments and clinical observations, and simulations of tumours can help to support the hypotheses taken from such observations. The review paper [31] contains a number of examples to illustrate how theory can drive experiments and vice versa.

Generally, tumour growth is divided into three major stages:

- Avascular growth, the tumour in this stage does not get any blood supply and has small size with a clustered and structured shape. The nutrient that feeds the tumour arrives by the mechanism of diffusion.

- Angiogenesis, which is a biological process by which new blood vessels are formed, is the next stage. Moreover, the tumour still has the structured shape associated with avascular growth.

- Vascular growth is the final stage, where the tumour has access to plentiful resources (blood supply), and gets a nutrient supply from surrounding tissue. This stage can be harmful and dangerous: rapid growth occurs in the mass of the tumour with unstructured shape, cells are able to exit through the walls of tumour vessel, these may then transmit through the blood stream and seed secondary tumours in other areas [27].

Most mathematical models focus on one particular aspect, such as avascular tumour growth (e.g. [26], [112] [107]), which is easier to validate against experiment, or vascular tumour growth (e.g. [27, 57, 83]). This project is focused upon the final stage of tumour growth, vascular tumours, which are generally the most dangerous. In particular, in this work we consider efficient numerical methods for the simulation of vascular tumour growth based upon the multiphase model introduced by Hubbard and Byrne [57].

The computational approach, which is used in this work, and in [57], is a combination of the finite element method (FEM) and the finite volume method (FVM). In this two dimensional

model, the hyperbolic PDEs of mass balance are discretised through an explicit, conservative, upwind, FVM scheme; whereas the momentum balance equations lead to generalised Stokes equations which are solved using a FEM scheme. Additionally, the discretization of the nutrient reaction-diffusion equation is also based upon a FEM combined with a Newton iteration to solve the resulting nonlinear algebraic equations.

In summary, the aim of this work is to develop efficient computational algorithms for the numerical approximation of PDE systems describing multiphase flow. In particular we set out to improve the efficiency of the numerical methods used, in order to reach optimal efficiency in this case linear time complexity and linear growth in memory with the problem size.

1.2 Achievements

In this research we used an optimal Newton-Krylov algorithm to efficiently solve the discrete algebraic system arising from finite element approximation of the reaction-diffusion equation for nutrient concentration. This algorithm uses the Newton method to linearise the nonlinear equations then solves the resulting linear systems with preconditioned GMRES. This combines sparse matrix techniques to minimise storage and algebraic multigrid (AMG) preconditioning for the Krylov method to minimise computational work. Secondly, we developed a novel preconditioner for the discrete form of the momentum balance equations. This is demonstrated to solve these finite element equations almost optimally. The third contribution of this research is solving a generalized model of tumour growth, with more phases, than in [57]. Discussion with the author suggests that up to 10 phases is a desirable target but this is currently limited by the efficiency of the existing algorithm [55]. We are able to demonstrate that models with greater number of phases are feasible provided efficient numerical methods are utilized.

The achievements can be summarised as:

- Optimal iterative solver for the nonlinear diffusion component of the solver.
- Almost optimal iterative solver for the momentum equations, based upon the development of a new preconditioner
- Testing on a range of regular and unstructured grids.
- Modification of the existing model in [57] by considering two extensions:
  - the model is extended to include drug delivery and the diffusion of this drug.
  - the mathematical model is extended to simulate five phases
- Generalization of our new preconditioner for solving the new five-phase model.

1.3 Overview of Thesis

This thesis contains nine chapters. From Chapter 2 to Chapter 4 selected background and review material from related work are shown. From Chapter 5 to Chapter 8 our numerical methods and the contributions described above are presented and assessed. The conclusions are discussed in the final chapter.

Chapter 2 begins with a review of several alternative tumour growth models, which are: continuum models, discrete models, and hybrid models. Then the specific mathematical model that has been chosen for our work is described in detail (see Section 2.2).

In Chapter 3, some relevant techniques in scientific computing are described. The chapter begins with a discussion of the main discretization techniques for approximating PDEs. The primary focus is on the finite element method (FEM) and finite volume method (FVM) due to the mixed FVM and FEM approach that is used in this work (see Section 3.1). In addition, solution methods for linear and nonlinear equation systems are discussed (see Section 3.2 and Section 3.4). Also, the idea of how the iterative solution of these discretised systems can be robust, efficient and optimal, through the use of preconditioning techniques, is presented.

Chapter 4 reviews some existing numerical models for tumour growth, presented in Section 4.1. The specific numerical schemes for the mathematical model which is discussed in Chapter 2 Section 2.2 are then introduced in Section 4.2. An introduction to some numerical experiments for the whole model, including the parameters values used, the initial and boundary conditions selected, and other key implementation details are presented in section 4.3.

Chapter 5 presents our results on the efficient solution of one component of the mathematical model in [57], which is the nonlinear system representing the reaction-diffusion equation (see Section 5.1). Furthermore, a comparison of results between the sparse direct solver, which is used in [57], and the proposed method for this work, which is based on an AMG preconditioned Newton-Krylov method, is presented (see section 5.2).

Chapter 6 focuses on the linear system representing the discretized momentum equations from the mathematical model. In this chapter a new block preconditioner is developed and used for solving the algebraic linear system resulting from the momentum balance equations approximated using the Taylor-Hood FEM. In Section 6.1 the structure of the discrete system and the solution of the algebraic linear system are introduced. Section 6.2 then describes the steps required to develop a new preconditioner. Then our numerical results are presented in Section 6.3.

In Chapter 7, the solution of the complete multiphase tumour model, incorporating each of our algorithmic improvements is considered. In this chapter, validation of our results against existing results in [57] is presented (see Section 7.1). Moreover, additional numerical simulations, with description of our results are shown in Section 7.2. Furthermore, the accuracy of our solutions is discussed in Section 7.3, where it is demonstrated that the additional mesh resolution that is permitted by the efficiency of our numerical solvers does indeed lead to improved accuracy. Finally, the computational time and the memory requirement for solving the whole

model are presented in Section 7.4.

In Chapter 8 the mathematical model in [57] is modified by adding a new governing equation representing the delivery and diffusion of a drug. Then, the model is extended further by increasing the number of phases from four to five. Section 8.1 presents how we model the drug diffusion, and illustrate the impact of the drug on the tumour growth. Then, the modification of the multiphase model to include an additional phase is presented in Section 8.2. The main objective in Chapter 8 is to extend our momentum equation preconditioner to a problem with a greater number of phases. This is achieved, along with a demonstration of the optimal efficiency when applied to this five phase system.

The conclusions are summarised in Chapter 9, in addition to a brief discussion of some possible further extensions.

Chapter 2

Review of Mathematical Modelling of Tumour Growth

A tumour is defined as a cluster of abnormal cell growth that arises in healthy tissue. It may be benign (not cancer) or malignant (cancer). Treatment of the tumour depends on the type and the behaviour of the tumour cells growth. One of the powerful tools to help predict and simulate the behaviour of the tumour progression is mathematical modelling. Research into mathematical modelling of tumour growth has increased dramatically in the last few decades to help understand the mechanisms of tumour growth [10].

This chapter is organised as follows: Section 2.1 gives an introduction to, and short review of, several alternative tumour growth models. Section 2.2, then goes into much greater detail of the specific mathematical model that has been chosen for this work.

2.1 Models of Tumour Growth

In general, three main classes of tumour model have been proposed [72]. Continuum (tumour-scale) models, discrete (cell-scale) models, and hybrid models. Our focus is on the continuum mathematical model, of which [57] is an example, however for completeness we briefly review other models too.

2.1.1 Continuum Models

In continuum modelling, tumours are treated as a collection of tissue. These models focus on the volume fractions of cells, densities of tumour components (e.g. [6,112]), cells concentrations and

existence of other species such as oxygen. It typically results in a system of PDEs to model the tumours. Most of the early continuum models consist of ordinary differential equations (ODEs), with one or more reaction-diffusion equations (for example, the model in [48], Greenspan's model, is one of the earliest continuum models).

There are many established names and groups that have contributed to develop mathematical models for tumour growth. For example, Adam; who developed one of the early mathematical models in his series of papers [2–4], Adam and Maggelakis [5, 74], Chaplain and co-workers (e.g [9, 83]), Byrne and co-workers (e.g. [26, 27, 32, 33, 57]), and Lowengrub and co-workers (e.g. [72, 73, 116]). The Byrne and Lowengrub groups have developed the most cited of the mathematical models in the recent years. Since the research in this area is increasing rapidly, rather than describe each paper in detail, we refer the reader to the review papers [30] for the mathematical models of cancer, [107] for mathematical models of avascular tumour growth, [10] and [72] for mathematical models of solid tumour, and, from biological and physical viewpoint, [105] for the biological background and biophysical processes of tumour growth.

Continuum mathematical modelling that is based on the multiphase framework is the main focus of this thesis. Hence we now describe some of these models in the remainder of this subsection.

One of the earliest papers describing multiphase modelling of tumour growth is Please et al. [89]. In this paper, the model is a two-phase model of the formation of necrotic regions in solid tumour growth in one dimension. These two phases are the tumour cells, with the assumption that the phase behaves as an inviscid fluid, and extracellular water that moves between the cells as a porous media flow. The oxygen concentration, which diffuses through the tumour surface, determines the rates of proliferation and death of the tumour cells. In this model, the interphase drag forces in the equilibrium equation was neglected. The boundary condition depends on whether the tumour boundary is retreating or advancing, and no surface tension is included in the boundary.

Breward et al. in [26] develop a two phase approach to simulate avascular tumour growth in one-dimension. This work divides the tissue into a tumour cell phase, handled as a viscous liquid, and an extracellular phase, treated as an inviscid liquid. This is derived by applying the principles of conservation of mass (which is including oxygen-dependent terms of cell growth and death), and momentum for incompressible fluids. A feature of the model is the assumption that the pressure in both phases, the cellular and the extracellular liquid, differ from each other. This can be due to the cells interactions, which may attract each other because of overlapping filopodia. Both phases interact by exchange of mass and momentum, respectively, through oxygen regulated cell birth and death, and through drag of inter-phase terms.

A similar two phase model of avascular tumour growth was introduced in [32] by Byrne et al. using the theory of mixtures. The phases in this model are cells and water, which are viewed as equal density incompressible fluids. This paper provides a basis for further developments by considering specific tumour types for which this model might be appropriate. An extension of

this model was later advanced in [33] by Byrne and Preziosi. In this two phase model, mass and momentum balance equations are supplemented by constitutive laws, and these equations are applied to derive the governing equations. A feature of this model is the inclusion of mass exchange between the phases, and net expansion of the solid phase indicating tumour growth.

Multiphase approaches have also been used in the research of vascular tumours. Breward et al.'s model in [27], is an extension of the two-phase model in one dimension from [26], to include blood vessels as a third phase in order to simulate vascular tumour growth. This model consists of conservation of mass and momentum for the volume fractions of these phases: tumour cells, extracellular material and blood vessels. The reaction-diffusion equation for oxygen is replaced by inclusion of a blood vessels phase, so that the effects of vessel and angiogenesis occlusion can be incorporated. In this framework it is demonstrated that these multiphase models can reproduce a lot of the distinctive features associated with the growth of tumours, including the development of a necrotic core behind an outward-moving rim of proliferating tumour cells.

The mathematical models from [26, 27] are extended in [57], where Hubbard and Byrne develop a multiphase model to simulate vascular tumour growth. This model proposed four phases: normal and tumour cells, blood vessels and extracellular material, with the assumption that each phase behaves as a viscous fluid, and that a diffusible nutrient can be clearly distinguished. This model is developed for simulation in two dimensions using an unstructured triangular mesh. The representation of tumour growth is derived from the principles of mass and momentum conservation for the various volume fractions. Section 2.2 describes Hubbard and Byrne's mathematical model, which is the chosen model of this research, in greater detail.

Another example of modelling using four phases is [99], which uses multiphase porous media mechanics, applied to model tumour growth in three dimensions. This model consists of four phases, which are the extracellular matrix, that is treated as a solid phase, with the others as fluid phases; the tumour cells, which include a necrotic portion relying on the conditions of the environment and pressure, the healthy cells, and the interstitial fluid with the dissolved chemical species. The governing PDEs are obtained by a theory of thermodynamically constrained averaging, and these are mass balance for the various phases with the appropriate linear momentum balance equations.

The mathematical models presented by Lowengrub and co-workers are concerned with solid tumour growth. They developed nonlinear mathematical models of solid tumours, which are described in the review paper [72]. Generally, published in the models of this group include calculations for cell velocity and nutrient concentration, in a series of papers [44, 116]. In the first part of this series [116], Lowengrub and co-workers developed a two and three dimensions multiphase continuum modelling of avascular tumour growth. This model is a diffuse interface model. Three phases are considered: host cells, tumour cells, and extracellular water, which are treated as separate phases. The mass and momentum constitutive laws are used to describe the mass flux and the velocities, which are determined using thermodynamic principles. The governing equations consist of advection-reaction-diffusion equations of Cahn-Hilliard type for

updating the cell species volume fractions for each phase, and reaction-diffusion equations to update the substrate component. This model is extended further in the second part of the series [44], which is related to the models that are described in the next subsection.

In more recent work, a two phase-model of tumour spheroids is used in [76]. The phases are the tumour cells, which split into living and necrotic cells, and the interstitial fluid. Here, the extracellular matrix is considered with the cells phase. The governing equations of the model, which are the mass and momentum balance equations, are derived from porous media theory, which are determined by a thermodynamically constrained averaging theory. This mathematical model was applied to analyse a multicellular growth of tumour spheroids in vitro. The same model is used in [75], but with tests of the variable parameters to study their effect in the tumour growth simulation.

2.1.2 Discrete and Hybrid Models

In this subsection, we briefly describe other possible tumour growth models.

Discrete models treat every tumour cell, how it grows and dies, individually. Such models allow us to study individual cells, and their interactions with each other, based around multiple factors including genetic instability. This approach may be divided into two main parts : lattice-based (cellular automata) and lattice-free (agent-based) [72]. A discrete (lattice-based) model is applied to simulate angiogenesis in [9, 73] for example.

Hybrid models combine continuum and discrete models. Such a model is presented in [119], where a continuum model is used to describe the solid tumour growth, while a discrete model is applied to simulate the angiogenesis. Moreover, Frieboes et al [44], developed a three-dimensional multiphase model of angiogenesis and vascular tumour growth. They used a hybrid (lattice-free continuum-discrete) approach to angiogenesis. Further references may be found in [37, 60], and in the review papers [72, 92].

In general, the continuum approach is most useful for problems on large spatial scales, and it is easier than other models to analyse computationally. Standard fast numerical solvers may be developed and/or applied consequently, continuum models typically require less computational cost. On the other hand, these models fail to describe the behaviour of the cells individually: such as cell proliferation and death, and cell-cell interactions. The advantages in continuum approach are disadvantages in the discrete approach. The discrete approach is useful to give a description of individual cells behaviour. However, it typically requires more computational cost, which increases nonlinearly (and rapidly) with problem size. The hybrid approach seeks to contain the advantages of both approaches. However it still needs more work at large-scale systems [60, 72].

In this research we focus only on the continuum approach, focussing on a specific model that is introduced in detail in the following section.

2.2 Chosen Mathematical Model

In this section we describe the continuum mathematical model presented in [57], which is based on multiphase model for studying vascular tumour growth in two dimensions. This model includes four phases: normal/healthy and tumour cells, blood vessels and extracellular material, with the assumption that each phase behaves as a viscous fluid. There are three governing systems of equations in this model, which are: mass balance equations for the volume fraction of each phase (θ_i for $i = 1, ..., 4$), momentum balance equations for the flow of each phase (velocities u_i and v_i and pressure p_i for $i = 1, ..., 4$), and a reaction-diffusion equation for the nutrient/oxygen concentration (c). The nutrient, which is supplied by the blood vessels, is consumed by healthy and tumour cells. The spatial domain is denoted by Ω, and the whole boundary of the domain is denoted by Γ.

The dimensional form of the mathematical model is introduced in [57], which are mass balance equations (1), momentum balance equations (8), and reaction-diffusion equation (14). Here, we present in detail the nondimensionalised form of the mathematical model for the dimensionless volume fractions. The independent and dependent variables have been scaled as:

$$\vec{x} = L_0\vec{x}', \;\; t = \frac{t'}{k_{1,1}}, \;\; \vec{u_i} = L_0 k_{1,1}\vec{u_i'}, \quad p_i = \Lambda p_i', \;\; c = c_v c', \tag{2.1}$$

in which L_0 is a length scale, which is the initial radius of tumour seeded in the healthy tissue, Λ is constant of the cell-cell interaction tension, c_v the blood vessels nutrient concentration and $k_{1,1}$ is the parameter of the birth rate for the normal cell, which used to scale the time t. The whole model is described in detail in [57].

2.2.1 Mass Balance Equations

We start with mass balance equations. All phases considered have the same density, so the mass balance for the healthy cells (θ_1), tumour cells (θ_2) and blood vessel (θ_3) volume fractions are given as follows :

$$\begin{aligned}
\frac{\partial\theta_1}{\partial t'} + \vec{\nabla}'.(\theta_1\vec{u}_1') &= \underbrace{\theta_1\theta_4\left(\frac{c'}{c_p^* + c'}\right)}_{\text{cell birth}} - \underbrace{k_{2,1}^*\theta_1\left(\frac{c_{c_1}^* + c'}{c_{c_2}^* + c'}\right)}_{\text{cell death}} \\
\frac{\partial\theta_2}{\partial t'} + \vec{\nabla}'.(\theta_2\vec{u}_2') &= \underbrace{k_{1,2}^*\theta_2\theta_4\left(\frac{c'}{c_p^* + c'}\right)}_{\text{cell birth}} - \underbrace{k_{2,2}^*\theta_2\left(\frac{c_{c_1}^* + c'}{c_{c_2}^* + c'}\right)}_{\text{cell death}} \\
\frac{\partial\theta_3}{\partial t'} + \vec{\nabla}'.(\theta_3\vec{u}_3') &= \underbrace{-k_3^*\theta_3\mathcal{H}(\theta_1 p_1' + \theta_2 p_2' - p_{crit}^*, \epsilon_3^*)}_{\text{occlusion}} + \underbrace{k_4^*(\theta_1 + \theta_2)\theta_3\left(\frac{\theta_4}{\varepsilon + \theta_4}\right)\left(\frac{c'}{(c_a^* + c')^2}\right)}_{\text{angiogenesis}},
\end{aligned} \tag{2.2}$$

with parameters defined as

$$k^*_{1,2} = \frac{k_{1,2}}{k_{1,1}}, \quad k^*_{2,1} = \frac{k_{2,1}}{k_{1,1}}, \quad k^*_3 = \frac{k_3}{k_{1,1}}, \quad k^*_4 = \frac{k_4}{c_v k_{1,1}},$$

$$c^*_p = \frac{c_p}{c_v}, \quad c^*_a = \frac{c_a}{c_v}, \quad c^*_{c_1} = \frac{c_{c_1}}{c_v}, \quad c^*_{c_2} = \frac{c_{c_2}}{c_v},$$

$$p^*_{crit} = \frac{p_{crit}}{\Lambda}, \quad \epsilon^*_3 = \frac{\epsilon_3}{\Lambda},$$

and the smooth switch function:

$$\mathcal{H}(p, \epsilon) = 0.5(1 + tanh\frac{p}{\epsilon}), \quad \epsilon \ll 1.$$

The primes denote dimensionless variables, where θ_1, θ_2, θ_3 and θ_4 denote the phase volume fractions of the healthy cells, tumour cells, blood vessels and extracellular material respectively, c' is the nutrient/oxygen concentration, p'_i are the pressures in each phase, c_p, c_{c_1}, c_{c_2} denote the nutrient concentration parameters, c_v is the nutrient concentration within the blood vessels, $k_{1,1}$,$k_{1,2}$,$k_{2,1}$,$k_{2,2}$, k_3 and k_4 are pre-defined rate constants, ε is the volume fraction of the extracellular material at half the maximal angiogenesis rate, while c_a is the nutrient concentration at the maximal angiogenesis rate, and p^*_{crit} is the critical pressure for vessel occlusion.

Equations (2.2) are evolution equations for updating the volume fractions of the cells (θ_1, θ_2) and blood vessels (θ_3). To determine the volume fraction for the extracellular phase (θ_4), we use the no-voids condition given by

$$\sum_{i=1}^{4} \theta_i = 1, \tag{2.3}$$

In the first and second equations from the system (2.2), θ_1 and θ_2 are increased during cell birth (proliferation) and decreased because of cell death, while θ_4, the volume fraction of extracellular material, provides the required material for cell growth and birth. The rates of birth are considered to increase when the nutrient concentration increase from $0 \longrightarrow \infty$, whereas, the rate of death is assumed to decrease when the nutrient concentration increases from $0 \longrightarrow \infty$. To satisfy this, the parameters are chosen such $k_{1,2} \geq k_{1,1}$, $k_{2,1} \geq k_{2,2}$ and also $c_{c_1} > c_{c_2}$.

In the third equation from the system (2.2), θ_3 is assumed to increase during angiogenesis and decrease because of vessel occlusion.

The extracellular material phase θ_4 appears in the birth and angiogenesis terms to provide the material needed to create other phases and the material required to remain when another phase is decreased due to vessel occlusion.

The right-hand sides in (2.2) are called the source and sink terms (the total of the source and sink terms must be zero in order to guarantee conservation of mass) which ensure the values of θ_i for $i = 1, 2, 3, 4$, belong on [0,1]. The terms relating to the birth of cells and to angiogenesis have a factor of θ_i and consequently ensure that $\theta_i \geq 0$. Also the source terms are constructed so that, because of (2.3), as $\theta_i \to 1$ other phase volume fractions that are constrained by $\theta_i \geq 0$,

must tend to zero. The initial conditions of the phase θ_i are given by $0 \leq \theta_i(\vec{x}, 0) \leq 1$ (with $\sum_{i=1}^{4} \theta_i(\vec{x}, 0) = 1$), thus ensuring $\theta_4(\vec{x}, t) \geq 0$.

2.2.2 Momentum Balance Equations

This subsection presents the conservation of momentum. The inertial terms in the incompressible Navier-Stokes equation are neglected due to the assumption that the Reynolds number is small. The following equations describe momentum balance for the dimensionless phase velocities $\vec{u_i'}$ and pressures p_i' $(i = 1, ..., 4)$:

$$\sum_{j \neq i} d_{ij}^* \theta_i \theta_j (\vec{u_j'} - \vec{u_i'}) - \theta_i \vec{\nabla}'. \ (\Lambda^* p_i' \mathbf{I}) + \vec{\nabla}'.[\theta_i[\mu_i^*(\vec{\nabla}' \vec{u_i'} + (\vec{\nabla}' \vec{u_i'})^T) + \lambda_i^* (\vec{\nabla}'.\vec{u_i'})\mathbf{I}]] = 0. \quad (2.4)$$

The incompressibility condition implies that

$$\Sigma_{i=1}^{4} \vec{\nabla}'. \ (\theta_i \vec{u_i'}) = 0,$$

where

$$p_1' = p_2' = p_4' + \Sigma'(\theta), \qquad p_3' = \frac{p_3^*}{\Lambda}.$$

Furthermore
$d_{ij}^* = \frac{d_{ij}}{d_{12}}$, $\Lambda^* = \frac{\Lambda}{d_{12} k_{1,1} L_0^2}$,
$\mu_i^* = \frac{\mu_i}{d_{12} L_0^2}$, $\lambda_i^* = \frac{\lambda_i}{d_{12} L_0^2}$,

and

$$\Sigma'(\theta) = \begin{cases} \frac{(\theta - \theta^*)}{(1-\theta)^2} & \text{if } \theta \geq \theta^* \\ 0 & \text{if } \ \theta < \theta^*, \end{cases} \quad (2.5)$$

in which d_{ij}^* is the drag coefficient, $d_{ij} = d_{ji}$ for $i, j = 1, 2, 3, 4$, and $i \neq j$, μ_i denotes the dynamic shear and λ_i denotes the bulk viscosities, which are related through $\lambda_i = -\frac{2}{3}\mu_i$. The total volume fraction of a cell is considered as $\theta = \theta_1 + \theta_2$, θ^* is called the cells natural density. $\Sigma'(\theta)$ describes the pressure in the cell resulting from cell-cell interaction.

The momentum balance equations (2.4), do not depend on time and are linear in $\vec{u_i}'$ and p_4'. Furthermore, p_3' is assumed to be constant ($p_3' = \frac{p_3^*}{\Lambda}$), where p_3^* is the externally-imposed pressure and it is assumed constant. Hence, the equations (2.4) update the velocities $\vec{u_i}$ (four velocities in x-direction and four velocities in y-direction) and only the pressure p_4'.

2.2.3 Reaction Diffusion Equation

The final governing equation in the model of [57] is the reaction-diffusion equation (quasi-steady-state) for the nutrient/oxygen concentration c', which can be written as

$$
\begin{aligned}
D_c^* \nabla'^2 c' &= \underbrace{\theta_3(1-c')}_{\text{replenishment}} \underbrace{-k_{6,1}^* \theta_1 c' - k_{6,2}^* \theta_2 c'}_{\text{baseline consumption}} \\
&- \underbrace{k_{7,1}^* \theta_1 \theta_4 \left(\frac{c'}{c_p^* + c'} \right) - k_{7,2}^* \theta_2 \theta_4 \left(\frac{c'}{c_p^* + c'} \right)}_{\text{consumption due to cell birth}},
\end{aligned} \tag{2.6}
$$

in which

$$
\begin{aligned}
&D_c^* = \tfrac{D_c}{k_5 L_0^2}, \quad k_{6,1}^* = \tfrac{k_{6,1}}{k_5}, \quad k_{6,2}^* = \tfrac{k_{6,2}}{k_5}, \\
&k_{7,1}^* = \tfrac{k_{7,1}}{c_v k_5}, \quad k_{7,2}^* = \tfrac{k_{7,2}}{c_v k_5}.
\end{aligned} \tag{2.7}
$$

where D_c^* is the diffusion coefficient, (k_5, $k_{6,1}$,$k_{6,2}$,$k_{7,1}$,$k_{7,2}$) are pre-defined rate constants, with assumed $\frac{k_{7,1}}{k_{7,2}} = \frac{k_{1,1}}{k_{1,2}}$. The equation (2.6) does not contain a time derivative, and the time scale, which is here scaled with k_5^{-1} instead of $k_{1,1}$ in equation (2.1), used to nondimensionalise is arbitrary. This equation is a nonlinear equation with a single variable c'.

2.3 Summary

In this chapter different types of models for tumour growth are briefly discussed: continuum, discrete and hybrid representations. The literature review is focused on the continuum mathematical models of tumour growth. Moreover, the specific mathematical model chosen for this research is introduced, which included three governing equation systems: the equations (2.2), (2.4) and (2.6), which constitute a four-phase model of vascular tumour growth in two spatial dimensions.

In the following chapter a number of important discretization methods are introduced which can be used to approximate the governing PDEs and produce a numerical solution.

Chapter 3

Review of Scientific Computing Techniques

As mathematical models become more complex especially these including partial differential equations (PDEs), it is generally difficult to find exact solutions, so approximate numerical solutions are required. Scientific computing is a field at the interface between computer science and mathematics. It is applied in a wide range of areas such as engineering, biomedical sciences and others.

In this chapter, some techniques of scientific computing are described, which are related to our work. In Section 3.1 discretization techniques are discussed for approximating PDEs. Following that is Section 3.2, which introduces solution methods for linear equation systems. Section 3.3 gives an idea of how the iterative solution of these discretised systems can be robust, efficient and optimal through the use of preconditioning techniques. Finally, in Section 3.4, the nonlinear system solutions are covered.

3.1 Discretization Methods for PDEs

Numerical simulations of multiphase flow are based on certain systems of PDEs, which are, generally, both time-dependent and nonlinear.

The essential ingredients in all multiphase flow numerical methods are an efficient approach to model the phase flow fields, and applying a strategy to keep track of the interface between these phases [80]. There are several techniques for solving the governing PDEs of basic flow, including: Finite Difference Methods (FDM), Finite Volume Methods (FVM), and Finite Element Methods (FEM).

In this work, a mixed FVM and FEM approach is used for approximating PDEs. In general, the FEM method is used for several reasons. Firstly, in the FEM method the mesh itself need not be structured. As a result of this unstructured form, complex geometries can be treated efficiently and easily. This important advantage of the FEM is not shared by the FDM, which typically requires a structured mesh. The second feature of the FEM is that the discrete problem solution is assumed to have a certain form, that belongs to a function space. Thus the solution is built by interpolating the values of nodes using local basis functions which span this function space. These nodes are the vertices of the elements in the case of linear basis functions. The representation for the function is linked strongly to the geometric representation of the domain, and this form of representation within a subspace of the solution space for the PDE itself is not present for FDM or FVM. The third feature is that the method looks for a solution of an integral form of the PDEs rather than looking for a solution of the PDE itself. Generally, this form is obtained from a weighted residual formulation. Due to this formulation the method becomes able to incorporate differential boundary conditions in a very natural manner. So, the FEM has a combination of properties that is not shared by any other methods [80]. Once the FEM is used for discretization then techniques are required to solve the resulting algebraic systems. Details of The FEM and FVM are discussed in the next subsections.

For completeness we also note some other discretization techniques that are used for PDE approximation, but which are not considered further in this work. These include the boundary element method (BEM) [24], spectral methods [34, 68], spectral element methods [68], collocation methods and discontinuous Galerkin (DG) methods [93].

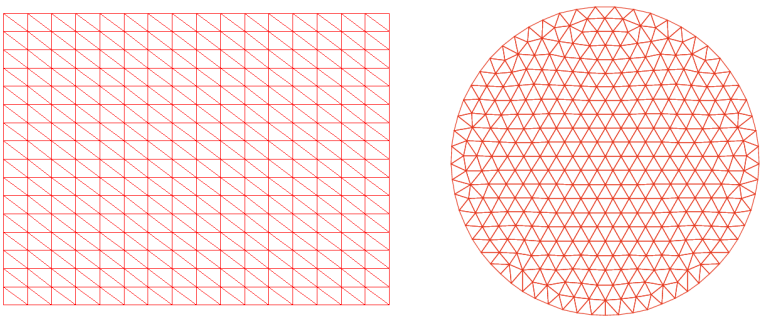
3.1.1 Finite Element Methods (FEM)

The FEM is considered one of the most established methods for the computer solution of partial differential equations in many fields of mathematics, physics and engineering [15]. FEM is a numerical technique based on the weak form of the PDE. The spatial domain is discretised into a set of non-overlapping elements, that cover the domain. The FEM approximation represents the solution locally on each element by interpolating data between a set of chosen points (nodes) within each element. These nodes are numbered locally anti-clockwise on each element, and numbered globally over the global domain. Integrating the weak form over the set of elements reduces the PDE to a system of algebraic equations which can be solved with standard techniques on a computer.

The set of non-overlapping elements that cover the domain is termed the spatial mesh, or grid. In the case of a square or rectangular domain this can be a simple uniform partition of the space, but more generally can be an unstructured collection that covers any arbitrary shape. In this work, two different shapes of the grid are used, one is a square regular triangular grid and the other is a circular unstructured triangular grid (see Figure 3.1).

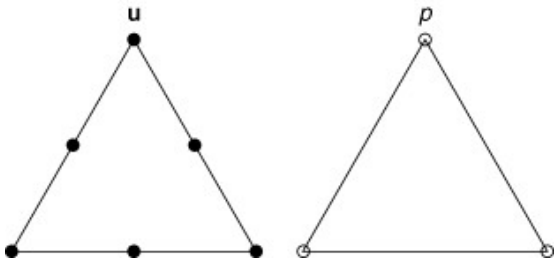
There are many common types of element in 2D, though we only use two in this work: firstly, the linear element has three nodes on the vertices of the element. Secondly, the quadratic

Figure 3.1: Regular and unstructured grids



element with six nodes (three in the vertices and three in the edges). In the model under study, the reaction diffusion equation (2.6) is approximated by a linear FEM, which uses linear basis functions (piecewise) for the nutrient c. While, the momentum balance equations (2.4) are approximated by a mixed FEM (Taylor-Hood FEM), which uses linear and quadratic basis functions (piecewise) for different variables (see Figure 3.2). In equations (2.4), we have a coupled systems of equations with quadratic approximation of the velocities (u_i, v_i) and linear for the pressure p_4.
In the next two subsubsections, the linear FEM and Taylor-Hood FEM are described briefly, using simple examples, to show the difference between them.

Figure 3.2: Taylor-Hood Element



Moreover, there are many books and papers that specialise in the FEM. Some of these books focus on the practical aspects (e.g. [91]), whereas the theoretical and analytical aspects of finite element are covered, respectively, in [35, 103].

3.1.1.1 The Linear FEM

Here, the FEM is described for a two dimensional problem, specifically the linear FEM for Poisson's equation in two dimensions [61]:

$$-\triangle u(x) = f(x), \qquad for \quad x \in \Omega \subset \mathbb{R}^2, \tag{3.1}$$

where $\triangle$ is the Laplace operator, and the domain of the solution given by $\Omega \subset \mathbb{R}^2$. Equation (3.1) is a linear second order elliptic problem with boundary conditions:

$$u = u_E \ on \ \Gamma_D \qquad and \qquad \frac{\partial u}{\partial \hat{n}} = g \ on \ \Gamma_N \tag{3.2}$$

where $\partial\Omega = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \{\}$. Γ_D denotes the Dirichlet boundary on one part of the boundary and Γ_N is the Neumann boundary on the other.
The weak formulation of Poisson's equation is obtained by multiplying by a weight function ω and using integration by parts:

$$\int_\Omega \bigtriangledown u \,.\, \bigtriangledown \omega \ dx - \int_{\Gamma_N} \frac{\partial u}{\partial \hat{n}} \, \omega \ ds = \int_\Omega f\omega \ dx + \int_{\Gamma_D} \frac{\partial u}{\partial \hat{n}} \, \omega ds. \tag{3.3}$$

The domain Ω is divided to a set of triangular elements $\Omega_e = \{e\}$. The solution on each element is represented by a polynomial interpolant, in this case polynomials of degree one [61]. Define the set of piecewise linear functions $\{L_1, ..., L_N\}$ where

$$L_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

in which L_i is linear on each element and x_j is the j^{th} node in the mesh.
The finite element approximation to u can be written as

$$\bar{u} = \sum_{i=1}^{n_B+n_E} \bar{u_i} L_i(x) \tag{3.5}$$

where n_B is the number of interior nodes + nodes on Γ_N, n_E is the number of nodes on Γ_D. The values of $\bar{u_i}$ are unknown for $i = 1, ..., n_B$ and are given by $u = u_E \ on \ \Gamma_D$, for $i = n_B + 1, ..., n_B + n_E$.

If we replace u by $\bar{u}$ and choose a weight function $\omega = L_j$ for $j = 1, ..., n_B$ in equation (3.3), then we will get a system of n_B equations with the assumption that the Neumann boundary condition $\frac{\partial u}{\partial \hat{n}} = g \equiv 0 \ on \ \Gamma_N$, so the finite element equations are given by:

$$\sum_{i=1}^{n_B} \bar{u_i} \int_\Omega \bigtriangledown L_j . \bigtriangledown L_i \ dx = \int_\Omega f L_j \ dx - \sum_{i=n_B+1}^{n_B+n_E} \bar{u_i} \int_\Omega \bigtriangledown L_j . \bigtriangledown L_i \ dx. \tag{3.6}$$

This can be written as:

$$A\bar{u} = f \tag{3.7}$$

where A is the global stiffness matrix ($n_B \times n_B$ matrix) with $A_{ji} = \int_\Omega \bigtriangledown L_j . \bigtriangledown L_i dx$, f is an $n_B \times 1$ known vector and $\bar{u}$ is the unknown vector. To find the global solution in the whole domain, we must assemble the local equations system on each element to compute the global

system. Then we must solve the global system of equations using direct or iterative methods (see Section 3.2).

Now, we consider that i_{local} and j_{local} are the vertex numbers ($i_{local}, j_{local} = 1, 2, 3$) and $e = 1, ..., E$ denote triangular elements, so the entry A_{ji} of the matrix A may be obtained from:

$$A_{ji} = \int_{\Omega} \bigtriangledown L_j . \bigtriangledown L_i \; dx = \sum_{e=1}^{E} \int_{\Omega_e} \bigtriangledown L^e_{j_{local}} . \bigtriangledown L^e_{i_{local}} \; dx = \sum_{e=1}^{E} A^{(e)}_{j_{local} i_{local}}. \tag{3.8}$$

Similarly, we compute the right hand side from equation (3.1) as:

$$f_j = \int_{\Omega} f L_j \; dx = \sum_{e=1}^{E} \int_{\Omega_e} f L^e_{j_{local}} dx = \sum_{e=1}^{E} f^{(e)}_{j_{local}}. \tag{3.9}$$

In general, we compute A_{ji} $(i, j = 1, .., n_B)$ in each element e where e= 1,...,E, before that we will define the position of three vertices of each element Ω_e (where $i_{local}, j_{local} = 1, 2, 3$) by

$$S^{(e)}_{i_{local}} = S_{icon(e, i_{local})},$$

where "icon" is an integer array that stores the global node number of each vertex of each element.
Similarly the piecewise functions on each element e will be defined as

$$L^{(e)}_{i_{local}} = L_{icon(e, i_{local})},$$

So

$$A^{(e)}_{j_{local} i_{local}} = \int_{\Omega_e} \bigtriangledown L^{(e)}_{j_{local}} . \bigtriangledown L^{(e)}_{i_{local}} dx = \frac{1}{4Area^{(e)}} (b^{(e)}_{j\,local} b^{(e)}_{i\,local} + c^{(e)}_{j\,local} c^{(e)}_{i\,local}),$$

and

$$f^{(e)}_{j_{local}} = \int_{\Omega_e} L^{(e)}_{j_{local}} f(x) \; dx,$$

where

$$b^{(e)}_{[123]} = S^{(e)}_{[231]y} - S^{(e)}_{[312]y},$$
$$c^{(e)}_{[123]} = S^{(e)}_{[312]x} - S^{(e)}_{[231]x},$$

and

$$2Area^{(e)} = c^{(e)}_3 b^{(e)}_2 - c^{(e)}_2 b^{(e)}_3.$$

This is the linear FEM , which is based upon the use of piecewise linear functions. Each

triangular element has three nodes on vertices.

3.1.1.2 Taylor-Hood FEM (Stokes Problem)

The mixed finite element method involves the approximation of more than one solution field simultaneously on the physical domain [94]. In Stokes problems, the mixed finite element method is based on a velocity and pressure formulation. The Stokes equations arise by simplifying the incompressible steady Navier-Stokes equations by neglecting the nonlinear terms and taking the Reynold's number (Re) as unity [61].

The Stokes Equation may be written in the stress-divergence form which written as:

$$\begin{aligned} -\bigtriangledown .\sigma &= f \\ \bigtriangledown .U &= 0, \end{aligned} \tag{3.10}$$

where the stress tensor

$$\sigma_{ij} = -p\delta_{ij} + \nu(\frac{dU_i}{dx_j} + \frac{dU_j}{dx_i})$$

and

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

with boundary condition

$$U = U_D \quad on \quad \Gamma_D$$

$$\sigma.\vec{n} = 0 \quad on \quad \Gamma_N,$$

where $U = (u, v)$ and p denote the velocities in x and y directions and pressure respectively. The domain $\partial\Omega = \Gamma_D \cup \Gamma_N$. The velocities have Dirichlet boundary condition, while the pressure boundary is imposed at a single point in the boundary to obtain a unique solution. Let us consider test functions ω and ϕ. Now we can multiply the equations (3.10) by test functions ω and ϕ then use the divergence theorem to obtain the following weak form:

$$\begin{aligned} -\int_{\partial\Omega} \omega\sigma.\vec{n}d\Omega + \int_\Omega \bigtriangledown\omega.\sigma d\Omega &= \int_\Omega \omega f d\Omega \\ \int_\Omega \phi \bigtriangledown .U d\Omega &= 0. \end{aligned} \tag{3.11}$$

The finite element approximation solution to u, v and p can be written as,

$$\bar{u} = \sum_{i=1}^{n_B+n_E} \bar{u_i} Q_i(x)$$

$$\bar{v} = \sum_{i=1}^{n_B+n_E} \bar{v_i} Q_i(x)$$

$$\bar{p} = \sum_{i=1}^{m_B+m_E} \bar{p_i} L_i(x)$$

in which $n_B + n_E$ is the total number of velocity points on the edges and vertices of a mesh, n_E denotes the quadratic node points on the Dirichlet boundary, $m_B + m_E$ is the total number of pressure points on the vertices of the mesh, m_E is the number of vertices on the Dirichlet pressure conditions (typically $m_E = 1$). Also, Q_i and L_i are the piecewise quadratic and piecewise linear basis functions respectively. This choice is known as the Taylor-Hood approximation. Each triangular element has 3 nodes at the vertices, and a node at the midpoint of each edge. We may define a function $Q_i(x)$ (where i is a node on the edges or at the vertices) which is equal 1 at node i, and 0 otherwise, and quadratic on each element. In this case, each element has 6 nodes, so the values at these nodes will define a quadratic on that element, so

- when node i is at vertex $Q_i = L_i(2L_i - 1)$,
- when node i is on an edge $Q_i = 4L_jL_k$,

where j and k are the vertices at the end of the edge i.
The number of unknowns is $(2n_B + m_B)$. We replace ω by Q_j for $j = 1, 2, ..n_B$, and ϕ by L_j for $j = 1, 2, .., m_B$. This leads to

$$\sum_{i=1}^{n_B} \int_\Omega \big[(2\nu \frac{\partial Q_i}{\partial x}\frac{\partial Q_j}{\partial x} + \nu \frac{\partial Q_i}{\partial y}\frac{\partial Q_j}{\partial y})u_i + (\nu \frac{\partial Q_i}{\partial x}\frac{\partial Q_j}{\partial y})v_i\big] d\Omega - \sum_{i=1}^{m_B} \int_\Omega (L_i \frac{\partial Q_j}{\partial x}) p_i d\Omega = \int_\Omega Q_j f_x d\Omega,$$

$$\sum_{i=1}^{n_B} \int_\Omega \big[(\nu \frac{\partial Q_i}{\partial y}\frac{\partial Q_j}{\partial x})u_i + (\nu \frac{\partial Q_i}{\partial x}\frac{\partial Q_j}{\partial x} + 2\nu \frac{\partial Q_i}{\partial y}\frac{\partial Q_j}{\partial y})v_i\big] d\Omega - \sum_{i=1}^{m_B} \int_\Omega (L_i \frac{\partial Q_j}{\partial y}) p_i d\Omega = \int_\Omega Q_j f_y d\Omega,$$

$$\sum_{i=1}^{n_B} \big[\int_\Omega L_j \frac{\partial Q_i}{\partial x} d\Omega\big] u_i + \sum_{i=1}^{n_B} \big[\int_\Omega L_j \frac{\partial Q_i}{\partial y} d\Omega\big] v_i = -\sum_{i=n_B+1}^{n_B+n_E} \int_\Omega \big[L_j \frac{\partial Q_i}{\partial x} u_i + L_j \frac{\partial Q_i}{\partial y} v_i\big] d\Omega.$$

Hence, the block-matrix of the system can be written as:

$$\begin{bmatrix} K_{xx} & K_{xy} & B_x \\ K_{yx} & K_{yy} & B_y \\ B_x^T & B_y^T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ f_p \end{bmatrix}$$

in which

$$(K_{xx})_{ji} = \int_\Omega (2\nu \frac{\partial Q_i}{\partial x}\frac{\partial Q_j}{\partial x} + \nu \frac{\partial Q_i}{\partial y}\frac{\partial Q_j}{\partial y}) \quad d\Omega,$$

$$(K_{xy})_{ji} = \int_\Omega \nu \frac{\partial Q_i}{\partial x}\frac{\partial Q_j}{\partial y} \quad d\Omega,$$

$$(K_{yx})_{ji} = \int_\Omega \nu \frac{\partial Q_i}{\partial y}\frac{\partial Q_j}{\partial x} \quad d\Omega,$$

$$(K_{yy})_{ji} = \int_\Omega (\nu \frac{\partial Q_i}{\partial x}\frac{\partial Q_j}{\partial x} + 2\nu \frac{\partial Q_i}{\partial y}\frac{\partial Q_j}{\partial y}) \quad d\Omega,$$

and

$$(B_x)_{ji} = -\int_\Omega L_j \frac{\partial Q_i}{\partial x} d\Omega, \qquad (B_y)_{ji} = -\int_\Omega L_j \frac{\partial Q_i}{\partial y} d\Omega.$$

Generally, we obtain the saddle point system:

$$\underbrace{\begin{bmatrix} K & B \\ B^T & 0 \end{bmatrix}}_{A} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ f_p \end{bmatrix}$$

where K known as the vector-Laplacian matrix, and B is the divergence matrix. Note that, in this case, the matrix A is symmetric indefinite.

In summary, the linear FEM approximates the PDEs by using the linear basis function for one variable, while the Taylor-Hood FEM approximates the PDEs by using both linear and quadratic basis functions for different variables. Both previous examples are time-independent problems.

3.1.1.3 The Linear FEM for a Time Dependent Problem

Here, we describe the FEM for two space dimensions to solve a simple time-dependent PDE. We examine finite element schemes for the linear diffusion equation taken from [61]:

$$\frac{\partial}{\partial t} u(\underline{x}, t) = \bigtriangledown^2 u(\underline{x}, t) + f(\underline{x}, t), \qquad for \quad (\underline{x}, t) \in \Omega \times (0, T] \tag{3.14}$$

in which $\underline{x}$ are spatial variables and t is the time variable, T is the final time, and the initial condition

$$u(\underline{x}, 0) = u^0(x) \qquad for\ all\ \underline{x} \in \Omega.$$

The spatial boundary is allowed to have Dirichlet (Γ_D) and Neumann (Γ_N) conditions given by

$$u = u_E \ \ on\ \Gamma_D \qquad and \qquad \frac{\partial u}{\partial \hat{n}} = g \ \ on\ \Gamma_N \quad for\ all\ t \in (0, T],$$

where $\partial\Omega = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \{\}$. Here, the Laplacian operator applies only to spatial variables which are grouped as $\underline{x}$ ($\bigtriangledown^2 u(\underline{x}, t) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$).

The weak form of the diffusion equation is obtained by multiplying by a test function, $L_j(\underline{x})$, which has no time dependence. Then, integration is only on the spatial domain. So the system of equations can be written as

$$\int_\Omega \frac{\partial u}{\partial t} L_j d\underline{x} = \int_\Omega \bigtriangledown^2 u L_j d\underline{x} + \int_\Omega f L_j d\underline{x} \tag{3.15}$$

and then use the divergence theorem to give

$$\int_\Omega \frac{\partial u}{\partial t} L_j d\underline{x} = -\int_\Omega \bigtriangledown u. \bigtriangledown L_j d\underline{x} + \int_{\partial\Omega} \frac{\partial u}{\partial \hat{n}} L_j ds + \int_\Omega f L_j d\underline{x}. \tag{3.16}$$

As before we divide the domain into a set of triangles with vertices from 1 to $n_B + n_E$ where n_B are the Dirichlet boundary. We can define the $n_B + n_E$ functions $L_j(\underline{x})$ which is the linear basis functions centred on the j^{th} node of the mesh.

The finite element approximation to u can be written as

$$\bar{u} = \sum_{i=1}^{n_B+n_E} \bar{u_i}(t) L_i(x) \tag{3.17}$$

in which the value of $\bar{u_i}$ are unknown for $i = 1, ..., n_B$ and are given by the Dirichlet boundary condition, $u = u_E$, for $i = n_B, ..., n_B + n_E$. Now, we replace u by $\bar{u}$ in Equation (3.16) to obtain a system of n_B equations for n_B unknowns as follows

$$\begin{aligned} \sum_{i=1}^{n_B} \frac{d\bar{u}_i}{dt} \int_\Omega L_i L_j d\underline{x} = \; & - \sum_{i=1}^{n_B} \bar{u_i} \int_\Omega \bigtriangledown L_i. \bigtriangledown L_j d\underline{x} + \int_{\Gamma_N} g L_j ds + \int_\Omega f L_j d\underline{x} \\ & - \sum_{i=n_B+1}^{n_B+n_E} \bar{u_i} \int_\Omega \bigtriangledown L_i. \bigtriangledown L_j d\underline{x} - \sum_{i=n_B+1}^{n_B+n_E} \frac{d\bar{u}_i}{dt} \int_\Omega L_i L_j d\underline{x}. \end{aligned}$$

We may express this in matrix notation, as follows:

$$M \frac{d\bar{\underline{u}}}{dt} = -K\bar{\underline{u}} + f(t). \tag{3.18}$$

This system of equations is not an algebraic system, it is a system of ordinary differential equations (ODEs), where K is the global stiffness matrix and M is the Galerkin mass matrix with entries given by $K_{ji} = \int_\Omega \bigtriangledown L_j. \bigtriangledown L_i d\underline{x}$ and $M_{ji} = \int_\Omega L_j L_i d\underline{x}$, respectively.

As for the stiffness matrix it is possible to calculate the mass matrix on each element as follows

$$M_{JI}^{(e)} = \int_\Omega L_J^{(e)} L_I^{(e)} d\underline{x} = \begin{cases} Area^{(e)}/6 & \text{if } I = J \\ Area^{(e)}/12 & \text{otherwise.} \end{cases} \tag{3.19}$$

The θ method is used for solving the differential equation

$$\frac{dy}{dt} = F(t, y), \quad y(0) = y^0,$$

which is given by

$$\frac{y^{n+1} - y^n}{k} = (1-\theta)F(t^n, y^n) + \theta F(t^{n+1}, y^{n+1}), \quad for \;\; n = 0, 1, ...$$

in which $t^n = k * n$. Applied to the system (3.18) gives

$$M\frac{(\underline{\bar{u}})^{n+1} - \underline{\bar{u}}^n}{k} = (1-\theta)(-K\underline{\bar{u}}^n + ft^n) + \theta(-K\underline{\bar{u}}^{n+1} + f(t^{n+1})),$$

which can be written as

$$(M + k\theta K)\underline{\bar{u}}^{n+1} = (M - k(1-\theta)K)\underline{\bar{u}}^n + kf_\theta, \tag{3.20}$$

where $f_\theta = [(1-\theta)f(t^n) + \theta f(t^{n+1})]$. Note that each time step will involve the solution of a single linear system. Also, this system is symmetric and positive definite when $\theta \geq 0$ [61].

3.1.2 Finite Volume Methods (FVM)

The FVM is a spatial discretization technique most commonly used for approximating hyperbolic partial differential equations arising from conservation laws. This method was developed by Spalding and Patankar [87]. The FVM is now a very popular method used in computational Fluid Dynamics (CFD) e.g. [110].

The FVM is a numerical method based on dividing the spatial domain into a set of control volumes (cells) then approximating by integrating the governing equations over each control volume using the divergence theorem and then solving the discrete algebraic equation systems. This method can be used with structured or unstructured grids. Further information about FVM is found in [62, 110]. There are different storage schemes for the FVM; the cell-centred scheme where the variables are stored at the cell centre; and the vertex-centred scheme where the variables are stored at the vertices of the cell [16]. In the following we describe a simple cell-centred scheme.

In general form, the two-dimensional conservation law can be written as

$$\frac{\partial u}{\partial t} + \vec{\nabla}.\vec{F} = 0, \tag{3.21}$$

where $u = u(x, y, t)$ is unknown, and $\vec{F} = (f, g)$ is the flux function. Let Δ denote a control volume, then the integral form of the conservation law (3.21) is given by

$$\int\int_\Delta \frac{\partial u}{\partial t} dxdy + \int\int_\Delta (\vec{\nabla}.F)dxdy = 0. \tag{3.22}$$

The Gauss divergence theorem is used for the flux integral to produce

$$\int\int_{\Delta} \frac{\partial u}{\partial t} dxdy + \oint_{\partial\Delta} \vec{F}.d\vec{n} = 0 \tag{3.23}$$

in which $\vec{n}$ is the outward-pointing unit normal to the boundary $\partial\Delta$.

By approximation the previous integral leads to

$$\frac{\partial u_i}{\partial t} = -\frac{1}{V_i} \sum_{k=1}^{E} \vec{F^*}_k.\vec{n_k}, \tag{3.24}$$

where u_i is the average value of u associated with the i^{th} control volume, V_i represents the area of the i^{th} control volume, E is the number of the edges of the control volume, and $\vec{F^*}_k$ is called the numerical flux across edge k of the volume. The flux term can be approximated by using a standard upwind scheme [110], for example in the i_{th} cell and its k_{th} edge the upwinding is applied such that

$$\vec{F^*}(u_i, u_k).\vec{n_k} = \begin{cases} u_i\vec{\lambda}.\vec{n_k} & \text{if } \vec{\lambda}.\vec{n_k} \geq 0 \\ u_k\vec{\lambda}.\vec{n_k} & \text{otherwise,} \end{cases} \tag{3.25}$$

in which u_k is the value of u in the adjacent cell, and $\vec{\lambda} = \left(\frac{\partial f}{\partial u}, \frac{\partial g}{\partial u}\right)^T$ is the so-called advection velocity.

By using a forward Euler scheme for the time derivative, which is defined as

$$\frac{\partial u_i}{\partial t} = \frac{u_i^{n+1} - u_i^n}{\Delta t}, \tag{3.26}$$

equation (3.24) becomes

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{V_i} \sum_{k=1}^{E} \vec{F^*}_{ik}.\vec{n_{ik}} \tag{3.27}$$

in which Δt is the time step size.

According to [56, 57], the algorithm (3.25) is known as the cell-centred MUSCL scheme, which is a conservative, upwind, finite volume scheme, with forward Euler time stepping (3.26). The hyperbolic PDEs (2.2) in the mathematical model introduced in the previous chapter may be discretised using this scheme on a two-dimensional unstructured triangular grid.

3.2 Solution of Linear Equation Systems

This section will consider numerical methods to solve a linear system of equation in the form

$$Au = f \tag{3.28}$$

in which $A \in \mathcal{R}^{N\times N}$, u and f are N-vectors.

The structure of matrix A can help to determine the type of the numerical methods used for solving the linear equation systems. Any matrix A that has $\mathcal{O}(N^2)$ non-zero entries, where N is the number of equations, is known as a dense matrix. In fact, an important drawback of a dense matrix is that it requires a large amount of memory and time for solving large problems because of storing the whole matrix. In contrast, this problem does not arise when the matrix A is sparse.

A sparse $N \times N$ matrix contains many zero entries and has only $\mathcal{O}(N)$ non-zeros. The sparse matrices techniques are used in many computational science applications to exploit this property in order to reduce both storage and computation effort requirements [52, 96]. Significantly, the matrices arising from FEM schemes are sparse matrices.

Generally, there are two main types of numerical methods for solving the sparse linear equations systems (3.28):

- Direct methods to find the exact solution (up to rounding error).
- Iterative methods to find an approximate solution.

3.2.1 Direct Methods

Theoretically, in the absence of rounding error, direct methods give an exact solution to a problem in a finite number of steps. The direct methods divide into two related classes:

- Elimination methods: such as Gauss elimination.
- Factorisation (decomposition) methods: such as LU factorisation and Cholesky factorisation for symmetric positive-definite matrices.

There are many problems facing us when using direct methods, which relate to the storage and handling of sparse matrices. The fill-in of a matrix are those entries that change from a zero value to a non-zero value during the execution of the algorithm (either elimination or factorisation). Here the objective is minimizing the fill-in, to reduce the memory requirements, whilst maintaining stability, so it is necessary to use the sparse direct methods to solve the problem.

Sparse factorisation methods solve systems of linear equations $Au = f$ by decomposing the coefficient matrix A into the form LU for a nonsymmetric matrix or LL^T for a symmetric matrix. The sparse direct solver is achieved in four phases [47]:

- An ordering phase, which reorders the rows and columns to exploit sparse structure.
- An analysis phase (so called symbolic decomposition), that analyses the matrix structure to determine a pivot sequence for efficient decomposition (minimizing fill-in) and to create suitable data structures for the factors and allocate memory for L and U.
- A numerical decomposition phase, that computes the L and U factors.

– A solve phase. The solution u is computed by forward and backward substitution.

The multifrontal massively parallel sparse direct solver, known as MUMPS, is a software application for sparse, parallel, direct solution of large linear systems equations that is used in [57]. MUMPS is based on a multifrontal method, which permits the parallel implementation of a direct factorisation $A = LU$ or $A = LDL^T$ depending on if the matrix is nonsymmetric or symmetric. The interested readers are directed to [7, 8, 50] for more details about MUMPS. Other sparse direct solvers have been developed, based on similar principles. We cite here SuperLU by way of one further example of this class of solver [38].

Sparse direct algorithms are efficient for moderate problem sizes. Complexity is typically $\mathcal{O}(N^{3/2})$ in which N is the number of unknowns [45]. The memory and computational cost requirement for solving very large linear systems in 2D problems or for solving 3D problems may therefore cause a challenge to efficient direct solution methods [96]. Consequently, iterative methods have been developed for these cases. Sparse iterative techniques can, in certain circumstances, produce $\mathcal{O}(N)$ algorithms [45], hence the purpose (and the challenge) of this work is to replace the use of MUMPS in [57] by designing optimal and efficient iterative algorithms for the problem of interest.

3.2.2 Iterative Methods

Iterative methods are designed primarily for solving large sparse linear systems. These methods produce a series of approximate solutions that are designed to converge to the exact solution of the linear system. Typically, iterative methods are easier to implement efficiently than sparse direct solvers [96]. By reading the history of iterative methods [17, 98], we can note that on digital computers these methods were widely implemented in two main periods. The first period began from the 1950s to the early 1970s, and is controlled by stationary iterative methods. The second period started from the mid 1970s and is controlled by Krylov subspace methods such as conjugate gradient methods and others [17,98]. In the remainder of this section we discuss some modern iterative methods such as multigrid methods (which also emerged in the 1970s [19]) and Krylov subspace iterative methods.

3.2.2.1 Multigrid Methods

Multigrid methods are a numerical technique for solving linear systems that arise from a grid-based discretization of elliptic PDEs, using a hierarchy of such discretizations. These methods have complexity $\mathcal{O}(N)$, i.e. optimal convergence behaviour [28]. There are two main types of multigrid method:

– Geometric multigrid (GMG), which require hierarchical spatial meshes.

– Algebraic multigrid (AMG), which require only the information of a single mesh but produce a hierarchy of linear systems.

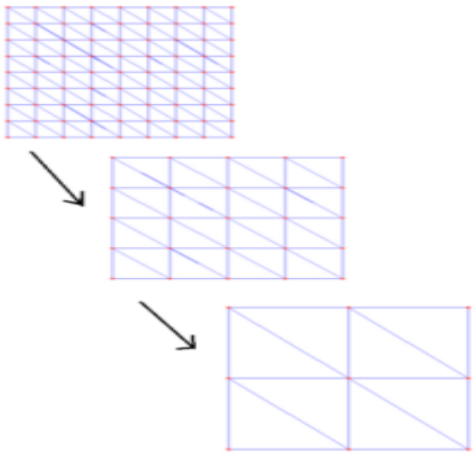


Figure 3.3: Geometric multigrid (hierarchy grids) in two dimensions

The first studies of the multigrid method are given by Fedorenko (1961,1964) [42, 43], who proposed the first multigrid scheme, which is a two grid scheme for the standard five point finite difference discretization of Poisson's equation in a rectangular domain. He also emphasized the complementary roles of the Jacobi iteration and the coarse-grid correction and proposed the first multigrid algorithm, and proved the convergence analysis. His work was generalized by Bakhvalov in 1966 [13]. However, the method was not put into practice at the time. The first practical and efficient demonstration of the multigrid method was recognised in 1977 by Achi Brandt [19]. In this paper, he described the multigrid method geometrically for solving linear and nonlinear boundary value problems. This paper is one of the most important papers that describes multigrid methods.

Use of geometric multigrid methods for geometrically complex applications was initially limited because of the need for the definition of a hierarchy of nested meshes as shown in Figure 3.3. One approach to overcome this problem efficiently is the Algebraic multigrid method (AMG). AMG is used when the GMG is difficult to use or when just the matrix information is given. The first introduction of AMG was in the early 1980s [20–23, 104]. However, research continued into the late 1980s and early 1990s [36, 54, 69, 70, 90]. AMG builds a hierarchy of matrices directly, without the need for coarser grids, and creates the necessary transfer operators by using information from the original matrix without any geometric information. However, there are disadvantages for AMG, that can present a high computational cost in building the matrix hierarchy and the transfer operators [88].

In fact, multigrid methods are known to provide the most efficient and optimal numerical solution methods because their convergence rate is independent on the mesh size. However, these methods are restricted in the problems we can solve, and have been generally used for

solving second-order elliptic PDEs [82, 117]. In this thesis, the mathematical model contains both hyperbolic and first-order elliptic PDEs. Although, multigrid methods can not be used as solver for all of these discrete systems, they can be used as an efficient preconditioner for an external iterative solver in order to accelerate the convergence rate of the Krylov subspace iterations (for more details, see [114]).

In this work, AMG is used with a Krylov subspace method to achieve optimal efficiency. Generally, the convergence rate of the preconditioned Krylov method depends on the preconditioning matrix properties. If this matrix is spectrally equivalent to the original matrix, the convergence rate may be independent of the problem size. Consequently the task of deriving efficient preconditioning is essential. The multigrid method is one way to solve this problem, and the resulting complexity of computation can be of optimal order [17].

In the late 1980s, there was further research to find alternative preconditioning techniques to solve problems optimally. The first attempts gave results with nearly optimal preconditioning matrices, proposed in [118] and generalized in [14] using hierarchical basis functions. Later attempts introduced in [109], use the standard nodal basis functions and a multilevel ordering of the points of the triangulation. Both studies were based on nested discretizations and proved to be nearly optimal: convergence for 2D boundary value problems increases as $\mathcal{O}(logh)$ or $\mathcal{O}((logh)^2)$, where h is size of mesh. Some early papers to use Algebraic multigrid as a preconditioner to achieve optimal, or nearly optimal convergence are [11, 12], which use algorithms that may be interpreted as approximate Schur complement methods. In [88] the authors present an approach to select the coarse grids that allows them to use the AMG preconditioned CG method to solve small and medium ill-conditioned problems. This approach leads to a significant reduction in the AMG solver times and the space of storage required, without losing the robustness.

There are numerous types of cycle schemes for the multigrid technique, such as V-cycle and W-cycle. The idea of the V-cycle scheme is to replace problem on fine grid by an approximation on a coarser grid then solve the problem approximately, and use the solution as a starting guess for the finer grid, which is iteratively updated. These steps are called pre-smoothing on the grids h, $2h$ and $4h$. A transfer operation of vectors on grid h to vectors on grid $2h$ is called restriction. The pre-smoothing and restriction steps are repeated until the algorithm reaches the coarsest grid. Then solve the coarsest grid problem directly. Following by correction the solution on each grid back to the finest grid. These steps are called post-smoothing. A transfer operation of vectors on grid $2h$ to vectors on grid h is called prolongation (or interpolation). Generally, the V-cycle scheme is the simplest multigrid cycle scheme and has only one coarsest grid, so the coarse-grid correction scheme is applied only once at each level of multigrid. While the W-cycle has at least two coarsest grids (see Figure 3.4, which is 3-grid V and W cycles).

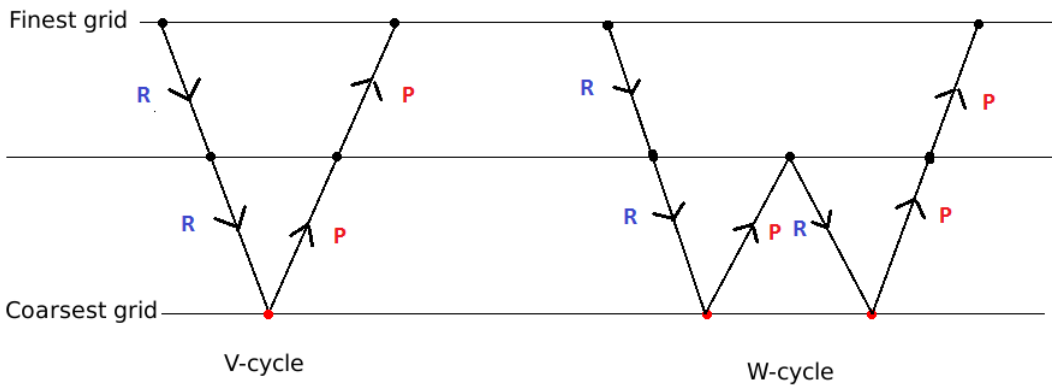


Figure 3.4: V-cycle and W-cycle multigrid solution method, black points called smooth, red points called coarest grids, R is the Restriction, and P is the prolongation

In this thesis, an Algebraic multigrid (AMG) method is used as a solver for the part of a new preconditioner in the linear system, and used as preconditioner with Newton-Krylov method in the nonlinear systems. The calculation of the algebraic multigrid algorithm only needs single grid and depends on matrix coefficients. The coarse grid matrix coefficients are formed by using the previous matrix information. The following procedure is called AMG two-grid correction cycle [28]:

Algorithm 1 two-grid V-cycle for Algebraic multigrid method

1: Smooth $A^h u^h = f^h$ with initial guess v^h
2: Compute the residual as $r^h = f^h - A^h v^h$ and restrict it to the coarse grid as $r^{2h} = I_h^{2h} r^h$
3: Solve $A^{2h} e^{2h} = r^{2h}$
4: Compute $e^h = I_{2h}^h e^{2h}$
5: Correct the fine grid approximation $v^h = v^h + e^h$
6: Smooth $A^h u^h = f^h$ with initial guess v^h

The thing that defines AMG is the way that A^{2h}, I_{2h}^h and I_h^{2h} are defined. The V-cycle starts with finest grid Ω^h and uses an iterative solver to apply pre-smoothing to the residual on Ω^h. This smoothing take a few iterations of Jacobi or Gauss-Seidel method. Then restrict the residual $r^h = f^h - A^h u^h$ to the coarsest grid using the restriction operator I_h^{2h}. In the coarsest grid a direct solver is used to find the solution for $A^{2h} e^{2h} = r^{2h}$ then corrected the solution back to a finest grid by apply the prolongation operator I_{2h}^h. On the finest grid an iterative solver is used to apply post-smoothing. This is purely algebraic.

One of the most important introductions to MG is [28], which explains in detail how to use 2-D finite difference multigrid for solving elliptic PDEs. This book includes linear and nonlinear problems, multilevel adaptive and AMG methods, and shows some examples of how multigrid methods can be applied.

Krylov subspace methods and their preconditioners are described in the following subsections.

3.2.2.2 Krylov Subspace Methods

Krylov subspace methods are one of the most important numerical methods for solving large sparse problems [39]. Large sparse linear system problems (3.28) appear in many scientific computing applications, in which A is a non-singular matrix. The Krylov subspace $\mathcal{K}_m$ is a vector space

$$\mathcal{K}_m(A, v) = span\{v, Av, A^2v, ..., A^{m-1}v\} = \{p(A)v : p \in \mathcal{P}_m\}$$

where v is a nonzero vector, and $\mathcal{P}_m$ is the space of all polynomials of degree (m-1). The Krylov subspace $\mathcal{K}_m$ is the column space of the Krylov matrix

$$K_m = [v \;\; Av \;\; A^2v \;\; ... \;\; A^{m-1}v].$$

There are different Krylov subspace methods depending on the properties of the system matrix such as: Conjugate Gradient Method (CG) for solving sparse symmetric positive definite systems (the matrix A is symmetric if $A = A^T$ and this symmetry property allows us to store only half of the matrix including diagonal entries. Also, matrix A is positive-definite if $z^TAz > 0$ for all non zero vectors z in R^N), Minimal Residual Method (MINRES) for solving sparse symmetric indefinite systems, and Generalized Minimal Residual Method (GMRES) is a general method for nonsymmetric systems [96].

The idea behind the Krylov subspace methods is finding the approximate solution $\tilde{u}_m$ in the subspace $\mathcal{K}_m$. The steps to find the approximate solution of the system (3.28) start from the initial guess u_0 with the initial residual $r_0 = f - Au_o$. So, the approximate solution at the mth step can be defined by the two conditions

$$\tilde{u}_m - u_0 \in \mathcal{K}_m$$

and

$$r_m \perp L_m$$

where L_m is a different subspace of dimension m and $r_m = f - A\tilde{u}_m$. The second condition is called Petrov-Galerkin condition, which states that the residual is orthogonal to a Krylov subspace [96]. The relation between L_m and $\mathcal{K}_m$ gives rise to different Krylov subspace methods.

– If $L_m = \mathcal{K}_m$, and the matrix A is symmetric positive definite then the approximate solution $\tilde{u}_m$ minimizes the A-norm of the error $e_m = u - \tilde{u}_m$ to produce

$$\| u - \tilde{u}_m \|_A = \min_{u_m \in u_0 + \mathcal{K}_m} \| u - u_m \|_A \; .$$

This is the basis of the CG method.

– If $L_m = A\mathcal{K}_m$, in this case, the approximate solution $\tilde{u}_m$ minimizes the residual norm such that

$$\| r_m \|_2 = \min_{p \in \mathcal{P}_m} \| p(A) r_0 \|_2 .$$

GMRES and MINRES methods are based on this concept (see [98]).

The convergence of Krylov sub-space methods depends on the spectral condition number, which for a symmetric matrix is the ratio between the maximum and minimum eigenvalues of A,

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}}.$$

In general, a small condition number gives better convergence. This may be proved for the CG case (see equation (3.29) in the next subsubsection).

In general, Krylov subspace methods are the most efficient iterative methods, and are considerably easier to implement than a sparse direct method. However, the Krylov subspace methods require a good preconditioner to be competitive.

In this work, preconditioned Krylov methods are used to achieve the aim of the research, which is efficient, robust performance and optimal solution time. In Section 3.3, the preconditioning strategies are described in more detail.

3.2.2.3 Conjugate Gradient Method (CG)

The conjugate gradient method (CG) is one of the most effective tools for solving large sparse symmetric positive-definite (SPD) systems. In the past, the CG method was developed as a direct method, but later became one of the most important iterative methods [101].

The convergence of CG in [46, 101] is shown to be bounded by

$$\|u - u_m\|_A \leq 2\|u - u_0\|_A \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^m \tag{3.29}$$

where m=1,2...n. It is apparent from this equation that the convergence of CG depends on κ, the condition number of A. If κ is small, the convergence is fast, while for larger κ, there is a significant decrease in the rate of convergence, requiring a preconditioner to improve the conditioning [101]. For typically finite element discretization of a second order self-adjoint elliptic problem $\kappa = \mathcal{O}(h^{-2})$, where h is the mesh size. Hence the need for preconditioning as the mesh is refined.

3.2.2.4 Generalized Minimal Residual Method(GMRES)

The Generalized Minimal Residual Method (GMRES) is one of Krylov subspace methods for nonsymmetric systems. This method was developed by Saad and Schultz in 1986 [97].

In general, the linear system (3.28) is solved iteratively using the GMRES Algorithm, which is started with a given initial guess, u_0. The GMRES algorithm is based on the Arnoldi process and applies the modified Gram-Schmidt algorithm (see Algorithm 1.2 in [96]) to orthonormalize a vector v_m against $V_m = \{v_1, ..., v_m\}$ for $\mathcal{K}_m(A, r_0)$ [96, 98]. In the GMRES Algorithm, the approximate solution $\tilde{u}_m$ does not get computed explicitly at each step. Furthermore, the GMRES Algorithm 2 keeps the Krylov subspace dimension increasing with each step, up to N, and produces the exact solution in at most N steps. In practice however the method is treated as iterative due to rounding errors, moreover the requirement of the storage and computational cost increases at each iteration [97]. The idea of this Algorithm is solving a least squares problem at each step (line 17). The minimizer y_m is cheap to compute due to solving an $(m+1) \times m$ least-squares problem provided m is small.

Algorithm 2 GMRES without preconditioning

1: Choose an initial guess u_0 and dimension m of Krylov subspace
2: Compute $r_0 = f - Au_0$, $\beta = \| r_0 \|_2$ and $v_1 = r_0/\beta$
3: **for** $j = 1, ..., m$ **do**
4: compute $w_j := Av_j$
5: **for** $i = 1, ..., j$ **do**
6: $h_{ij} := (w_j, v_i)$
7: $w_j := w_j - h_{ij}v_i$
8: **end for**
9: $h_{j+1,j} = \| w_j \|_2$
10: **if** $h_{j+1,j} = 0$ **then**
11: set $m = j$ and go to step 16
12: **else**
13: $v_{j+1} = w_j/h_{j+1,j}$
14: **end if**
15: **end for**
16: set Hessenbreg matrix $H_m = [h_{ij}]_{1 \ll i \ll m+1, 1 \ll j \ll m}$
17: compute $y_m = argmin_y \| \beta e_1 - H_m y \|_2$ and $u_m = u_0 + V_m y_m$ ▷ where $e_1 = [1, 0, 0..]^T$. solving a least squares problem

The convergence theory for GMRES is less precise than for CG, however some convergence bounds for GMRES include [40, 41, 66, 106]. The most straightforward of these is based on eigenvalues of A, where the convergence relies upon the spectral condition number. If the eigenvalues of A spread widely, then the GMRES has slow convergence. Hence, if the eigenvalues of a matrix are clustered into groups, the faster convergence is obtained.

Generally, the GMRES method is used to solve large sparse linear systems that are non-symmetric. However, as mentioned before, it is needs lots of memory and computational time unless it converges in a small number of iterations. To avoid this disadvantage, it is possible

to choose the restarted GMRES (GMRES(m)) algorithm, which has a fixed Krylov subspace dimension, with value m, and restarts of the process with an initial guess $u_0 = u_m$ and a residual $r_0 = r_m$ for the next set m iterations to create a new Krylov subspace of dimension up to m. This process continues restarting until the residual is small enough and converged. The restart parameter m is chosen to be smaller than N to reduce the storage and computational cost requirement. However, restarting leads to slower convergence of the GMRES method. The other choice is to use preconditioned GMRES instead of basic GMRES, which requires less memory and computational cost provided a suitable preconditioner can be find. This method is chosen to be the main method used in this research, preconditioned GMRES is explained in more details in Section 3.3.

3.3 Preconditioning

Generally, a preconditioner P of a non-singular matrix A is chosen such that the preconditioned system is more efficient to solve [45]. It is often desirable that the chosen matrix P has the same features as the matrix A and approximates A. There are two common approaches to choose P:

- Approximate factorisation of A.
- Approximate inverse of A.

In both cases these approximations must be sparse for reasons of efficiency. There are three wayes in which a preconditioner can be applied

(1) Left preconditioning:

$$P_L^{-1}Au = P_L^{-1}f$$

(2) Right preconditioning:

$$AP_R^{-1}y = f, \quad u = P_R^{-1}y$$

(3) Split preconditioning:

$$P_L^{-1}AP_R^{-1}y = P_L^{-1}f, \quad u = P_R^{-1}y$$

where the preconditioner $P = P_L P_R$. The choice of the type of preconditioning depends on which iterative method is used to solve the problem. Overall, a good preconditioner P should be cheap to construct and the linear system $Pz = r$ should be efficient to solve for a given vector r. However, a good preconditioner choice depends upon the problem under consideration [17]. A good preconditioner should provide fast convergence and also be cheap to solve. Moreover, the good preconditioner leads to bounding eigenvalues of AP_R^{-1} clustered into group.

In general, preconditioned Krylov subspace method solve the system $Pz = r$ at each step instead of using AP_R^{-1} explicitly, which is computationally more expensive and may lead to loss of sparsity [17, 96]. So, only matrix-vector products are needed for the GMRES method. Preconditioned GMRES is the main approach used to solve the problem in this research.

Preconditioned GMRES convergence depends on the type of the preconditioning. Generally, the preconditioned GMRES algorithm starts with computing the initial residual $r_0 = f - Au_0$ (see Algorithm 3). In the right preconditioning, the relative residual $r_m = f - Au_m = f - AP_R^{-1}y_m$ is computed, which is different from left preconditioning, which needs to compute the residual in the form $r_m = f - P_L^{-1}Au_m$. The different residuals may effect the stopping criterion and then lead to stop algorithm [96]. The right preconditioning for the GMRES is often used as the residual is unchanged [17].

Algorithm 3 GMRES with right preconditioning

1: Compute $r_0 = f - Au_0$, $\beta = \| r_0 \|_2$ and $v_1 = r_0/\beta$
2: **for** $j = 1, ..., m$ **do**
3: compute $z_j := P^{-1}v_j$
4: compute $w_j := Az_j$
5: **for** $i = 1, ..., j$ **do**
6: $h_{ij} := (w_j, v_i)$
7: $w_j := w_j - h_{ij}v_i$
8: **end for**
9: $h_{j+1,j} = \| w_j \|_2$
10: **if** $h_{j+1,j} = 0$ **then**
11: set $m = j$ and go to step 16
12: **else**
13: $v_{j+1} = w_j/h_{j+1,j}$
14: **end if**
15: **end for**
16: set $H_m = [h_{ij}]_{1 \ll i \ll m+1, 1 \ll j \ll m}$
17: compute $y_m = argmin_y \| \beta e_1 - H_m y \|_2$ and $u_m = u_0 + P^{-1}V_m y_m$

In this research, the systems of algebraic equations arise from the discretization methods, such as those described in Section 3.1, are solved using the right preconditioned GMRES method. The CG method is also used to solve a SPD system, which is explained in detail in Section 6.2.

For the theorems and algorithms of preconditioned Krylov methods, we direct the interested readers to (Saad [96], van der Vorst [108], Elman et al. [40]). Moreover, for excellent reviews papers on the preconditioning techniques, we refer the readers to [17, 115].

The next subsections are used to describe the particular preconditioners, which are used in this work.

3.3.1 ILU Preconditioning

ILU factorization is an incomplete factorization algorithm, which is based upon the direct solution method of LU factorization for nonsymmetric matrices. In the incomplete LU decomposition, we form :

$$A = LU + R$$

where L and U are sparse lower and upper triangular matrices respectively, and R is error matrix. This method computes LU by undertaking an incomplete factorization of the matrix A. The exact factorization has the problem of fill-in, which means that although A is sparse, some entries that are initially zero in L and U may become nonzero entries as part of the complete factorization process. So, L and U are considerably less sparse than A. $ILU(0)$ is Incomplete LU with zero level of fill-in (no-fill), i.e. $L + U$ has the same sparsity pattern as A. $ILU(0)$ inexpensive and easy to implement since it is based on conventional LU decomposition but without allowing any fill-in to occur. However, the preconditioned Krylov subspace solver based on ILU(0) often needs too many iterations to converge due to LU being such a crude approximation of A [96]. To avoid this problem, a drop tolerance strategy is used in order to drop any element in L and U that less than a certain tolerance. ILU preconditioned GMRES with drop tolerance is tested and the results are shown later in this thesis.

Further information about ILU factorization can be found in [96], section 10.3.

3.3.2 Block Preconditioning

In many problems the linear system can be written in block form as

$$\underbrace{\begin{pmatrix} K & C \\ B^T & 0 \end{pmatrix}}_{A} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{3.30}$$

where $K \in \mathbb{R}^{n\times n}$ and $B, C \in \mathbb{R}^{n\times m}$. This equation called a saddle point system arising in problems such as the Taylor-Hood discretization of Stokes equation for example. The coefficient matrix A is indefinite. Such a problem can be solved iteratively using MINRES (e.g [86, 114]) or GMRES (e.g [97]) in case the matrix A is symmetric (if $B = C$) or nonsymmetric (if $B \neq C$) respectively. This research focuses on the nonsymmetric case.

The coefficient matrix A may preconditioned by

$$P1 = \begin{pmatrix} K & 0 \\ 0 & S \end{pmatrix}; \tag{3.31}$$

this is a block diagonal preconditioner [79], where $S = B^T K^{-1} C$, is the Schur complement. This form of preconditioner leads to three distinct eigenvalues [40, 79, 114], which are obtained from solving:

$$\begin{pmatrix} K & C \\ B^T & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \lambda \begin{pmatrix} K & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix};$$

this satisfies: $\lambda = 1$, $\lambda = \frac{1+\sqrt{5}}{2}$ and $\lambda = \frac{1-\sqrt{5}}{2}$ for a nonsingular matrix (see proof in [79]).

The second preconditioner, in the nonsymmetric case, takes the form:

$$P2 = \begin{pmatrix} K & C \\ 0 & \pm S \end{pmatrix}, \tag{3.32}$$

which is a block triangular preconditioner with $S = B^T K^{-1} C$; this form of preconditioner leads to two distinct eigenvalues or only one distinct eigenvalue depending on which the sign of S is selected. So from:

$$\begin{pmatrix} K & C \\ B^T & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \lambda \begin{pmatrix} K & C \\ 0 & \pm S \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix}$$

that lead to $\lambda = \pm 1$ if the + sign is chosen or lead to $\lambda = 1$ if the - sign is chosen [79, 115]. In the both cases, the Krylov method with this preconditioning requires only two iterations to converge.

The advantage of the preconditioner with exact Schur complement matrix is the convergence rate of Krylov subspace is bounded independently of the problem size and discretization parameters. However, the exact Schur complement involves an inverse matrix and that may be expensive to compute and require lots of memory and storage.

To improve the previous preconditioners, the block matrix K approximated is by $\hat{K}$, and S is replaced by the approximate Schur complement, which is the pressure mass matrix (mp). $\hat{S}$ may be equivalent to mp or $diag(mp)$. The new preconditioners

$$P3 = \begin{pmatrix} \hat{K} & 0 \\ 0 & mp \end{pmatrix}, \tag{3.33}$$

$$P4 = \begin{pmatrix} \hat{K} & C \\ 0 & mp \end{pmatrix}, \tag{3.34}$$

P3 and P4 require less memory and time to converge, however, they need more Krylov iterations to converge [114].

In Wathen's paper [114], the properties of the matrix A is symmetric indefinite. He improved the diagonal preconditioning in (3.33) to get the optimal solution time. A multigrid method with one V-cycle and one pre-and post smoothing stage is used to solve the block $\hat{K}$, and Conjugate Gradients is used to solve mp. The preconditioned MINRES method was used and achieved the best solver in Stokes equation case.

Based on the ideas on Wathen's paper [114], Murphy et al. [79] and Elman et al. [40], we improve a novel block triangular preconditioner for the nonsymmetric linear systems, which is

described in Section 6.2.

3.4 Solution of Nonlinear Systems

Application of FEM to nonlinear PDEs results in a system of nonlinear algebraic equations. The most common solution approach is to linearise the system using Newton's method.

3.4.1 Newton's Method

The Newton's method is a process for finding the solution of a nonlinear system $F(u) = 0$ for a vector, $u = \{u_1, u_2, .., u_n\}$, of n unknown values, where F is a set $\{F_1, F_2, ..., F_n\}$ of n nonlinear equations [65].

The Newton iteration is defined as

$$u_{k+1} = u_k - \left(\frac{\partial F(u_k)}{\partial u}\right)^{-1} F(u_k), \quad for \;\; k = 0, 1, 2, .. \tag{3.35}$$

where k is the nonlinear iteration index, u_k is a known approximation to solution.

The $n \times n$ matrix is called the Jacobian, J, where $J_{ij} = \frac{\partial F_i}{\partial u_j}$. So we can write the Newton method as

$$u_{k+1} = u_k - \left(J_k\right)^{-1} F(u_k), \tag{3.36}$$

Generally, we can achieve this by solving the linear system

$$J\delta = -F$$

and updating $u_{k+1} = u_k + \delta$. However, solving this linear system directly at each Newton iteration is very expensive in general.

The convergence of Newton method is determined based on a required drop in the nonlinear residual 2-norm [67]

$$\frac{\parallel F(u_k) \parallel_2}{\parallel F(u_0) \parallel_2} < tol \tag{3.37}$$

where u_0 is an initial guess and tol is a given tolerance. As mentioned above, solving the linear system directly at each Newton iteration is very expensive. To increase efficiency an alternative method is needed to solve this system. The alternative method is covered in the next subsection. We direct the reader to [65] for further detail about Newton's method

3.4.2 Newton-Krylov Method

Newton-Krylov methods are combinations of Newton methods, for linearization of nonlinear equations, and iterative Krylov subspace methods for solving the linear system. The Jacobian-vector product is the link between both methods since the most expensive step of each iteration

of a Krylov solver is the execution of a single matrix-vector product. In Jacobian-free methods this product is computed approximately without forming and storing the true Jacobian elements. Various approximations to the Jacobian matrix may be still required in order to precondition the resulting Krylov iteration.

In Newton Krylov methods (see e.g [29, 64]), the nonlinear iterative method is the outer iteration, while the Krylov based linear iterative method is the inner iteration. Usually, the outer iteration is an inexact Newton, and strict quadratic convergence is not achieved, only asymptotic quadratic convergence, which is achievable with sufficient effort on the inner iteration.

A preconditioned Newton-Krylov method is applied in this work to linearise the nonlinear algebraic system. In particular, AMG preconditioned GMRES is used for solving the linear system $J\delta = -F$ optimally (for further information about AMG preconditioner see Subsection 3.2.2.1).

3.5 Summary

This chapter was organised to cover two main scientific computing tools that are related to our work.

- Discretization methods for PDEs focused on Finite Element Method (FEM) and Finite Volume Method (FVM).
- The solution of linear and nonlinear algebraic systems is described in detail. Right block triangular preconditioned GMRES is chosen to solve the linear systems, and AMG preconditioned GMRES method is chosen to solve the linearised systems arising from using Newton's method for solving the nonlinear systems.

More information about the new preconditioner and the numerical results are shown in the following Chapters.

Chapter 4

Numerical Approximation of the Multiphase Tumour Growth

In the previous chapter, we introduced a general description of discretization methods and corresponding solution methods for the resulting discrete systems. This chapter presents reviews of some existing numerical models of tumour growth that are presented in Section 4.1. The numerical schemes for the mathematical model, which is discussed in Chapter 2 Section 2.2, are introduced in Section 4.2. The details of the numerical experiments for the whole model, including the parameters values, the initial and boundary conditions and key implementation details are presented in Section 4.3.

4.1 Numerical Models of Tumour Growth

Generally, as stated previously, there are three classical choices for mesh-based discretizations of PDEs which are the FDM, the FEM and the FVM. In Section 2.1 we described some of the research related to the multiphase modelling of tumour growth. In this section we describe numerical models for the solution of multiphase systems arising in a range of applications as well as other numerical models for tumour growth.

There are many examples of the use of numerical models for multiphase flow. In [78], Mudde and Simonin used the multiphase flow models relevant to bubbly flows (bubble plume) and implemented a code based on FVM for the space discretization of the two-fluid equations on two and three dimensional structured grids. As another example, in porous media systems, the authors in [63] developed a two dimensional FEM model to predict simultaneous flow of a three-fluid phase system which are water, oil and gas, which is assumed to be at constant

pressure.

In Breward et al. [26,27] numerical simulations are presented in one dimension using a FDM scheme on structured meshes. Also, there is some research using the finite difference method, in multiple space dimensions, for simulating tumour growth, for example in [53,73]. In [84,85] the finite element method is used on unstructured meshes, which can be designed for a particular geometry.

In [99] a computational model is used to solve three cases of biological relevance. The governing equations are solved by a FEM to predict the growth rate of the mass of tumour as a function of the initial ratio of tumour density to density of healthy cells, mechanical strain, concentration of nutrient, cell adhesion and geometry. This model allows straightforward insertion of additional extra phases and nutrient types.

In [57] the computational approach is a combination of FEM and FVM. In this two dimensional model, the hyperbolic PDEs of mass balance are discretised through a conservative, upwind, FVM scheme, whereas the momentum balance equations lead to generalised Stokes equations solved using a FEM scheme. Additionally, the discretization of nutrient reaction-diffusion equations is also based upon a finite element method used with a Newton iteration to solve the nonlinear equations. This numerical model demonstrates that the framework can be applied in multiple dimensions. In fact, extensions to more phases and 3D are claimed to be straightforward if the storage and time requirements are sufficient.

4.2 Numerical Schemes for Chosen Model

In this section, the numerical solution schemes are described for the mathematical model introduced in Section 2.2. In [57] the computational approach is a combination of FEM and FVM on unstructured triangular meshes. In the following subsections, the numerical schemes for mass balance equations, momentum balance equations and reaction diffusion equation are described.

4.2.1 Mass Balance Equations

Equation (2.2) is a set of hyperbolic mass balance equations which are time dependent PDEs. These equations are approximated using an explicit solver in time with a standard cell centred finite volume scheme with forward Euler time stepping. The integral form for the mass balance equation is given by

$$\int_{\Delta} \frac{\partial \theta_i}{\partial t} d\vec{x} + \int_{\Delta} \vec{\nabla}.(\theta_i \vec{u}_i) d\vec{x} = \int_{\Delta} q_i d\vec{x} \qquad i = 1, ..., 3. \tag{4.1}$$

The discrete system of equations is designed to update the unknown θ_i, which are the cell-average values. By using the Gauss divergence theorem for the flux integrals in the previous equation we obtain

$$\int_{\Delta} \frac{\partial \theta_i}{\partial t}\, d\vec{x} + \oint_{\partial\Delta} (\theta_i \vec{u}_i).\vec{n}\; ds = \int_{\Delta} q_i d\vec{x} \qquad i = 1, ..., 3, \tag{4.2}$$

where Δ is the control volume, q_i represents the source/sink terms, $\partial\Delta$ is the boundary of the control volume, and $\vec{n}$ is the outward-pointing unit normal to the boundary. In [57] a cell-centred MUSCL approach is used, which is a conservative, upwind, finite volume scheme. This is used to update θ_1, θ_2, and θ_3 in time, while the no-voids condition (2.3) is used to update θ_4. The fluxes $\theta_i \vec{u}_i$ are approximated by using a standard upwind scheme [71]. So, the discrete equation for each triangle can be written as

$$\overline{\theta}_i^{n+1} = \overline{\theta}_i^{n} - \frac{\Delta t}{\mid \Delta \mid} \sum_{k=1}^{3} (\theta_i^n \vec{u_i^n})_k^*.\vec{n_k} + \Delta t (q_i^n)^*, i = 1, ..., 4. \tag{4.3}$$

in which n denotes the time level, Δt is the time step size, $\mid \Delta \mid$ is the control volume area, and $\vec{n_k}$ is the outward-pointing unit normal to the edge of the cell opposite the vertex k. The asterisk (*) is used to distinguish the quantities, which have been approximated depending on the variables $\bar{\theta_i}$, which is a cell average.

To define the boundary condition, the volume fraction θ_i is described for each phase on the inflow section Γ_i^{inflow} for $t \geq 0$. These inflow sections belong to the domain boundary Γ where $\vec{u_i}.\vec{n} < 0$, where $\vec{n}$ is the outward-pointing unit normal to Γ. In this model, the boundary condition is only required for the inflow section of the boundary.

4.2.2 Momentum Balance Equations

The momentum balance equations (2.4), do not depend on time, and they are linear in $\vec{u_i}$ and p_4. The weak form is obtained by applying a Galerkin finite element scheme with Taylor-Hood elements (see Section 3.1.1) as

$$\int_{\Omega} \omega^q \vec{\nabla}.(\theta_i \sigma_i) d\vec{x} + \int_{\Omega} \omega^q \vec{F_i} d\vec{x} = 0 \quad i = 1, ..., 4,$$

$$\int_{\Omega} \omega^l \sum_{i=1}^{4} \vec{\nabla}.\ (\theta_i \vec{u}_i)\ d\vec{x} = 0 \tag{4.4}$$

where ω^l and ω^q are the standard linear and quadratic Lagrange test functions respectively, σ_i are the stresses in each individual phase

$$\sigma_i = -p_i \mathbf{I} + \mu_i (\vec{\nabla}\vec{u}_i + (\vec{\nabla}\vec{u}_i)^T) + \lambda_i (\vec{\nabla}.\vec{u}_i)\mathbf{I},$$

and $\vec{F_i}$ are the momentum sources

$$\vec{F_i} = p_i \mathbf{I} \vec{\nabla} \theta_i + \sum_{j=1, j\neq i}^{4} d_{ij} \theta_i \theta_j (\vec{u}_j - \vec{u}_i) \quad i = 1, ..., 4$$

Using integration by parts for the first equation from (4.4) leads to

$$\oint_{\partial\Omega} \omega^q \theta_i \sigma_i . \vec{n}\ ds - \int_\Omega \vec{\nabla}\omega^q . \theta_i \sigma_i d\vec{x} + \int_\Omega \omega^q \vec{F}_i d\vec{x} = 0 \quad i = 1, ..., 4. \tag{4.5}$$

The velocities $\vec{u_i}$ and pressures p are approximated by using piecewise polynomials, which are written using the trial function as

$$\vec{u_i} \approx \sum_{k=1}^{N_q} (\vec{u_i})_k \omega_k^q, \qquad p \approx \sum_{k=1}^{N_l} p_k \omega_k^l, \tag{4.6}$$

in which N_q and N_l are, respectively, the numbers of degrees of freedom related to the quadratic and linear Lagrange elements.

There are two possible boundary conditions used for the momentum balance equations, which are Dirichlet condition on $\vec{u_i}$ or $\sigma_i . \vec{n}$ specified. Here, the boundary conditions imposed are zero normal stress $\sigma_i . \vec{n} = 0$ for the three phases of healthy cells, tumour cells and blood vessels, and zero velocity for the extracellular phase $\vec{u}_4 = 0$.

These choices of boundary conditions for the normal stress lead to the boundary term in (4.5) becoming zero, and so (4.4) may be expressed as:

$$\begin{aligned} -\textstyle\int_\Omega \vec{\nabla}\omega^q . \theta_i \sigma_i d\vec{x} + \int_\Omega \omega^q \vec{F}_i d\vec{x} &= 0, \quad i = 1, ..., 4. \\ \textstyle\int_\Omega \omega^l \Sigma_{i=1}^4 \vec{\nabla} .\ (\theta_i u_i)\ d\vec{x} &= 0. \end{aligned} \tag{4.7}$$

Applying integration by parts to the continuity equation, which is the second equation in (4.7), and using (4.6) gives:

$$\begin{aligned} &\textstyle\sum_{i=1}^4 \sum_{k=1}^{N_q} \int_\Omega [\omega_m^l \theta_i \dfrac{\partial \omega_k^q}{\partial x} + \omega_m^l \omega_k^q \dfrac{\partial \theta_i}{\partial x}](u_i)_k d\vec{x} + \\ &\textstyle\sum_{i=1}^4 \sum_{k=1}^{N_q} \int_\Omega [\omega_m^l \theta_i \dfrac{\partial \omega_k^q}{\partial y} + \omega_m^l \omega_k^q \dfrac{\partial \theta_i}{\partial y}](v_i)_k d\vec{x} = 0 \end{aligned} \tag{4.8}$$

where $m = 1, ..., N_l$. From the definition of σ_i, the first equation in (4.7) lead to:

$$\begin{aligned} \textstyle\sum_{k=1}^{N_q} \int_\Omega \big\{ \big[\mu_i \theta_i (2 \frac{\partial \omega_k^q}{\partial x} \frac{\partial \omega_m^q}{\partial x} &+ \frac{\partial \omega_k^q}{\partial y} \frac{\partial \omega_m^q}{\partial y}) + \lambda_i \theta_i \frac{\partial \omega_k^q}{\partial x} \frac{\partial \omega_m^q}{\partial x} + \omega_k^q \omega_m^q d_{ij} \theta_i \theta_j \big](u_i)_k \\ &- \big[\omega_k^q \omega_m^q d_{ij} \theta_i \theta_j \big](u_j)_k \\ &+ \big[\mu_i \theta_i \frac{\partial \omega_k^q}{\partial y} \frac{\partial \omega_m^q}{\partial x} + \lambda_i \theta_i \frac{d\omega_k^q}{dx} \frac{d\omega_m^q}{dy} \big](v_i)_k \big\} d\vec{x} \\ &- \textstyle\sum_{k=1}^{N_l} \int_\Omega \big\{ [\omega_k^l \theta_i \frac{\partial \omega_m^q}{\partial x} - \omega_k^l \omega_m^q \frac{\partial \theta_i}{\partial x}](p_i)_k \big\} d\vec{x} = \int_\Omega \omega^q \vec{F}_{ix} d\vec{x}, \end{aligned} \tag{4.9}$$

$$\begin{aligned}\sum_{k=1}^{N_q}\int_\Omega \Big\{\big[\mu_i\theta_i(\tfrac{\partial\omega_k^q}{\partial x}\tfrac{\partial\omega_m^q}{dx} &+ 2\tfrac{\partial\omega_k^q}{\partial y}\tfrac{\partial\omega_m^q}{\partial y}) + \lambda_i\theta_i\tfrac{\partial\omega_k^q}{\partial y}\tfrac{\partial\omega_m^q}{\partial y} + \omega_k^q\omega_m^q d_{ij}\theta_i\theta_j\big](v_i)_k \\ &- \big[\omega_k^q\omega_m^q d_{ij}\theta_i\theta_j\big](v_j)_k \\ &+ \big[\mu_i\theta_i\tfrac{\partial\omega_k^q}{\partial x}\tfrac{\partial\omega_m^q}{\partial y} + \lambda_i\theta_i\tfrac{\partial\omega_k^q}{\partial y}\tfrac{\partial\omega_m^q}{\partial x}\big](u_i)_k\Big\}d\vec{x} \\ &- \sum_{k=1}^{N_l}\int_\Omega\Big\{\big[\omega_k^l\theta_i\tfrac{\partial\omega_m^q}{\partial y} - \omega_k^l\omega_m^q\tfrac{\partial\theta_i}{\partial y}\big](p_i)_k\Big\}d\vec{x} = \int_\Omega \omega^q\vec{F}_i y d\vec{x},\end{aligned} \tag{4.10}$$

where $m = 1, ..., N_q$, $i \neq j$ and $i, j = 1, ..., 4$. The structure of this system of equations is described in greater detail in Chapter 6.

In [57] the resulting large sparse system of linear equations is solved using the direct solution package MUMPS. The resulting system is nonsymmetric and indefinite and has similarities with the standard arising from a mixed FEM solution of Stokes' equation for incompressible flow. One of the contributions of this thesis is the use of a novel preconditioned GMRES method instead of MUMPS in this problem. More details for new solver are shown in Chapter 6

4.2.3 Reaction Diffusion Equation

Equation (2.6) is the quasi/steady state, reaction-diffusion equation. This equation is a time-independent PDE, and nonlinear, and is approximated using a standard Galerkin finite element scheme with linear elements to get:

$$\int_\Omega \omega^{l_f} D_c\vec{\nabla}^2 c \; d\vec{x} - \int_\Omega \omega^{l_f} q_c d\vec{x} = 0, \tag{4.11}$$

in which D_c denotes to the diffusion coefficient, which is assumed to be constant, ω^{l_f} is a linear test function defined on the finer mesh, which is used to update the volume fractions of the phases, and q_c is a source term, which is a nonlinear function of the nutrient concentration c.

Using integration by parts leads to

$$\int_{\partial\Omega} \omega^{l_f} D_c\vec{\nabla}c.\vec{n}ds - \int_\Omega \vec{\nabla}\omega^{l_f}.D_c\vec{\nabla}c d\vec{x} - \int_\Omega \omega^{l_f} q_c d\vec{x} = 0, \tag{4.12}$$

The nutrient concentration c is written using the trial functions as:

$$c \approx \sum_{k=1}^{N_{l_f}} c_k\omega_k^{l_f}. \tag{4.13}$$

There are two possible boundary conditions used for the reaction diffusion equation, which are Dirichlet conditions c or Neumann conditions $\vec{\nabla}c.\vec{n}$. In this work $\vec{\nabla}c.\vec{n} = 0$ is specified in all cases throughout the boundary.

Here, FEM leads to a nonlinear system of equations with a sparse Jacobian which is solved using a Newton method to linearise it. The MUMPS sparse direct algorithm is again used to solve the linear system of equations at each nonlinear iteration in [57]. In this thesis, AMG

preconditioned GMRES method is used to solve the linear system instead of MUMPS direct solver. Further details are described in Chapter 5.

4.2.4 Meshes for Discretization

In this work two different triangular meshes are used for each solve: (i) the original mesh, on which the momentum equations are solved; and (ii) a uniform refinement of this original mesh, on which the mass equations and the reaction-diffusion equation are solved. Because we use the Taylor-Hood FEM scheme to approximate the momentum equations, each edge of this original mesh has velocity unknowns at its midpoint due to the quadratic Lagrange polynomial basis functions used for the velocity unknowns. Consequently, the vertices of the uniformly refined mesh correspond to the locations of the velocity degree of freedom: hence the mass and concentration unknowns on the refined mesh correspond to the velocity unknowns [57].

4.3 Test Problem Using the Mathematical Model

The current model under consideration in [57] is limited by the sub-optimal algorithms used to solve the linear and nonlinear algebraic systems resulting from the FEM approximations. The use of a sparse direct solver on a single processor machine has limited the mesh resolution and the computational domain size that can be applied to simulations. Sparse direct methods are expensive for large systems in CPU time and also in memory requirements. In practice solving in three dimensions is not feasible unless a more efficient solver is developed. Even increasing the mesh resolution or the number of phases present in the 2D model would add significant extra memory and CPU overhead. Before proposing improvements however we summarize how the model is simulated in [57]:

- Each phase has volume fraction θ_i, velocities u_i' and v_i', and pressure p_i' where $i = 1, ..., 4$.
- Find the volume fraction θ_i for $i = 1, 2, 3$, using mass conservation, then using the no-voids condition (2.3), so, $\theta_4 = 1 - \sum_{i=1}^{3} \theta_i$.
- For the linear system arising from discretization of momentum balance:
  - 4 (u,v) unknowns and 1 p unknown must be found across the domain at each time step.
  - Taylor-Hood FEM is used, due to this the discrete system is quadratic in the velocities (u,v) and linear in pressure p.
  - We calculate the pressure for extracellular material phase p_4' only, due to $p_1' = p_2' = p_4' + \Sigma'(\theta)$ and $p_3' = 0$ constant, in which

$$\Sigma'(\theta) = \begin{cases} \frac{(\theta - \theta^*)}{(1-\theta)^2} & \text{if } \theta \geq \theta^* \\ 0 & \text{if } \theta < \theta^*, \end{cases} \tag{4.14}$$

where $\theta = \theta_1 + \theta_2$ and θ^* is the cell's natural density.

- Use sparse direct linear algebra to solve the discrete algebraic system.

- For the nonlinear system arising from discretization of the reaction-diffusion system:

  - the nutrient concentration c' is the only unknown.
  - the linear FEM is used to approximate the system.
  - we linearise the reaction-diffusion system using Newton's method.
  - use sparse direct linear algebra at each Newton iteration.

The main contribution of the research in this thesis is improving the efficiency of the numerical methods used in order to reach almost optimal efficiency. We achieve this by replacing the sparse direct solver (MUMPS) with more efficient iterative solvers.

The computational model in [57] is used to simulate tumour growth by seeding a small number of tumour cells in a healthy tissue in an equilibrium state. [57] presents three different ways to simulate the growth of tumour cells. Firstly, a single cluster of the tumour cells seeded in the centre of the computational domain, which is circular and based upon an unstructured triangular mesh in two-dimensions. Secondly, two clusters of tumour are seeded in the centre of the computational domain, and finally, a single cluster of tumour is seeded in a tapered domain. Furthermore, they studied the effect of varying parameter values on the behaviour of the growth of the tumour. This research focuses on the first type of the tumour growth simulation for validation, however the methods developed are applicable in all cases. The values of parameters that are used in this work, which were used in [57], are shown in Table 4.1.

In all the experiments we used the mathematical and computational models that are written in FORTRAN language by Hubbard and Byrne [57]. In addition, we make use of the software implementation of iterative methods that is available in Harwell Subroutine Library (HSL) [1]. This includes: HSL-MI20 for AMG preconditioner, HSL-MI21 for CG method and HSL-MI24 for GMRES method. We also make use of SPARSKIT for GMRES method and the ILU preconditioner [95] that is used for comparison. The numerical experiments presented in this work have been carried out on a standard desktop machine with 16 GB of memory. Furthermore, some preliminary numerical results, which are presented in the next chapter, are obtained using MATLAB codes.

Table 4.1: The nondimensional parameters values are taken from [57]

Parameter	Symbol	Values
Birth rate of tumour cell	$k^*_{1,2}$	2.0
Death rate of healthy cell	$k^*_{2,1}$	0.15
Death rate of tumour cell	$k^*_{2,2}$	0.075
Vessel occlusion rate	k^*_3	0.1
Angiogenesis rate	k^*_4	0.0029449
Healthy cell baseline (nutrient consumption rate)	$k^*_{6,1}$	0.01
Tumour cell baseline (nutrient consumption rate)	$k^*_{6,2}$	0.01
Healthy cell birth (nutrient consumption rate)	$k^*_{7,1}$	0.1
Tumour cell birth (nutrient consumption rate)	$k^*_{7,2}$	0.2
Cell birth rate dependence on nutrient	c^*_p	0.25
Cell death rate dependence on nutrient	c^*_{c1}, c^*_{c2}	0.2,0.1
Angiogenesis rate dependence on nutrient	c^*_a	0.05
Critical pressure for vessel occlusion	p^*_{crit}	0.3
Smoothness of occlusion pressure	ϵ^*_3	0.2
Angiogenesis rate	ϵ	0.01
Cell tension constant	Λ^*	0.1
Phase dynamic shear viscosities	μ^*_i	10.0
Phase bulk viscosities	λ^*_i	$-\frac{2}{3}\mu^*_i$
Interphase drag coefficients	d^*_{ij}	1.0
Nutrient diffusion coefficient	D^*_c	1.0

4.3.1 Initial and Boundary Conditions

In this subsection we introduce the initial and boundary conditions that are used in our experiments.

The two-dimensional domains Ω are defined by $[-16, 16]$ for square domains, or a circle of radius 16. The initial conditions for simulation of a single tumour seeded in the centre of healthy tissue can be giving as following

–

$$\theta_2(x, y, t = 0) = \begin{cases} 0.05 \quad cos^2(\frac{\pi r}{2}) \quad for \quad r \lesssim 1 \\ 0 \qquad\qquad\qquad otherwise \end{cases} \tag{4.15}$$

in which $r = \sqrt{x^2 + y^2}$.

– $\theta_1(x, y, 0) = 0.6 - \theta_2(x, y, 0)$.

– $\theta_3 = 0.0174978$.

– $\theta_4 = 0.3825022$.

– Each phase has zero velocity.

– $p_3 = 0$ and $p_4 = 0$, then $p_1 = p_2 = 0$ due to $\theta_1 = \theta^*$ in equation (4.14).

– The nutrient concentration $c' = 0.2532031$ everywhere.

The boundary conditions are chosen as

– the boundary conditions for the system of equations (2.2) are imposed on the inflow section of the boundary Γ_i^{inflow} on which $\vec{u}_i'.\vec{n} < 0$:

$$\theta_i = \theta_i^{\infty}, \quad i = 1, .., 4.$$

– the boundary conditions for the system of equations (2.4) are imposed on the whole of the boundary Γ

$$\sigma_i'.\vec{n} = 0, \;\; i = 1, .., 3 \quad and \qquad \vec{u}_4' = 0,$$

where σ_i' are the stresses in each individual phase.

– the boundary condition for the system of equation (2.6) is imposed on the whole of the boundary Γ

$$\vec{\nabla}' c'.\vec{n} = 0.$$

where $\vec{n}$ is the unit outward-pointing normal to Γ.

4.3.2 Review of Hubbard and Byrne Results

In this subsection a selection the original results from [57] are presented. The domain of the numerical simulation was a circular domain of radius 16 with the number of nodes 2349. We have re-computed these results ourselves but they correspond precisely to cases considered in [57].

The numerical simulation in Figure 4.1 (which is Fig 5 in [57]) shows how the volume fraction of each phase develops with the time on the circle domain. Initially, the simulation starts with a seed cluster of tumour cells in the middle of healthy tissue (not shown). This cluster of tumour extends over time to produce a high tumour cell density around the initial seed. The volume fraction of tumour cells grows rapidly and reaches a maximum value, which is in the range between 0.8 and 0.9. Then the tumour cells spread out and the die in the centre, forming a necrotic core due to the local nutrient concentration being insufficient to maintain the tumour cells. It can be noted that the tumour has an imperfect circle shape, due to the unstructured grid causing the radial symmetry to be lost.

Figure 4.2, Figure 4.3 and Figure 4.4 (which are Fig 6, Fig 7 and Fig 8 in [57]) illustrate the evolutions of the phase fluxes, pressures and the nutrient concentration, which also generate the characteristic tumour growth pattern.

The proliferation and death rates of the tumour cells are assumed to be double and half the values of the proliferation and death rates of the healthy cells respectively (see Table 4.1). So, the tumour cells grow faster and die slower than the healthy cells. In this case, the tumour cells absorb the extracellular material during their growth. This leads to a fall in the extracellular material θ_4 and therefore leads to decreasing the healthy cells' birth rate. Also, when the tumour cells grow faster than the healthy cells, that leads to the tumour cells pushing the healthy cells in front and replacing the volume by the extracellular material (as illustrated in Figure 4.2). Furthermore, the high tumour cell density generates high pressures which leads to the occlusion of the blood vessels and therefore restricts the supply of the nutrient inside the tumour (cf. Figure 4.4). This exacerbates the problem and makes the imbalance between the proliferation and death for the healthy cells increase.

The purpose of this subsection has been to provide some further insight into the model of [57], which we are studying here. In Chapter 7 we present further numerical results as part of the validation and assessment of our improvements to the linear algebra routines used in the model.

Figure 4.1: Evolution of the volume fraction for each phase arranged from the top row to the bottom row: healthy cells θ_1, tumour cells θ_2, blood vessels θ_3 and extracellular material θ_4, with increasing time from left to right, t=100, 200 and 300.

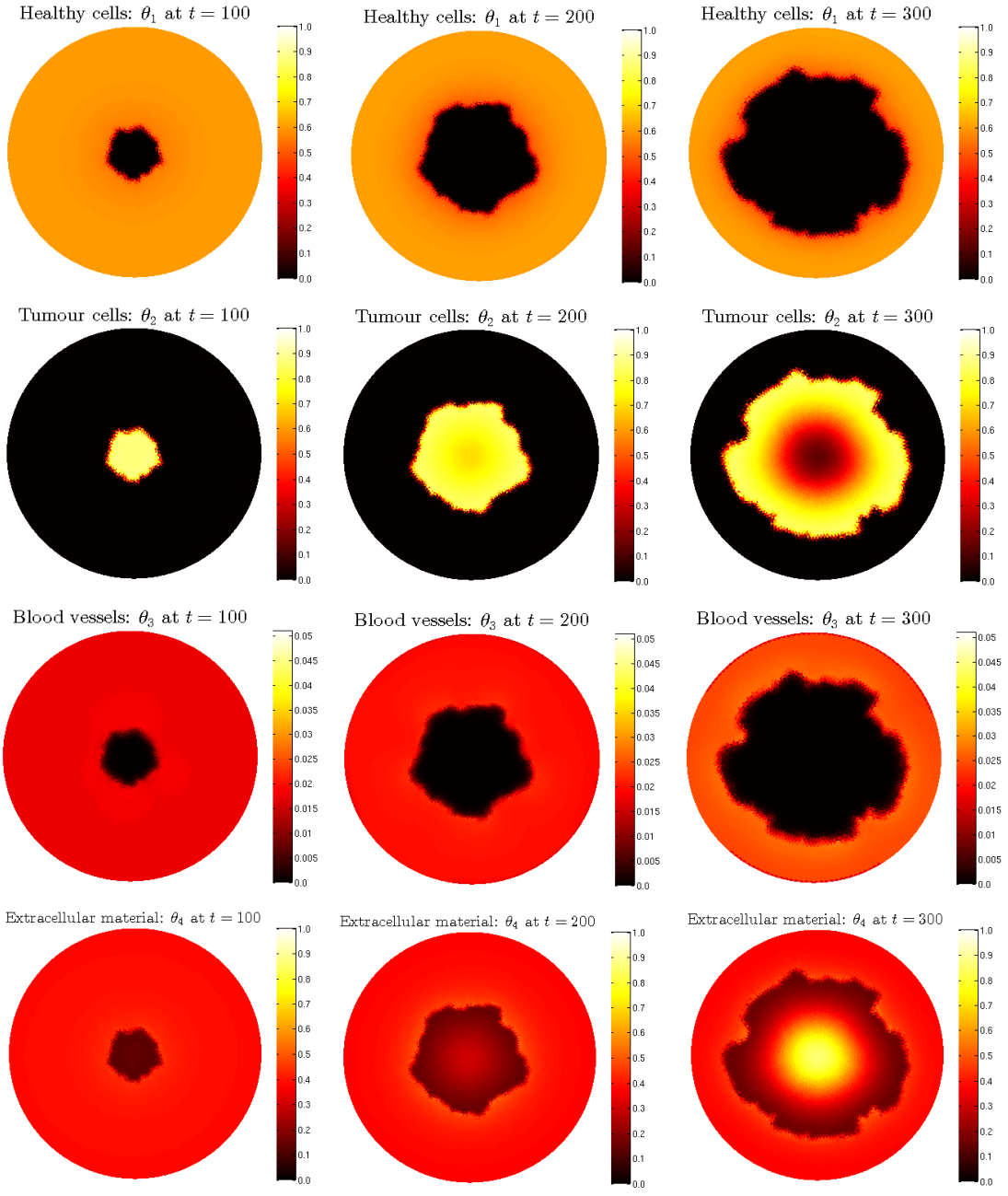


Figure 4.2: Evolution of the fluxes of phases, arranged from the top row to the bottom row: healthy cells $\theta 1\vec{u}_1'$, tumour cells $\theta_2\vec{u}_2'$ and extracellular material $\theta_4\vec{u}_4'$, with increasing the time from left to right, t=100, 200 and 300.

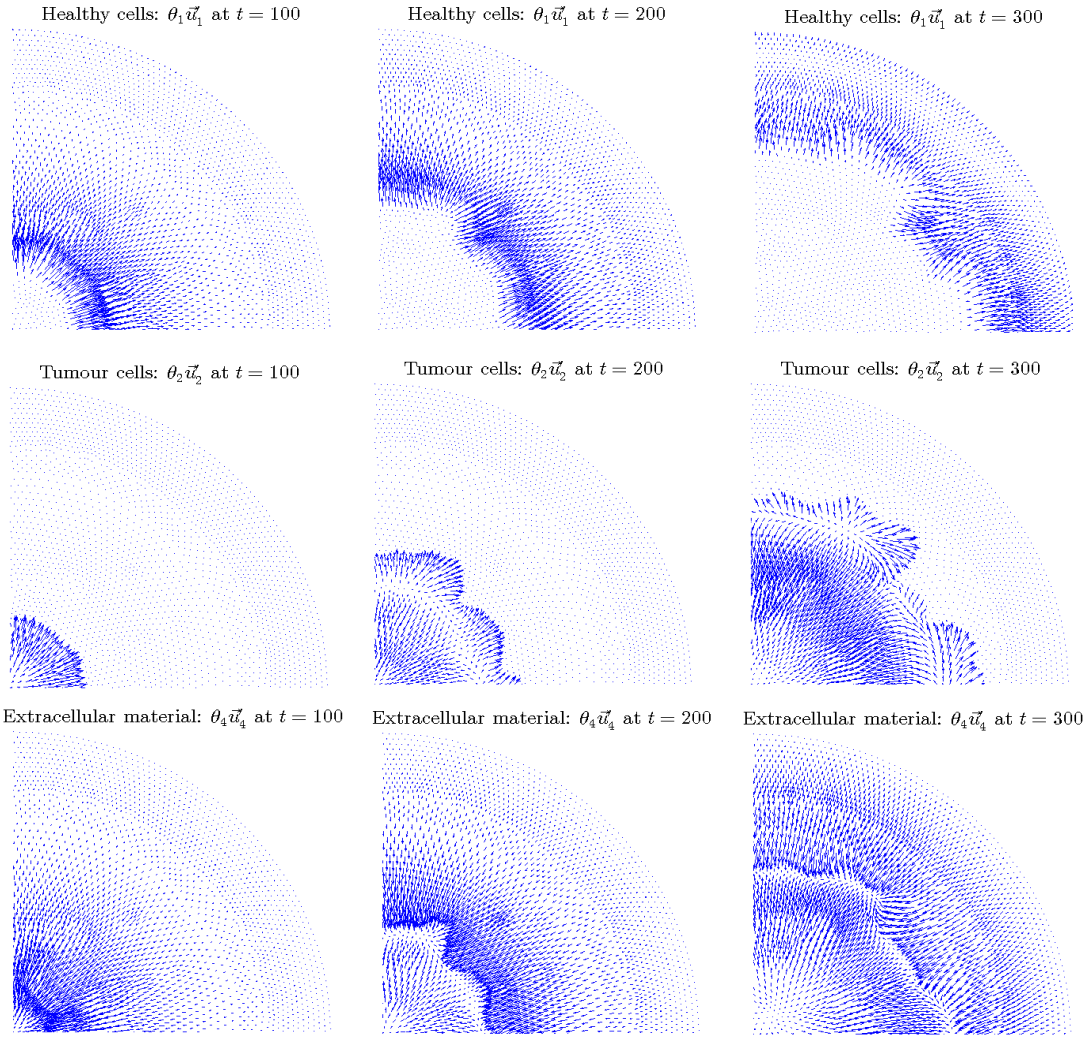


Figure 4.3: Evolution of the pressures arranged from the top row to the bottom row: healthy and tumour cells $p_1 = p_2$ and extracellular material (ECM) p_4, with increasing time from left to right, t=100, 200 and 300. The blood vessels pressure p_3 is zero in this model.

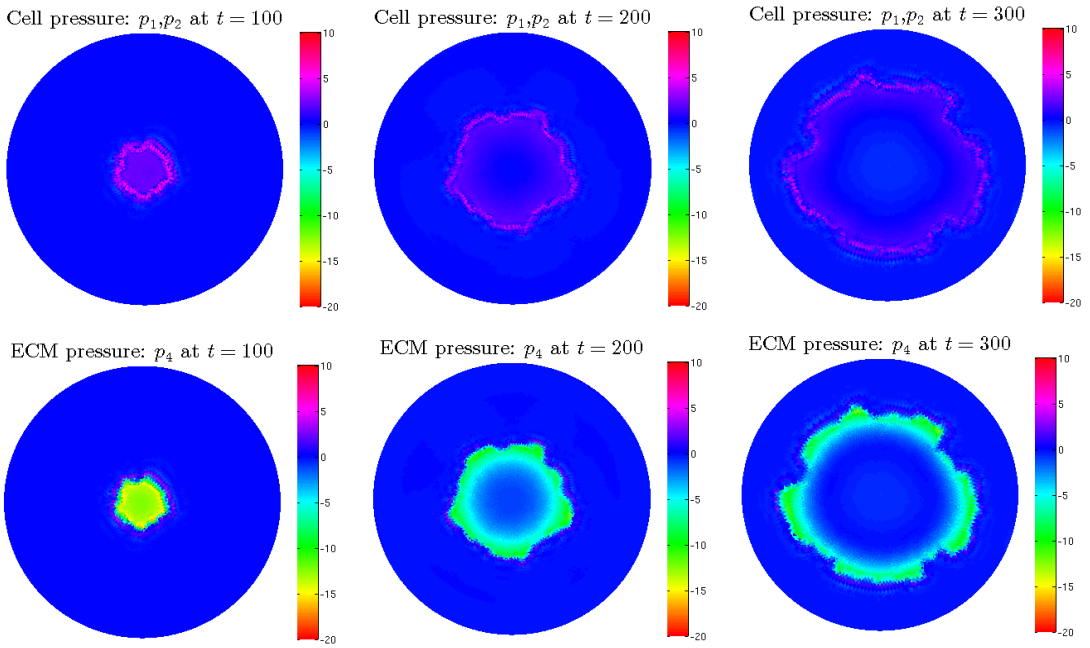
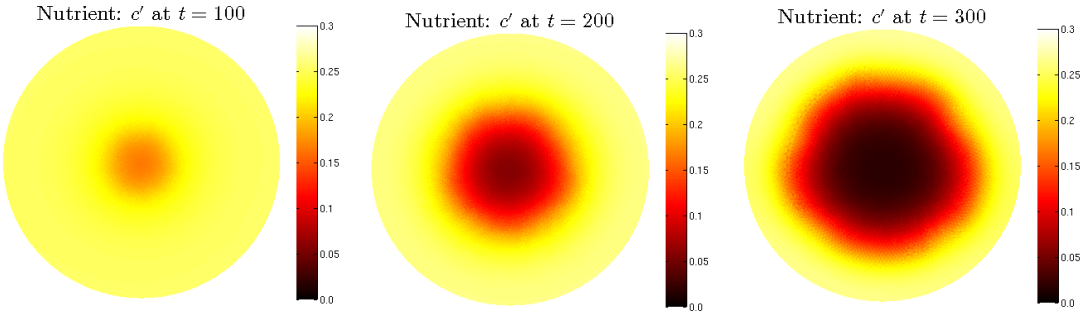


Figure 4.4: Evolution of the nutrient concentration c with increasing time from left to right, t=100, 200 and 300.



4.4 Summary

This chapter provides a brief overview of some numerical schemes that have been used to solve multiphase problems, including some multiphase models of tumour growth. It then describes in more detail the discretisation schemes used for the particular multiphase PDE model of tumour growth that is studied in this work, [57]. In summary, at each time step:

- The cell-centred finite volume scheme is used to approximate the mass balance equations to update the cell-average θ_i for three phases while the no-void condition is used to update the fourth phase.
- The Galerkin finite element scheme with Taylor-Hood elements is used to approximate the momentum balance equations to solve for new $\vec{u}_i$ and p_i,
- The standard Galerkin finite element scheme with linear elements is used to approximate the reaction diffusion equation to update the nutrient c.

We then illustrate the behaviour of some typical simulations using this model, for given parameter, boundary condition and initial condition values. The focus of the research described in this thesis is to improve the efficiency of the second and third steps above, through the use of improved sparse linear algebra. Further details of these bottlenecks, and our improvements, are described in the next two chapters.

Chapter 5

Reaction Diffusion Equation

One focus of this work is the development and use of a new block preconditioned GMRES method for solving the linear system resulting from the discretised momentum balance equations (2.4) based upon the Taylor-Hood FEM. Additionally we use an AMG preconditioned Newton-GMRES method for solving the linear system arising from discretised nonlinear reaction-diffusion system of equation (2.6) based upon the piecewise linear FEM. This chapter is concerned with the latter problem, whilst the optimal solution of the momentum equations is considered in Chapter 6.

The mathematical model and computational model have been discussed respectively in Chapter 2 Section 2.2 and Chapter 4 Section 4.2. In this chapter, the efficient solution of the simpler system in this mathematical model, which is the nonlinear equation system (2.6), is described in greater detail. In Section 5.1 the efficient solution of the reaction-diffusion equation is discussed. Moreover, comparison of results between the MUMPS solver, which is used in [57], and the proposed method in this work, which is AMG preconditioned GMRES, are presented. Finally, the current chapter is concluded by highlighting the main achievements.

5.1 Nonlinear Diffusion Solver

In this section the solutions of the nonlinear system (2.6) are introduced. In general, a nonlinear system can be written as

$$F(c) = 0$$

for a vector, $c = \{c_1, c_2, .., c_n\}$, of n unknown values, where F is a set $\{F_1, F_2, ..., F_n\}$ of n nonlinear equations. The most popular solver for such a nonlinear system is Newton's method (see Chapter 3 Section 3.4.1 for full description).

In this work, as mentioned in the previous chapter, Newton's method following a Galerkin FEM discretization is used to solve the reaction-diffusion equation system (2.6). A preconditioned GMRES method is used to solve the algebraic linear system that arises at each Newton iteration :

$$J(c)\Delta c = -F(c)$$

and updating $c_{k+1} = c_k + \Delta c$ where k is the nonlinear iteration. $J_{ij}(c) = \frac{\partial F_i(c)}{\partial c_j}$ is the Jacobian matrix and c is the nutrient concentration.

The Jacobian matrix entries are calculated using a numerical differentiation technique to reduce the required memory. So, the Jacobian entry in row i and column j can be approximated using a forward difference formula as

$$J_{ij}(c) \approx \frac{F_i(c + \epsilon\delta_j) - F_i(c)}{\epsilon}, \tag{5.1}$$

in which ϵ is a small positive number and δ_j is the vector with all zero entries expect for a one in row j:

$$\delta_j = \begin{pmatrix} 0 \\ . \\ . \\ 1 \\ 0 \\ . \\ . \\ 0 \end{pmatrix},$$

The Jacobian matrix in this model is sparse. The matrix is stored in compressed sparse row (CSR) format. The form of the sparsity pattern of Jacobian is illustrated in Figure 5.1 for square regular grid and in Figure 5.2 for circle unstructured grid. The structure of the pattern depends on the ordering of the nodes (unknowns) on the grid. The nonlinear system in this model is solved on a once refined grid. This creates a set of sub-nodes that are formed by uniformly refining the existing triangular mesh. These are listed after the nodes from the original mesh (see Figure 5.1 and Figure 5.2 the sparsity pattern structure).

The main contribution of this research is to use an AMG preconditioned Newton-GMRES method for this problem. In this method the nonlinear iterative method is the outer iteration, while the GMRES based linear iterative method is the inner iteration. At each iteration, the linear system

$$J(c_k)\Delta c_k = -F(c_k)$$

is solved using AMG preconditioned GMRES. We use HSL-MI20 for the AMG preconditioner, which is is available in Harwell Subroutine Library (HSL) [1]. We refer the reader to [18] for greater detail about HSL-MI20.

Figure 5.1: The sparsity pattern of matrix J for grid size 33^2 and unknowns N=4225.

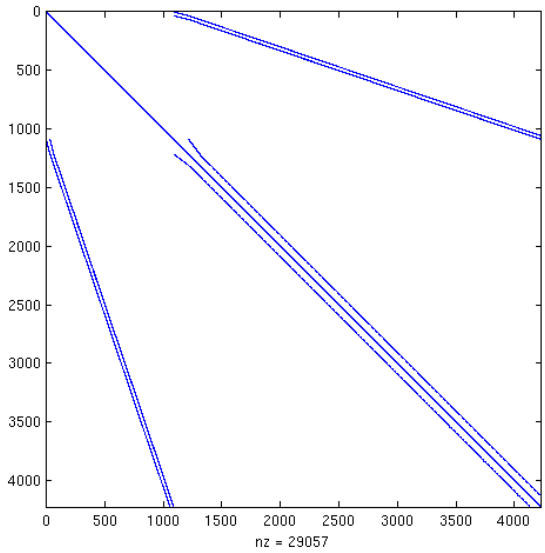
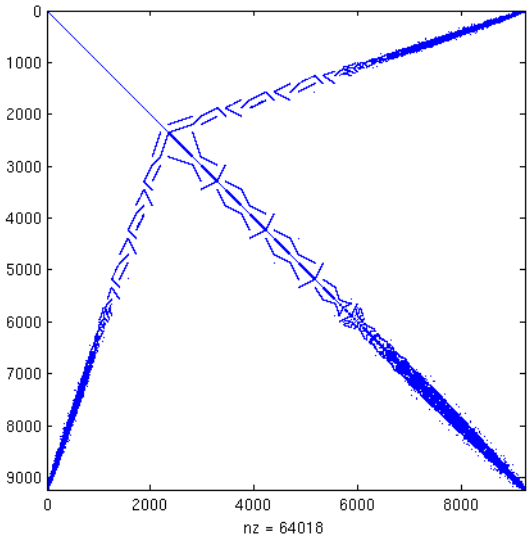


Figure 5.2: The sparsity pattern of matrix J for an unstructured grid with unknowns N= 9236.



5.2 Numerical Results

For testing purposes all the results in this section are obtained using a fixed time step size $\Delta t = 0.25$ with the number of steps 1000. Moreover, the tolerance value, which is used to determine the convergence of Newton iteration, is $1e-12$.

For ILU preconditioned GMRES (see Section 3.3.1), we used the ILU preconditioner from SPARSKIT package called ILUD. ILUD depends only on one parameter, which is a drop tolerance. The best choice of drop tolerance is found to be $1e-2$. The restart parameter (m) of GMRES and the maximum number of GMRES iterations (maxGI) are given the same value that is $m = maxGI = 60$, to avoid restarting GMRES. Furthermore, the absolute and relative tolerances for the GMRES are chosen respectively 0.0 and $1e-3$, i.e. we only consider the relative tolerance. This is justified since we are using GMRES as an inner approximate solver rather than the outer iteration.

For the AMG-preconditioned GMRES method, we make use of the software implementation that is available in Harwell Subroutine Library (HSL) [1]. This includes: HSL-MI20 for the AMG method and HSL-MI24 for the GMRES method. In practice, one V-cycle AMG is used with two pre- and post-smoothing steps. The Gauss-Seidel method is used as smoother. Here, the restart parameter (m) and the maximum number of GMRES iterations are given the same value as $m = maxGI = 8$. Also, the absolute and relative tolerances are given respectively 0.0 and $1e-3$. Then later we test another relative tolerance, which is $1e-6$. For greater details about the implementation of the AMG preconditioning from HSL-MI20 read [18].

5.2.1 Regular Grids

We begin with the application of the model on a square domain using regular triangular grids such as in Figure 5.3. The sparse direct solver MUMPS is implemented in this work to investigate and compare the improvement in the performance with the iterative solvers that are based upon ILU and AMG-preconditioned GMRES.

Figure 5.3: Regular triangular grid

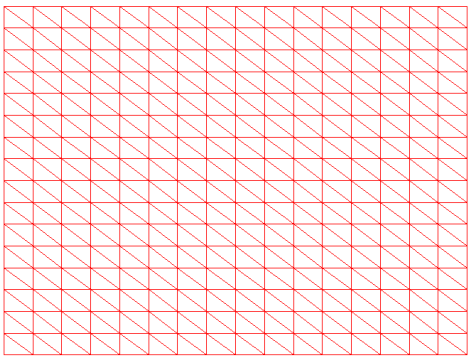


Table 5.1: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using MUMPS on regular grids: N is the number of unknowns, NI is Newton iterations.

Grid	N	average NI	average time per step (sec.)
33^2	4225	3	0.055270868
65^2	16641	3	0.3989106238
129^2	66049	3	2.013567176
257^2	263169	-	-

The results in Table 5.1 are obtained by using the MUMPS solver. The number of Newton iterations is fixed for all grids sizes, while the run-time increased by factor more than 5. This means that the solution required at best $\mathcal{O}(NlogN)$ complexity, where N is the number of unknowns. The MUMPS solver, in this model, is unable to solve large problems on the test computer without excessive memory. That is due to the momentum balance system solver rather than the reaction-diffusion system.

Table 5.2: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using $ILUD(1e-2)$ preconditioned GMRES on regular grids: N is the number of unknowns, NI is Newton iterations, GI is GMRES iterations and the relative tolerance for GMRES is $tol = 1e-3$.

Grid	N	average NI	average GI per NI	average time per step (sec.)	Time per GI
33^2	4225	3.9970	15.7693	0.03749776958	0.00059492041523
65^2	16641	4	29.9853	0.1999540808	0.001667101
129^2	66049	4	57.2512	1.5315831429	0.0066879958
257^2	263169	-	-	-	-

Table 5.3: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using AMG preconditioned GMRES on regular girds: N is the number of unknowns, NI is Newton iterations, GI is GMRES iterations and the relative tolerance for GMRES is $tol = 1e-3$.

Grid	N	average NI	average GI per NI	average time per step (sec.)	Time per GI
33^2	4225	3.9840	2.0346	0.04086334347	0.0050412185789
65^2	16641	3.9860	2.0484	0.16147179	0.019776279809
129^2	66049	3.9830	2.2471	0.6678587251	0.074619424966
257^2	263169	3.6643	2.4975	2.57317857999	0.28117286623

Tables 5.2 and 5.3 show that the average Newton iterations, over all time steps, required for the convergence of different problems are still fixed for both cases using the ILU preconditioner and the AMG preconditioner with the GMRES solver. In Table 5.2 the number of GMRES iterations per Newton iteration, which is obtained by using ILU preconditioned GMRES, grows

like the square root of the number of unknowns: $GI \propto \sqrt{N}$. Since a single iteration has cost $\mathcal{O}(N)$ time, overall this solver has $\mathcal{O}(N^{1.5})$ complexity. In Table 5.3 the number of GMRES iterations per Newton iteration behaves as $\mathcal{O}(1)$ when using the AMG preconditioner and hence the time scales linearly, $\mathcal{O}(N)$. From the results recorded in the last column in Table 5.2 and Table 5.3, it can be observed that the ILU preconditioning is cheaper to implement than the AMG preconditioning, where a single iteration of GMRES with ILU needs less time than a single iteration of GMRES with AMG. However, the ILU based solver is inefficient due to its sub-optimal complexity.

By comparing the results of the three previous solvers, we clearly see that with MUMPS and ILU preconditioned GMRES solvers, 129×129 is largest problem we able to solve when we sequentially halve the grid spacing. However, with AMG preconditioned GMRES we are able to solve the larger problem 257×257. Also, our method, AMG preconditioned GMRES, uses less time to converge that other two methods (see Figure 5.4). So, AMG is more efficient and can solve this problem in $\mathcal{O}(N)$ time which is optimal.

From our results, we can observe that good preconditioning can significantly reduce the number of iterations and the run time. Figure 5.5 shows the difference between the number of GMRES iterations required using the different preconditioning methods. It is appears that the iteration number from using ILU preconditioning is approximately $\mathcal{O}(N^{1/2})$ complexity, while it is fixed when using AMG preconditioning.

Figure 5.4: Comparison between running times requirement for using difference solvers: ILUD preconditioned GMRES (blue line), AMG preconditioned GMRES (red line) and MUMPS (yellow line), with relative tolerance $1e-3$, the results taken from Table 5.1, Table 5.2 and Table 5.3.

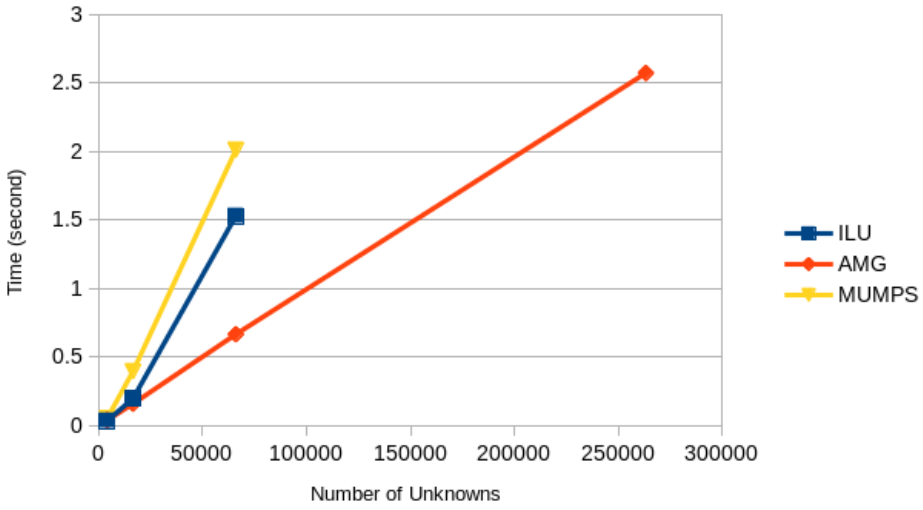


Figure 5.5: Comparison between number of GMRES iterations required for using different solvers: ILUD preconditioned GMRES (blue line) and AMG preconditioned GMRES (red line), with relative tolerance $1e-3$, taken from Table 5.2 and Table 5.3.

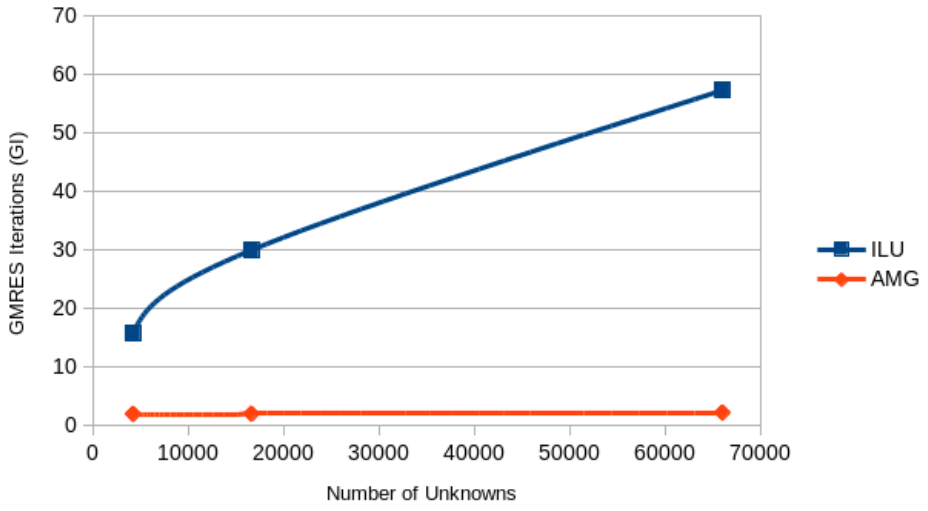


Table 5.4: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using AMG preconditioned GMRES on regular grids: N is the number of unknowns, NI is Newton iterations, GI is GMRES iterations and the relative tolerance for GMRES is $tol = 1e-6$.

Grid	N	average NI	average GI per NI	average time per step (sec.)	Time per GI
33^2	4225	3	4	0.03464901336	0.00288741778
65^2	16641	3	4	0.135496634	0.011291386167
129^2	66049	3	4	0.546035211599	0.045502934299
257^2	263169	3	4.003	2.27421902699	0.18937622008

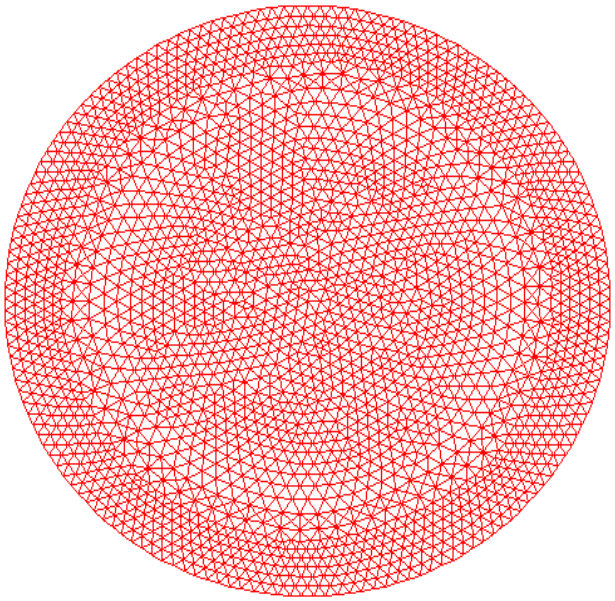
For more investigation of our solver (AMG preconditioned GMRES), the GMRES relative tolerance is reduced to the $1e-6$ and the results obtained are shown in Table 5.4. The numbers of Newton and GMRES iterations is still fixed. However, the Newton method converges at 3 iterations, in Table 5.4, rather than approximately 4 iterations in Table 5.3. That make the run-time slightly faster when $1e-6$ is used. Furthermore, the GMRES required slightly more iterations to converge. This is expected since we are solving the inner, linear system more precisely. The run-time still behaves optimally.

5.2.2 Unstructured Grids

The other domain used in this work is an approximately circular unstructured triangular grid of radius 16 shown in Figure 5.6. We start with the same grid from [57] that has the number of unknowns 9236. We then apply uniform mesh refinements to obtain a sequence of finer and finer grids, each with approximately four times more unknowns than the previous.

In this subsection, we compare between the old solver, which is MUMPS, and the new solver, which is AMG preconditioned GMRES. The ILU preconditioner is not considered here because we proved, in the previous subsection, that the performance of the GMRES solver with AMG preconditioning is much better than that with ILU preconditioning.

Figure 5.6: Unstructured triangular grid.



The MUMPS solver results that are presented in Table 5.5 show a fixed number of Newton iterations as with the regular grids. The computational time grows by factor of nearly 5. In this case it is difficult to determine the true behaviour of the time because we are only able to solve two levels of grids size due to memory limitations. However it appears that, like the regular grids, we can expect at best $\mathcal{O}(NlogN)$ complexity.

Also for this domain, our method proved much better performance than MUMPS. The number of Newton iterations is fixed in both Tables 5.6 and 5.7 and the number of GMRES iterations is almost fixed for both GMRES solver with $1e-3$ and $1e-6$ tolerances. In addition, in both tables the computational cost and the time has optimal solution time behaviour. Furthermore, the effect of the GMRES tolerance value on the performance of Newton's method can be seen. When the GMRES tolerance is large the Newton's method required slightly more

iterations to converge comparison with the Newton iterations that obtained from using small GMRES tolerance. Because of that the running time is better when the GMRES tolerance $1e-6$ is used. On the contrary, the GMRES method needed more iterations with the small GMRES tolerance to converge, while it is need less iterations when the value of the tolerance increased.

Table 5.5: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using MUMPS on fully unstructured grids: N is the number of unknowns, NI is Newton iterations.

N	average NI	average time per step (sec.)
9236	3	0.236746629
36627	3	1.1655954159
145877	-	-

Table 5.6: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using AMG preconditioned GMRES on fully unstructured grids: N is the number of unknowns, NI is Newton iterations, GI is GMRES iterations and the relative tolerance for GMRES is $tol = 1e-3$.

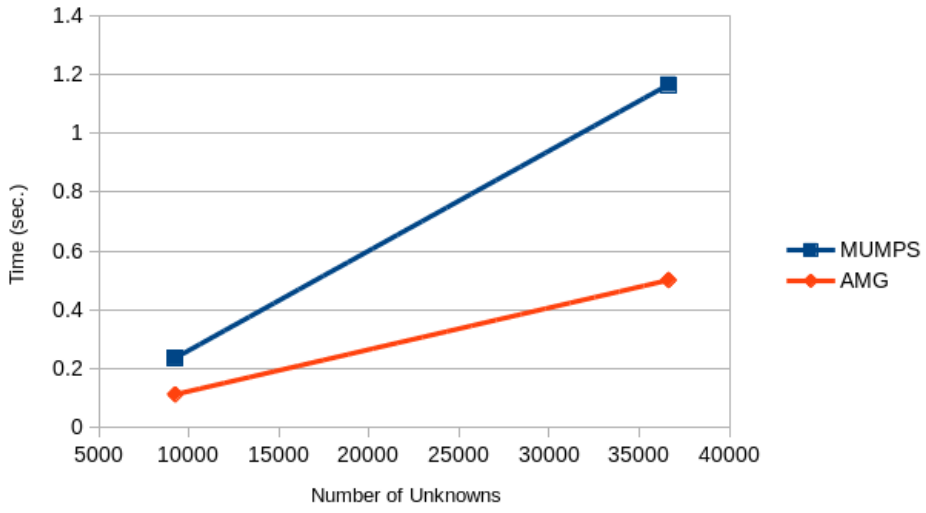
N	average NI	average GI per NI	average time per step (sec.)	Time per GI
9236	4	2.8108	0.1128536177	0.0100374998
36627	4	2.9897	0.501162218	0.04190740024
145877	4	3.2523	2.066785337	0.1588710556

Table 5.7: The average running times (in seconds), over the 1000 time steps, required for solving the linear algebraic system using AMG preconditioned GMRES on fully unstructured grids: N is the number of unknowns, NI is Newton iterations, GI is GMRES iterations and the relative tolerance for GMRES is $tol = 1e-6$.

N	average NI	average GI per NI	average time per step (sec.)	Time per GI
9236	3	4.667	0.09999410271	0.00714192577
36627	3	5.380	0.421321088	0.02610415663
145877	3	5.748	1.747686061	0.101350386279

Our results in the three previous tables illustrates that the run-time required for solving the problem when using AMG preconditioned GMRES on unstructured grids is increasingly superior to the time required by MUMPS as N increases (see Figure 5.7). Also, with AMG preconditioning we are able to solve larger problem than with MUMPS. The MUMPS solver can solve the system with up to 36627 nodes due to the cost of momentum balance systems solver, whereas our solver can be used for systems upto 4 times larger.

Figure 5.7: Comparison between running times using difference solvers: MUMPS (blue line) and AMG preconditioned GMRES (red line), with relative tolerans $1e-3$, the results taken from Table 5.5 and Table 5.6.



5.3 Discussion

The main achievement of this chapter is to apply an optimally preconditioned Newton-GMRES algorithm , with preconditioner based on AMG, to efficiently solve the discrete nonlinear system resulting from FEM approximation of the reaction-diffusion equation in [57]. This allows us to replace the MUMPS sparse direct solver used in [57], and therefore solve systems of larger size and with greater efficiency. From the results presented in the previous section, we noted that the MUMPS solver is much slower than AMG preconditioned GMRES. Moreover, MUMPS is unable to solve very large problems due to its use of memory.

In conclusion, in our algorithm we used the Newton's method to linearise the nonlinear equation then solved by the AMG preconditioned GMRES. This involved sparse matrix techniques to minimise storage and AMG preconditioning for the GMRES method to minimise computational work. The optimally efficient algorithm for this specific problem has not previously been developed.

In the next chapter, the discretised momentum balance equation system is solved with a separate preconditioned GMRES solver. Specifically, a novel block preconditioning is developed to achieve optimal convergence behaviour. That is then following by Chapter 7, which includes the computational memory cost and CPU time for the whole model.

Chapter 6

Momentum Balance Equations

In the previous chapter, the nonlinear part from the mathematical model, which is discussed in Chapter 2 Section 2.2, is solved using AMG preconditioned GMRES and that achieved optimal convergence behaviour. This chapter focuses on the linear system representing the momentum equations (2.4) from the mathematical model. The most important parts in the current chapter are developing and using a new block preconditioned GMRES method for solving the algebraic linear system resulting from the approximated momentum balance equations (2.4) based upon the Taylor-Hood FEM.

This chapter is organised as follows: Section 6.1 introduces the structure of the discrete system and the solution of the algebraic linear system. Section 6.2 then describes the steps required to develop a new preconditioner. Following this is Section 6.3, which presents the numerical results. Finally, this chapter is concluded by summaries of the main achievements.

6.1 The Discrete Linear System

In this description, the block-matrix system has been written in four different forms to make the steps to design our new preconditioner clearer to explain.
Firstly, the block-matrix system for the discrete momentum equations in our model may be

expressed in the form:

$$\begin{pmatrix} k_{xx11} & k_{xy11} & k_{xx12} & 0 & k_{xx14} & 0 & k_{xx13} & 0 & C_{x1} \\ k_{yx11} & k_{yy11} & 0 & k_{yy12} & 0 & k_{yy14} & 0 & k_{yy13} & C_{y1} \\ k_{xx21} & 0 & k_{xx22} & k_{xy22} & k_{xx24} & 0 & k_{xx23} & 0 & C_{x2} \\ 0 & k_{yy21} & k_{yx22} & k_{yy22} & 0 & k_{yy24} & 0 & k_{yy23} & C_{y2} \\ k_{xx41} & 0 & k_{xx42} & 0 & k_{xx44} & k_{xy44} & k_{xx43} & 0 & C_{x4} \\ 0 & k_{yy41} & 0 & k_{yy42} & k_{yx44} & k_{yy44} & 0 & k_{yy43} & C_{y4} \\ k_{xx31} & 0 & k_{xx32} & 0 & k_{xx34} & 0 & k_{xx33} & k_{xy33} & 0 \\ 0 & k_{yy31} & 0 & k_{yy32} & 0 & k_{yy34} & k_{yx33} & k_{yy33} & 0 \\ B_{x1}^T & B_{y1}^T & B_{x2}^T & B_{y2}^T & B_{x4}^T & B_{y4}^T & B_{x3}^T & B_{y3}^T & 0 \end{pmatrix} \begin{pmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x4} \\ u_{y4} \\ u_{x3} \\ u_{y3} \\ p_4 \end{pmatrix} = \begin{pmatrix} fx_1 \\ fy_1 \\ fx_2 \\ fy_2 \\ fx_4 \\ fy_4 \\ fx_3 \\ fy_3 \\ 0 \end{pmatrix}. \quad (6.1)$$

The coefficient matrix A has a 9×9 block structure with 8 velocity variables and 1 pressure variable. The first 8 block rows are obtained from equations (4.9) and (4.10). Each two rows express the x and y directions of momentum of each phase, starting from the top healthy phase, tumour phase, extracellular phase and blood vessels phase. The final block row is obtained from equation (4.8), the continuity equation. Figure 6.1 and Figure 6.2 show the sparsity pattern of this coefficient matrix for the regular and unstructured grids (as illustrated by Figure 5.3 and Figure 5.6), respectively. As mentioned earlier, the momentum balance equations (2.4) are approximated by Taylor-Hood FEM, which uses linear and quadratic basis functions for pressure and velocities variables (as shown in Figure 3.2). So, the number of unknowns in this linear system is $N = 8n^q + n^l$ in which n^q is the nodes on the vertices and the edge of the elements and n^l is the node on the vertices of the elements only.

Figure 6.1: Structure of sparse matrix A for a 33^2 grid with unknowns N=34889.

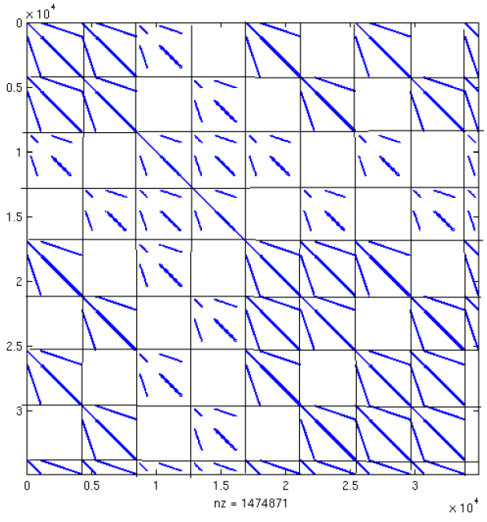
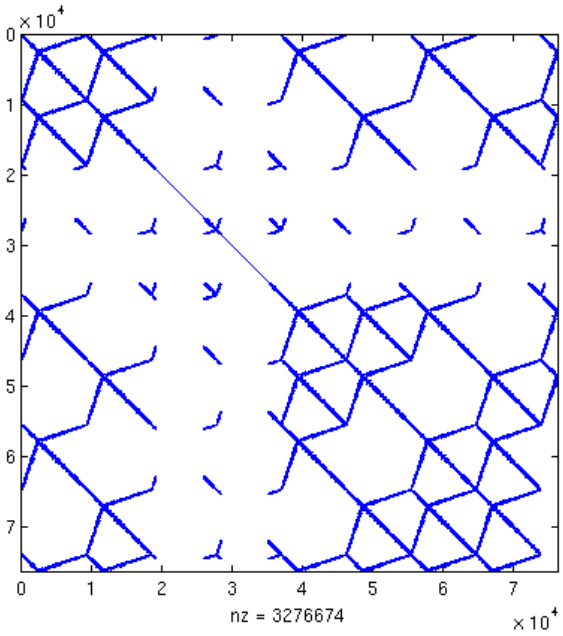


Figure 6.2: Structure of sparse matrix A for unstructured grid with unknowns N=76237.



The second form that this linear system can be written in is:

$$\underbrace{\begin{pmatrix} k_{11} & k_{12} & k_{13} & k_{14} & cc_1 \\ k_{21} & k_{22} & k_{23} & k_{24} & cc_2 \\ k_{31} & k_{32} & k_{33} & k_{34} & cc_3 \\ k_{41} & k_{42} & k_{43} & k_{44} & cc_4 \\ b_1^T & b_2^T & b_3^T & b_4^T & 0 \end{pmatrix}}_{A} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ p_4 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ 0 \end{pmatrix}. \tag{6.2}$$

The coefficient matrix A in (6.2) is determined from (6.1), in which the blocks k_{ij} of (6.2) are simply 2×2 block matrices from (6.1). The blocks cc_j and b_j^T are 2×1 and 1×2 block matrices from (6.1). For example

$$k_{11} = \begin{pmatrix} k_{xx11} & k_{xy11} \\ k_{yx11} & k_{yy11} \end{pmatrix},$$

$$cc_1 = \begin{pmatrix} C_{x1} \\ C_{y1} \end{pmatrix},$$

$$u_1 = \begin{pmatrix} u_{x1} \\ u_{y1} \end{pmatrix},$$

and

$$b_1^T = \begin{pmatrix} B_{x1}^T & B_{y1}^T \end{pmatrix}.$$

The third form that this linear system may be written in is as follows:

$$\underbrace{\begin{pmatrix} K_{11} & K_{12} & C_1 \\ K_{21} & K_{22} & C_2 \\ B_1^T & B_2^T & 0 \end{pmatrix}}_{A} \begin{pmatrix} U_1 \\ U_2 \\ p_4 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ 0 \end{pmatrix} \tag{6.3}$$

where

$$K_{11} = \begin{pmatrix} k_{xx11} & k_{xy11} & k_{xx12} & 0 \\ k_{yx11} & k_{yy11} & 0 & k_{yy12} \\ k_{xx21} & 0 & k_{xx22} & k_{xy22} \\ 0 & k_{yy21} & k_{yx22} & k_{yy22} \end{pmatrix},$$

$$U_1 = \begin{pmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \end{pmatrix}, \qquad U_2 = \begin{pmatrix} u_{x4} \\ u_{y4} \\ u_{x3} \\ u_{y3} \end{pmatrix},$$

and

$$F_1 = \begin{pmatrix} fx_1 \\ fy_1 \\ fx_2 \\ fy_2 \end{pmatrix}, \qquad F_2 = \begin{pmatrix} fx_4 \\ fy_4 \\ fx_3 \\ fy_3 \end{pmatrix}.$$

Finally, the fourth form that we express the block-matrix system in is

$$\underbrace{\begin{pmatrix} K & C \\ B^T & 0 \end{pmatrix}}_{A} \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}, \tag{6.4}$$

in which K is 8×8 block matrix that includes all the k blocks in (6.1), C is the 8×1 block matrix, which includes all the C blocks in (6.1), and B^T is the 1×8 block matrix that includes all the B^T in the system (6.1). Also, U denote all the 8 velocity variables and P only p_4.

It is clear that this problem is Stokes-like (though it is far more complex than a standard Stokes problem). In this model, matrix A is large and sparse and the system should be solved iteratively. The discrete system is nonsymmetric, hence to solve this problem iteratively we choose a preconditioned GMRES method. In general, Krylov methods converge slowly when applied to this type of problem, so we will also need a good preconditioner to achieve robust convergence rates.
In the next section, our techniques for preconditioning this system are described in detail.

6.2 The block preconditioning

This section presents how we have developed a new efficient block preconditioner for the system (6.4). We refer the reader to Section 3.3.2 for an introduction to block preconditioning.

The block preconditioners in this work divide into two types: firstly, preconditioners based upon an exact Schur complement and secondly, preconditioners based upon approximate Schur complements. The structure of the block preconditioners, in addition to the difference between them, and the performance of each of them is explained in the following subsections.

6.2.1 Exact Schur Complement Methods

In this subsection we present some preconditioners that are based on the exact Schur complement.

We start with the coefficient matrix A in the form (6.4), our first preconditioner can be written as follows:

$$P1 = \begin{pmatrix} K & 0 \\ 0 & S \end{pmatrix}$$

which is a block diagonal preconditioner, where $S = B^T K^{-1} C$ is the standard Schur complement matrix. Algorithm 4 illustrates the steps require to solve the linear system $P1z = r$, which is in the third step in the GMRES Algorithm 3, where z and r vectors are defined as

$$z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \qquad r = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}.$$

Algorithm 4 Solving P1z=r in step 3 from GMRES Algorithm 3

1: Solve $Kz_1 = r_1$ directly.
2: Solve $Sz_2 = r_2$ directly.

The second preconditioner may be written in the form:

$$P2 = \begin{pmatrix} K & C \\ 0 & \pm S \end{pmatrix}, \tag{6.5}$$

which is a block upper triangular preconditioner with the same Schur complement S as P1. The system $P2z = r$ is solved by using block backward substitution and solving each equation directly (see Algorithm 5 for the case in which $+S$ is used in the block of (6.5)).

Algorithm 5 Solving P2z=r in step 3 from GMRES Algorithm 3

1: Solve $Sz_2 = r_2$ directly.
2: Update $r_1 = r_1 - Cz_2$.
3: Solve $Kz_1 = r_1$ directly.

The coefficient matrix A also can be written in the 3×3 block structure of (6.3). To generate the third preconditioner we apply a block elimination on matrix A in this form to get:

$$\begin{pmatrix} K_{11} & K_{12} & C_1 \\ 0 & K_{22}-K_{21}K_{11}^{-1}K_{12} & C_2-K_{21}K_{11}^{-1}C_1 \\ 0 & B_2^T-B_1^TK_{11}^{-1}K_{12} & -B_1^TK_{11}^{-1}C_1 \end{pmatrix}.$$

Again applying a second elimination step leads to:

$$P3 = \begin{pmatrix} K_{11} & K_{12} & C_1 \\ 0 & A1 & A2 \\ 0 & 0 & S2 \end{pmatrix}, \tag{6.6}$$

in which

$$\begin{aligned} A1 &= K_{22}-K_{21}K_{11}^{-1}K_{12}, \\ A2 &= C_2-K_{21}K_{11}^{-1}C_1, \end{aligned}$$

and

$$S2 = -(B_1^TK_{11}^{-1}C_1+(B_2^T-B_1^TK_{11}^{-1}K_{12})(K_{22}-K_{21}K_{11}^{-1}K_{12})^{-1}(C_2-K_{21}K_{11}^{-1}C_1)) \ .$$

Note that S2 has a minus sign due to application of the elimination steps. However as with preconditioner P2, we may consider either a + sign or a - sign with the Schur complement, so $\pm S2$. This type of preconditioner is demonstrated to allow GMRES to converge with three iterations independent of the size of the grid. The steps in Algorithm 6 may be followed to solve the system $P3z = r$, in which z and r vectors are defiend here such as

$$z = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix}, \qquad r = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}.$$

Algorithm 6 Solving P3z=r in step 3 from GMRES Algorithm 3

1: Solve $S2z_3 = r_3$ directly
2: Update $r_2 = r_2 - A2z_3$
3: Solve $A1z_2 = r_2$ directly
4: Update $r_1 = r_1 - K_{12}z_2 - C_1z_3$
5: Solve $K_{11}z_1 = r_1$ directly

In conclusion, we began with the preconditioners P1 and P2 from [79] then we generalized these to develop other preconditioners, using Gaussian elimination to get the upper triangular preconditioner with the generalized Schur complement. The advantage of these preconditioners with an exact Schur complement matrix is that we observe that the convergence rate of

our Krylov subspace solver is independent on the problem size. This is demonstrated in our numerical experiments. However, additional memory and computational time is required to form these types of preconditioner due to using the inverse matrices. To attempt to develop a new efficient preconditioner, we take the upper triangular blocks of matrix A, and also we replaced the exact Schur complement by a simple sparse approximation, which is the pressure mass matrix.

The next subsection contains descriptions of these preconditioners which include the pressure mass matrix.

6.2.2 Approximate Schur Complement Methods

In the previous subsection, we presented three preconditioners dependent on using Gaussian elimination to get block diagonal or block upper triangular preconditioners with the Schur complement. In addition, we introduced the coefficient matrix A in four different structures (2×2 blocks in (6.4), 3×3 blocks in (6.3) , 5×5 blocks in (6.2) and 9×9 blocks in (6.1)).

In this subsection we create different preconditioners without using the exact Schur complement in an attempt to improve the efficiency. The new preconditioners are created using only upper triangular blocks with the bottom right block taking the form of the mass matrix for the finite elements used to approximate the pressure. This is known as the pressure mass matrix, mp.

We begin with the matrix A structured as in (6.3), so the preconditioner $P4$ takes the form

$$P4 = \begin{pmatrix} K_{11} & K_{12} & C_1 \\ 0 & K_{22} & C_2 \\ 0 & 0 & mp \end{pmatrix},$$

in which mp is a pressure mass matrix. Because the pressure space is piecewise linear, this mass matrix is calculated on each element as

$$mp_{JI}^{(e)} = \begin{cases} Area^{(e)}/6 & \text{if } I = J \\ Area^{(e)}/12 & \text{otherwise,} \end{cases} \tag{6.7}$$

in which $I, J \in \{1, 2, 3\}$. These element matrices are then assembled to create mp.
The fifth preconditioner used in this work, with the matrix A structured in the form of (6.2), can be written as follows

$$P5 = \begin{pmatrix} k_{11} & k_{12} & k_{13} & k_{14} & cc_1 \\ 0 & k_{22} & k_{23} & k_{24} & cc_2 \\ 0 & 0 & k_{33} & k_{34} & cc_3 \\ 0 & 0 & 0 & k_{44} & cc_4 \\ 0 & 0 & 0 & 0 & mp \end{pmatrix}$$

Finally, if the matrix A is written as in (6.1), in this case we introduce preconditioners P6 and

P7 which take the forms as follows

$$P6 = \begin{pmatrix} k_{xx11} & K_{xy11} & k_{xx12} & 0 & k_{xx14} & 0 & k_{xx13} & 0 & C_{x1} \\ 0 & k_{yy11} & 0 & k_{yy12} & 0 & k_{yy14} & 0 & k_{yy13} & C_{y1} \\ 0 & 0 & k_{xx22} & k_{xy22} & k_{xx24} & 0 & k_{xx23} & 0 & C_{x2} \\ 0 & 0 & 0 & k_{yy22} & 0 & k_{yy24} & 0 & k_{yy23} & C_{y2} \\ 0 & 0 & 0 & 0 & k_{xx44} & k_{xy44} & k_{xx43} & 0 & C_{x4} \\ 0 & 0 & 0 & 0 & 0 & k_{yy44} & 0 & k_{yy43} & C_{y4} \\ 0 & 0 & 0 & 0 & 0 & 0 & k_{xx33} & k_{xy33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{yy33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & mp \end{pmatrix},$$

and

$$P7 = \begin{pmatrix} k_{xx11_{AMG}} & k_{xy11} & k_{xx12} & 0 & k_{xx14} & 0 & k_{xx13} & 0 & C_{x1} \\ 0 & k_{yy11_{AMG}} & 0 & k_{yy12} & 0 & k_{yy14} & 0 & k_{yy13} & C_{y1} \\ 0 & 0 & k_{xx22_{AMG}} & k_{xy22} & k_{xx24} & 0 & k_{xx23} & 0 & C_{x2} \\ 0 & 0 & 0 & k_{yy22_{AMG}} & 0 & k_{yy24} & 0 & k_{yy23} & C_{y2} \\ 0 & 0 & 0 & 0 & k_{xx44_{AMG}} & k_{xy44} & k_{xx43} & 0 & C_{x4} \\ 0 & 0 & 0 & 0 & 0 & k_{yy44_{AMG}} & 0 & k_{yy43} & C_{y4} \\ 0 & 0 & 0 & 0 & 0 & 0 & k_{xx33_{AMG}} & k_{xy33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{yy33_{AMG}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & mp_{CG} \end{pmatrix}.$$

P6 and P7 have almost the same entries but the AMG subscript and the CG subscript in P7 denote the solution methods that are used to solve these blocks. For solving both systems $P6z = r$ and $P7z = r$, z and r are considered such that

$$z = \begin{pmatrix} zu_1 \\ zv_1 \\ zu_2 \\ zv_2 \\ zu_4 \\ zv_4 \\ zu_3 \\ zv_3 \\ z_p \end{pmatrix}, \qquad r = \begin{pmatrix} ru_1 \\ rv_1 \\ ru_2 \\ rv_2 \\ ru_4 \\ rv_4 \\ ru_3 \\ rv_3 \\ r_p \end{pmatrix}.$$

The differences between $P6$ and $P7$ are: in $P6$ all the diagonal blocks (and mp) are solved using a direct solver (backslash in Matlab). The solver steps for using P6 are illustrated in Algorithm 7. While in $P7$ the diagonal blocks are solved only approximately, using just one V-cycle of the algebraic multigrid method, and mp is solved using the conjugate gradient method (CG). In Algorithm 8, the steps of solving $P7z = r$ is shown. In practice, one AMG V-cycle is used with Gauss Seidel smoothing and 2 pre-and post-smoothing iterations to solve the diagonal blocks, and CG is used, without preconditioning and tolerance $1e - 6$, to solve mp.

Algorithm 7 Solving P6z=r in step 3 from GMRES Algorithm 3

1: Solve $mpz_p = r_p$ directly.
2: Solve $k_{yy33}zv_3 = rv_3$ directly.
3: Update $ru_3 = ru_3 - k_{xy33}zv_3$
4: Solve $k_{xx33}zu_3 = ru_3$ directly.
5: Update $rv_4 = rv_4 - k_{yy43}zv_3 - C_{y4}z_p$
6: Solve $k_{yy44}zv_4 = rv_4$ directly.
7: Update $ru_4 = ru_4 - k_{xy44}zv_4 - k_{xx43}zu_3 - C_{x4}z_p$
8: Solve $k_{xx44}zu_4 = ru_4$ directly
9: Update $rv_2 = rv_2 - k_{yy24}zv_4 - k_{yy23}zv_3 - C_{y2}z_p$
10: Solve $k_{yy22}zv_2 = rv_2$ directly.
11: Update $ru_2 = ru_2 - k_{xy22}zv_2 - k_{xx24}zu_4 - k_{xx23}zu_3 - C_{x2}z_p$
12: Solve $k_{xx22}zu_2 = ru_2$ directly
13: Update $rv_1 = rv_1 - k_{yy12}zv_2 - k_{yy14}zv_4 - k_{yy13}zv_3 - C_{y1}z_p$
14: Solve $k_{yy11}zv_1 = rv_1$ directly.
15: Update $ru_1 = ru_1 - K_{xy11}zv_1 - k_{xx12}zu_2 - k_{xx14}zu_4 - k_{xx13}zu_3 - C_{x1}z_p$
16: Solve $k_{xx11}zu_1 = ru_1$ directly.

Algorithm 8 Solving P7z=r in step 3 from GMRES Algorithm 3

1: Solve $mpz_p = r_p$ using CG.
2: Solve $k_{yy33}zv_3 = rv_3$ using AMG.
3: Update $ru_3 = ru_3 - k_{xy33}zv_3$
4: Solve $k_{xx33}zu_3 = ru_3$ using AMG.
5: Update $rv_4 = rv_4 - k_{yy43}zv_3 - C_{y4}z_p$
6: Solve $k_{yy44}zv_4 = rv_4$ using AMG.
7: Update $ru_4 = ru_4 - k_{xy44}zv_4 - k_{xx43}zu_3 - C_{x4}z_p$
8: Solve $k_{xx44}zu_4 = ru_4$ using AMG
9: Update $rv_2 = rv_2 - k_{yy24}zv_4 - k_{yy23}zv_3 - C_{y2}z_p$
10: Solve $k_{yy22}zv_2 = rv_2$ using AMG.
11: Update $ru_2 = ru_2 - k_{xy22}zv_2 - k_{xx24}zu_4 - k_{xx23}zu_3 - C_{x2}z_p$
12: Solve $k_{xx22}zu_2 = ru_2$ using AMG
13: Update $rv_1 = rv_1 - k_{yy12}zv_2 - k_{yy14}zv_4 - k_{yy13}zv_3 - C_{y1}z_p$
14: Solve $k_{yy11}zv_1 = rv_1$ using AMG.
15: Update $ru_1 = ru_1 - K_{xy11}zv_1 - k_{xx12}zu_2 - k_{xx14}zu_4 - k_{xx13}zu_3 - C_{x1}z_p$
16: Solve $k_{xx11}zu_1 = ru_1$ using AMG.

To conclude, the preconditioners in this subsection are block-upper-triangular form, created by dropping all the entries under the diagonal of the matrix A and using the pressure mass matrix for the final diagonal block. In practice, the novel preconditioner P7 is tested with both mp and with only the diagonal of mp due to the pressure mass matrix mp being spectrally equivalent to its diagonal [113], to further improve the performance of the solution.

6.2.3 Eigenvalues

Before presenting the numerical results that are obtained from using the proposed preconditioners, the eigenvalues of each of those preconditioned system are considered in this subsection. One of the motivations for computing the eigenvalues for these problems is to understand the quality of the preconditioner for the GMRES solution [115]. If the eigenvalues of the matrix A have a large spread, then the GMRES method typically has slow convergence [115]. Moreover, a good preconditioner should be a good approximation to the original matrix A and the eigenvalues of the preconditioned system should be clustered in small groups and be bounded away from zero and infinity.

As mentioned earlier, the block preconditioners in this work are divided into two categories: firstly, the preconditioners that are approximated using block elimination and using the Schur complement, which are P1 to P3, and secondly, the preconditiuoners that are approximated by taking the upper triangular blocks and using the pressure mass matrix, which are P4 to P7.

We begin by calculating the eigenvalues of the first category. Each table, from Table 6.1 to Table 6.5, compares between the minimum and the maximum eigenvalues of the coefficient matrix A and the minimum and the maximum eigenvalues of the right preconditioned matrix AP^{-1}, for a selection of finite element grids. From these tables, it can be observed that the eigenvalues of the matrix A are spread in a wide range and this range increases as the grid size increases. Conversely, the eigenvalues of the preconditioned systems are clearly independent of the grid size. The maximum and the minimum eigenvalues associated with P1 and P2, respectively, are shown in Table 6.1 and Table 6.2. From both tables, it can be seen that, computing the eigenvalues of preconditioned systems numerically is identical to computing the eigenvalues theoretically (see Section 3.3.2). Figure 6.3 and Figure 6.4 show the N eigenvalues of the matrix A and the preconditioned matrix, $AP1^{-1}$ and $AP2^{-1}$, respectively in grid size 9^2 in which the number of unknowns $N = 2393$. Although, as shown in Figure 6.3 the eigenvalues of A are not all real, the eigenvalues of $AP1^{-1}$ and $AP2^{-1}$ are always real. In Figure 6.4, both of these preconditioners cluster the eigenvalues around the points on the real axis. The preconditioned matrix $AP1^{-1}$ clusters the eigenvalues around three points 1, $\frac{1+\sqrt{5}}{2}$ and $\frac{1-\sqrt{5}}{2}$, whereas the preconditioned matrix $AP2^{-1}$ clusters the eigenvalues around two points 1 and -1. If we choose to use the minus of the Schur complement the eigenvalues of the system $AP2^{-1}$ are clustered around $+1$ only (see Table 6.3).

Table 6.4 and Table 6.5 illustrate the largest and smallest eigenvalues of the preconditioned system $AP3^{-1}$ with +S and -S respectively. As the results in the previous preconditioner, the eigenvalues are clustered around two points, which are +1 and -1 when +S is selected, while they are clustered only around one point, which is 1 when -S is selected.

Table 6.1: Minimum and maximum eigenvalues of the coefficient matrix A and preconditioned matrix $AP1^{-1}$

Grid	MIN $\lambda(A)$	MAX $\lambda(A)$	MIN $\lambda(AP^{-1})$	MAX $\lambda(AP^{(-1)})$
9^2	0.0013	85.6117	-0.6180	1.6180
17^2	6.9685e-07	101.1776	-0.6180	1.6180
33^2	4.7706e-08	113.4432	-0.6180	1.6180

Table 6.2: Minimum and maximum eigenvalues of the coefficient matrix A and preconditioned matrix $AP2^{-1} with + S$

Grid	MIN $\lambda(A)$	MAX $\lambda(A)$	MIN $\lambda(AP^{-1})$	MAX $\lambda(AP^{-1})$
9^2	0.0013	85.6117	-1.0000	1.0000
17^2	6.9685e-07	101.1776	-1.0000	1.0000
33^2	4.7706e-08	113.4432	-1.0000	1.0000

Table 6.3: Minimum and maximum eigenvalues of the coefficient matrix A and preconditioned matrix $AP2^{-1} with - S$

Grid	MIN $\lambda(A)$	MAX $\lambda(A)$	MIN $\lambda(AP^{-1})$	MAX $\lambda(AP^{-1})$
9^2	0.0013	85.6117	1.0000	1.0000
17^2	6.9685e-07	101.1776	1.0000	1.0000
33^2	4.7706e-08	113.4432	1.0000	1.0000

Table 6.4: Minimum and maximum eigenvalues of the coefficient matrix A and preconditioned matrix $AP3^{-1} with + S$

Grid	MIN $\lambda(A)$	MAX $\lambda(A)$	MIN $\lambda(AP^{-1})$	MAX $\lambda(AP^{-1})$
9^2	0.0013	85.6117	-1.0000	1.0000
17^2	6.9685e-07	101.1776	-1.0000	1.0000
33^2	4.7706e-08	113.4432	-1.0000	1.0000

Table 6.5: Minimum and maximum eigenvalues of the coefficient matrix A and preconditioned matrix $AP3^{-1} with - S$

Grid	MIN $\lambda(A)$	MAX $\lambda(A)$	MIN $\lambda(AP^{-1})$	MAX $\lambda(AP^{-1})$
9^2	0.0013	85.6117	1.0000	1.0000
17^2	6.9685e-07	101.1776	1.0000	1.0000
33^2	4.7706e-08	113.4432	1.0000	1.0000

Figure 6.3: The eigenvalues of the coefficient matrix A in the grid size 9^2.

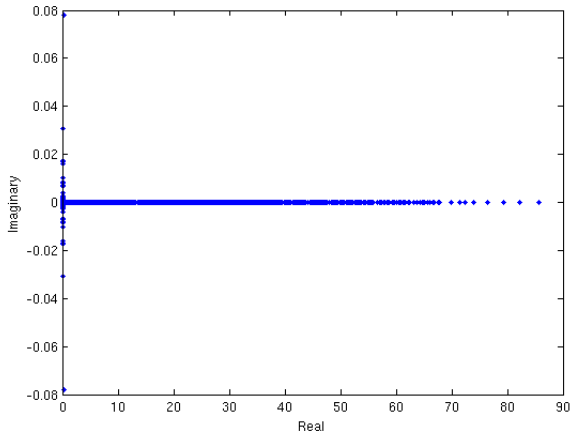
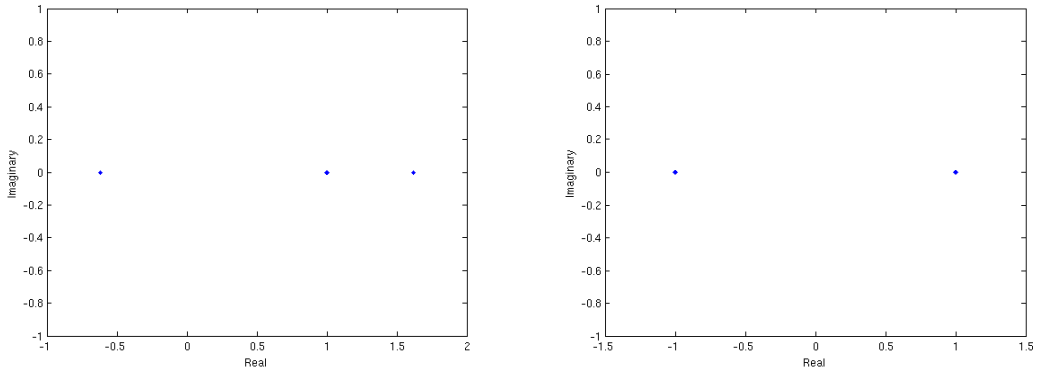


Figure 6.4: The eigenvalues of the preconditioned matrix $AP1^{-1}$ and $AP2^{-1}$ in the grid size 9^2 with +S



The eigenvalues of the preconditioned matrices for the second category of preconditioners are presented in Table 6.6-Table 6.8. As shown in the tables, the eigenvalues of resulting preconditioned systems are bounded in a small range compared to the eigenvalues of A, which are spread widely over the region. Furthermore, the spread range of the eigenvalues of the original matrix increases with the problem size, while the range of the eigenvalues of the preconditioned systems is almost fixed for all problems sizes.

Comparing Figure 6.3 with Figure 6.5, it can be observed that, using the preconditioning technique improves the results and makes the majority of the eigenvalues cluster in a small range around 1 in the complex plane, with just a small number of isolated eigenvalues near to

the origin. Importantly, Table 6.6 to Table 6.8 suggest that these eigenvalues near to the origin stay bounded away from it as the mesh is refined. Also, from Figure 6.5 we may observe that all the three preconditioners P4, P5 and P6 have a similar range of spread the eigenvalues.

Table 6.6: Minimum and maximum eigenvalues of the coefficient matrix A and preconditioned matrix $AP4^{-1}$

Grid	MIN $\lambda(A)$	MAX $\lambda(A)$	MIN $\lambda(AP^{-1})$	MAX $\lambda(AP^{-1})$
9^2	0.0013	85.6117	0.0044	1.0000
17^2	6.9685e-07	101.1776	0.0045	1.0000

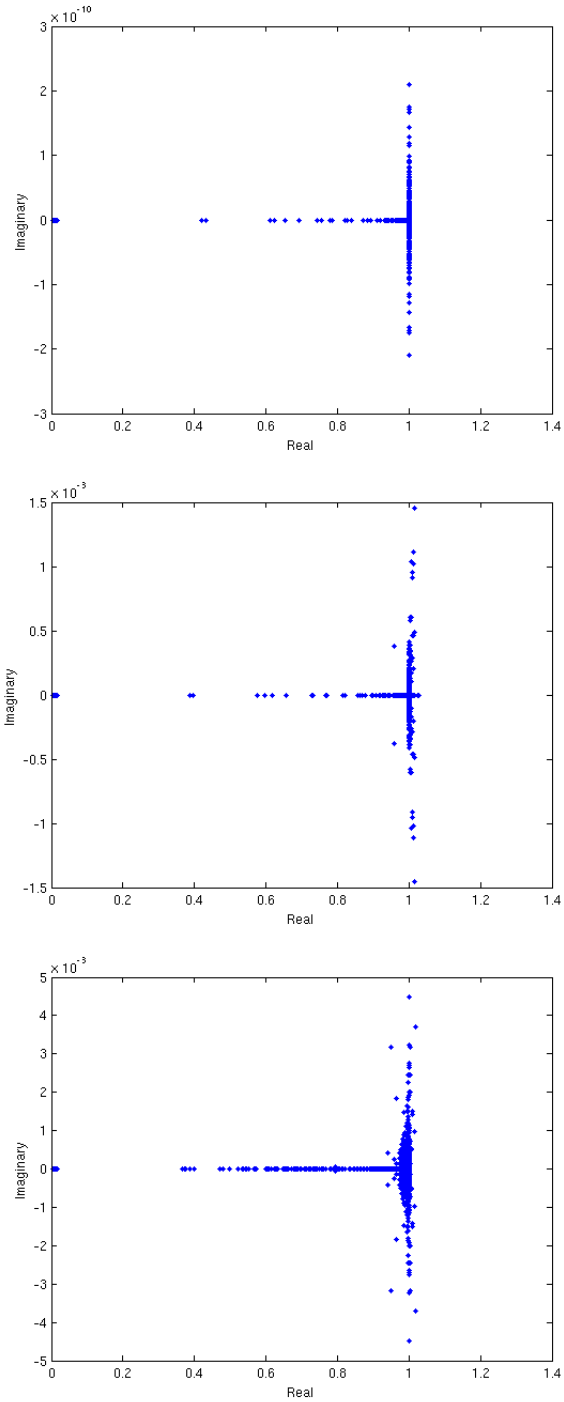
Table 6.7: Minimum and maximum eigenvalues of the coefficient matrix A and preconditioned matrix $AP5^{-1}$

Grid	MIN $\lambda(A)$	MAX $\lambda(A)$	MIN $\lambda(AP^{-1})$	MAX $\lambda(AP^{-1})$
9^2	0.0013	85.6117	0.0044	1.0270
17^2	6.9685e-07	101.1776	0.0045	1.0163

Table 6.8: Minimum and maximum eigenvalues of the coefficient matrix A and preconditioned matrix $AP6^{-1}$

Grid	MIN $\lambda(A)$	MAX $\lambda(A)$	MIN $\lambda(AP^{-1})$	MAX $\lambda(AP^{-1})$
9^2	0.0013	85.6117	0.0044	1.0182 + 0.0037i
17^2	6.9685e-07	101.1776	0.0045	1.0146 + 0.0002i
33^2	4.7706e-08	113.4432	0.0046	1.0154 + 0.0001i

Figure 6.5: The eigenvalues of the preconditioned matrix arranged from the top to the bottom: $AP4^{-1}$, $AP5^{-1}$ and $AP6^{-1}$ in the grid size 9^2.



6.3 Numerical Results

In this section we present numerical results for the solution of the modified linear system that arises from applying the different preconditioners. The results are sorted into two parts depending on the programming language that are used. In the first part, the results, which are obtained from using Matlab codes, are shown in Subsection 6.3.1 . In the second part, the results, which are obtained from using the Fortran application code, are presented in Subsection 6.3.2.

6.3.1 MATLAB

In this subsection, the numerical results are given from applying the different preconditioners, from P1 to P7, with GMRES solver using Matlab version R2012b. We output the matrix A from the Fortran code that is written by Hubbard and Byrne [57]. In all cases, the time step size is chosen to be $\Delta t = 0.25$ with the system solved is that generated at a single time step, $step = 500$. The matrix A is imported into Matlab and all steps to develop the preconditioners are done using Matlab. This has the advantage of allowing P1, P2 and P3 to be tested using the exact Schur complements computed by Matlab.

Each table in this subsection shows the results are obtained using the regular square grids (as illustrated in Figure 5.3). Each table presents the number of GMRES iterations (GI), solver time (seconds), which is the solution time ignoring the additional set-up time spent before the solver, and finally the whole time that is the time spent to solve the problem including additional time, such as that for computing the Schur complement or pressure mass matrix. The restart parameter and the maximum number of iterations are chosen to be the same value, to avoid restarting the GMRES. The GMRES relative convergence tolerance is selected to be $1e-3$.

Table 6.9: Number of GMRES iterations (GI) without Preconditioning, N is the number of unknowns.

Grid	N	GI	Solver time	Whole time
9^2	2393	136	0.1763	0.1839
17^2	9001	286	3.5453	3.6021
33^2	34889	593	35.9111	36.3238

Table 6.9 presents the number of GMRES iterations and the time required for solving the problem using GMRES without preconditioning on a sequence of uniformly refined grids. Clearly, both the computation time and the number of iterations increases significantly with increasing the number of unknowns. This confirms the need to develop preconditioners with GMRES to reduce the time and iterations required.

Table 6.10: Solving the momentum balance equations. Number of GMRES iterations (GI) with preconditioner $P1$, N is the number of unknowns.

Grid	N	GI	Solver time	Whole time
9^2	2393	3	0.0881	0.3099
17^2	9001	3	0.3980	3.6924
33^2	34889	3	1.9738	52.7889

Table 6.11: Solving the momentum balance equations. Number of GMRES iterations (GI) with preconditioner $P2$, N is the number of unknowns.

Grid	N	GI	Solver time	Whole time
9^2	2393	2	0.0669	0.2833
17^2	9001	2	0.3018	3.6076
33^2	34889	2	1.5313	52.2043

The results shown in Table 6.10 and Table 6.11 correspond to the theories in the papers [79, 114]. They show that preconditioned GMRES converges in 3 and 2 iterations, respectively, in all cases. These results prove that we can, at least in theory, use such a preconditioner with this model. Moreover, from both tables it can observed that, the diagonal preconditioner P1 achieved slightly slower time than the upper triangular preconditioner P2 due to P1 needing one more GMRES iteration than P2.

By comparing Table 6.9, Table 6.10 and Table 6.11, it can be seen that, using the preconditioning technique makes the solver itself much faster. However, the additional time required to create the preconditioner before the solver makes the code much slower when using the preconditioned GMRES due to the cost of computing the Schur complement.

Table 6.12: Solving the momentum balance equations. Number of GMRES iterations (GI) with preconditioner $P3$, N is the number of unknowns.

Grid	N	GI	Solver time	Whole time
9^2	2393	3	0.1183	2.6716
17^2	9001	3	4.1601	47.1814
33^2	34889	3	209.9459	1464.3

In Table 6.12 the equivalent results are presented when using the block upper triangular preconditioner, $P3$, with exact Schur complement. The number of iterations are still fixed while both solver and whole times in this case increased dramatically.

Overall, the best preconditioner from the all previous preconditioners, which are using the Schur complement, is P2. However, all the previous preconditioners failed to solve the large grid

problems. Furthermore, exact application of these preconditioners is prohibitively expensive and so a much faster approximation is required.

The following tables shows the impact of replacing the Schur complement by the pressure mass matrix on the results. As mentioned above, the preconditioner P2 is the best and faster preconditioner from the previous preconditioners, so, all the following experiments will be compared with P2. In the following experiment we solved the linear system using preconditioned GMRES with P4. The results reported in Table 6.13 illustrate that the GMRES with P4 needed more solver time due to it needing more iterations to converge than the GMRES with P2. However, the whole time requirement is much faster when using P4 due to the first that computing the pressure mass matrix is much cheaper than computing the Schur complement. Moreover, the preconditioner P4 required much less memory and is therefore able to solve larger problems.

Table 6.13: Solving the momentum balance equations. Number of GMRES iterations (GI) with preconditioner $P4$, N is the number of unknowns.

Grid	N	GI	Solver time	Whole time
9^2	2393	15	0.2186	0.2477
17^2	9001	22	1.3179	1.5317
33^2	34889	24	6.6800	7.4113
65^2	137353	28	38.7258	39.4313

Table 6.14: Solving the momentum balance equations. Number of GMRES iterations (GI) with preconditioner $P5$, N is the number of unknowns.

Grid	N	GI	Solver time	Whole time
9^2	2393	16	0.1431	0.1658
17^2	9001	23	0.8785	0.9962
33^2	34889	25	4.3663	5.1612
65^2	137353	29	25.2672	25.6787

Table 6.15: Solving the momentum balance equations. Number of GMRES iterations (GI) with preconditioner $P6$, N is the number of unknowns.

Grid	N	GI	Solver time	Whole time
9^2	2393	21	0.1554	0.1915
17^2	9001	29	0.7775	0.8122
33^2	34889	35	3.8615	4.5921
65^2	137353	39	19.5940	20.0121

Table 6.16: Solving the momentum balance equations. Number of GMRES iterations (GI) with preconditioner $P7$ using the full pressure mass matrix, mp, N is the number of unknowns.

Grid	N	GI	Solver time	Whole time
9^2	2393	22	0.0578	0.0878
17^2	9001	30	0.2236	0.2716
33^2	34889	36	0.8034	1.0634
65^2	137353	41	3.5356	3.8960

Table 6.17: Solving the momentum balance equations. Number of GMRES iterations (GI) with preconditioner $P7$ using the diagonal of the pressure mass matrix, N is the number of unknowns.

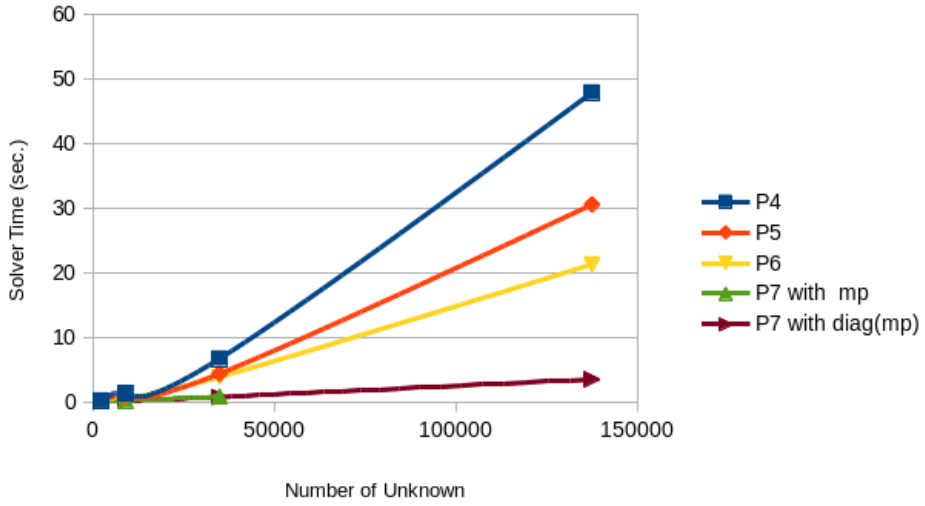
Grid	N	GI	Solver time	Whole time
9^2	2393	29	0.0645	0.0940
$17^"$	9001	33	0.2402	0.2945
33^2	34889	35	0.7439	1.0227
65^2	137353	36	3.1527	3.6472

As seen from Table 6.14 and Table 6.15 the P5 and P6 preconditioners perform better than the P4 preconditioner, as reported in Table 6.13. In the both tables, the computational time is sup-optimal (i.e. super-linear) and the required number of GMRES iterations very slightly grows as the grid is refined.

The speed of the preconditioner P6 is improved significantly by solving the diagonal blocks using the AMG method (HSL-MI20) that is available in Harwell Subroutine Library (HSL) [1], and solving the pressure mass matrix mp using the CG method to improve the efficiency of the solution compared to backslash in Matlab. The new efficient preconditioner, which is called P7, only requires marginally more iterations than P6 and achieved much faster computational time than all previous preconditioners. The best performance in this subsection however is presented in Table 6.17. This table shows the computational time and the number of iteration resulting from using the preconditioner P7 with the pressure mass matrix mp replaced by the diagonal matrix of mp (which is trivial to invert). In this table the solver and whole times behave almost optimally and the number of iterations is almost fixed. The impact of using the diagonal matrix of mp rather than using mp is shown in the next subsection in greater detail.

Overall, the preconditioners with pressure mass matrix are the best with the large problems due to using much less CPU time and also requiring far less memory. The computational solver times (in seconds) that are required to converge the preconditioned GMRES with preconditioners P4 to P7 are plotted in Figure 6.6. This figure illustrates that the performance of P7 is scaling almost linearly.

Figure 6.6: Comparison between solver times for different preconditioners with GMRES: with relative tolerance $1e-3$, the results are taken from Table 6.13 to Table 6.17.



6.3.2 FORTRAN

This subsection presents further numerical results, which are obtained using Fortran 95 and 77, for solving the linear system that results from the approximated momentum balance equations (2.4) based upon the Taylor-Hood FEM on structured and unstructured grids. The purpose is to compare the original solver used in [57], based upon the sparse direct library MUMPS, with our solver. Based upon the previous subsection, we have implemented the P7 preconditioned GMRES solver, in order to contrast the numerical results. We start the numerical experiments with fixed time step size $\Delta t = 0.25$ and the number of time steps used is 1000 to determine the cost of our new method.

The ILU preconditioner is not a topic of this thesis, however it is used in the square domain to compare the results of our new preconditioning with ILU results. For ILU preconditioned GMRES, as in the nonlinear case, the ILUD preconditioner from the SPARSKIT package is used with the best choice of drop tolerance, which is found to be $1e-2$.

For the block-preconditioned GMRES, we make use of the software implementation that is available in Harwell Subroutine Library (HSL) [1]. This includes: HSL-MI20 for the AMG method, HSL-MI21 for the CG method and HSL-MI24 for the GMRES method. The restart parameter (m) of GMRES and the maximum number of GMRES iterations (maxGI) are again given the same value. Also, the absolute and relative tolerances for the GMRES are set to $1e-10$ and $1e-3$ respectively. Later we test another relative tolerance, which is $1e-6$. The blocks in the diagonal of P7 are solved using one V-cycle of AMG with two pre-and post-smoothing stages. The pressure mass matrix mp is solved using CG with absolute and relative tolerances

$1e-10$ and $1e-3$ (or $1e-6$) respectively.

6.3.2.1 Regular Grids

This subsection present the numerical results for solving the linear system on regular square grids (as illustrated in Figure 5.3). Each table includes the number of the unknowns and the average running time (per time step) in seconds for 1000 time steps.

We begin with testing the sparse direct solver MUMPS, which is used in [57], to investigate the improvement in the performance with the iterative solvers that are based upon ILU and P7-preconditioned GMRES. As seen in Table 6.18 the CPU running time increased dramatically when the number of unknowns increased and behaves at best $\mathcal{O}(NlogN)$. This model is unable to solve large problems 257^2, when using MUMPS solver, on the test computer within available memory.

From Table 6.19 it can be observed that the performance of the iterative method is much better than for the MUMPS solver. Using ILU preconditioned GMRES reduced the computational time. However, in this model, many GMRES iterations are required. Furthermore, we unable to solve larger problems due to the memory requirement of the incomplete factorization.

Table 6.18: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using MUMPS on regular grids, N is the number of unknowns.

grid	N	average Time per step (sec.)
33^2	34889	2.96101515
65^2	137353	17.752259399
129^2	545033	134.251013799
257^2	2,171,401	-

Table 6.19: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using ILU preconditioned GMRES on regular grids: GI is GMRES iterations (the relative tolerance for GMRES is $tol = 1e-3$), N is the number of unknowns.

grid	N	average GI per step	average Time per step (sec.)
33^2	34889	31.8641	0.768367237
65^2	137353	66.9724	3.7827191406
129^2	545033	173.0649	33.09145694
257^2	2,171,401	-	-

The results that are reported in Table 6.20 and Table 6.21 are obtained from using P7 preconditioner. The difference between these tables is in the form of the pressure mass matrix mp. In Table 6.20 the mp is used, and in Table 6.21 the diagonal of mp is used. As seen from theses tables, as in the Matlab implementations, the computing efficiency of the preconditioner P7 is dependent on the choice of mp or diag(mp). When using mp, the running time is faster

than the running time than results from using the ILU preconditioner and MUMPS. Moreover, the preconditioner P7 with mp is able to solve larger problems. The computational time is scaling almost linearly and the number of GMRES iterations is almost fixed. Interestingly, when the version of P7 which uses diag(mp) is applied, not only does the average time per step decrease but also the average number of iterations. The results are shown in Table 6.21 however we are unable to explain why this variant performs much better than P7 with the full mp. In this table, the number of GMRES iterations is approximately fixed and independent on the grid size, and the running time behave almost optimally as $\mathcal{O}(N)$.

For further investigation, the GMRES relative tolerance is reduced from $1e-3$ to $1e-6$ and the results for using P7 (with diagonal of mp) are reported in Table 6.22. We can note that reducing the tolerance makes the GMRES need more iterations and more time to converge. Nevertheless, the running time still scales quite close to linearly, and the number of iterations grows only very slowly.

Table 6.20: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using P7 (with mp) preconditioned GMRES on regular grids: GI is GMRES iterations (the relative tolerance for GMRES is $tol = 1e-3$), N is the number of unknowns.

grid	N	average GI	average Time per step (sec.)
33^2	34889	33.6963	0.87885845630
65^2	137353	38.8392	3.608719996
129^2	545033	52.1099	16.437623969999
257^2	2,171,401	55.0819	68.91270631999

Table 6.21: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using P7 (with diag(mp)) preconditioned GMRES on regular grids: GI is GMRES iterations (the relative tolerance for GMRES is $tol = 1e-3$), N is the number of unknowns.

grid	N	average GI	average Time per step (sec.)
33^2	34889	34.0300	0.871865183
65^2	137353	31.0380	3.3357644739
129^2	545033	34.2438	14.1031290899
257^2	2,171,401	34.7632	57.193146920

Table 6.22: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using P7 (with mp=diag(mp)) preconditioned GMRES in regular grids: GI is GMRES iterations (the relative tolerance for GMRES is $tol = 1e-6$), N is the number of unknowns.

grid	N	average GI	average Time per step (sec.)
33^2	34889	96.2338	1.474060657999
65^2	137353	94.4456	5.75383737799
129^2	545033	112.6653	26.0799201599
257^2	2,171,401	122.5974	116.699118

Overall, the P7 preconditioned GMRES (with diagonal of mp), and the GMRES tolerance $1e-3$, achieved the best performance when compared with MUMPS and other preconditioners. The running time for MUMPS, the preconditioner ILU and preconditioner P7 with mp, and diagonal of mp are plotted in Figure 6.7. It can be observed clearly that the contributions of our proposed preconditioner P7 in the solution of this model can be summarised into two points: firstly, our solver achieves the fastest computational time. Secondly, it is able to find the solution for larger problems due to requiring much less memory than other solvers. Indeed, both the CPU time and the memory requirements grow approximately linearly with N.

Finally, Figure 6.8 shows the number of GMRES iterations that are required to converge the preconditioned GMRES with each preconditioner used in this subsection. Clearly, the best preconditioner is P7 (with diagonal of mp).

Figure 6.7: Comparison between running times using different solvers: MUMPS, ILU preconditioned GMRES, the preconditioner P7 with mp, and P7 with diag(mp), with the GMRES relative tolerance $1e-3$, the results taken from Table 6.18, Table 6.19, Table 6.20 and Table 6.21.

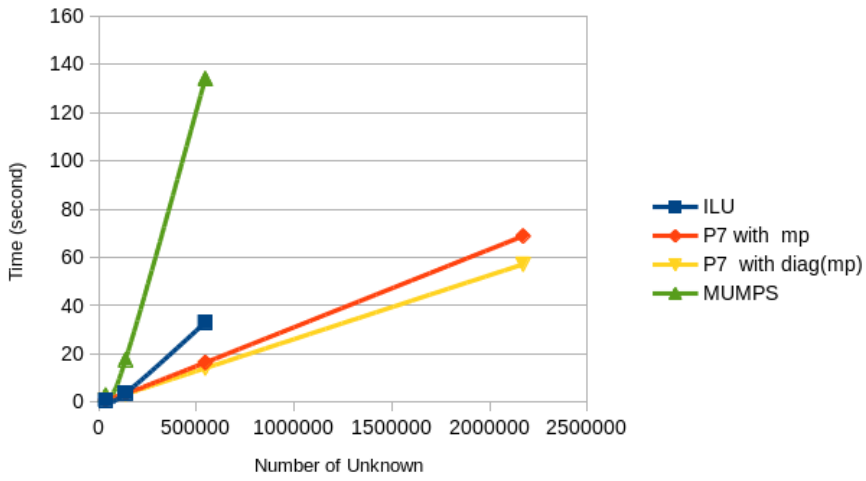
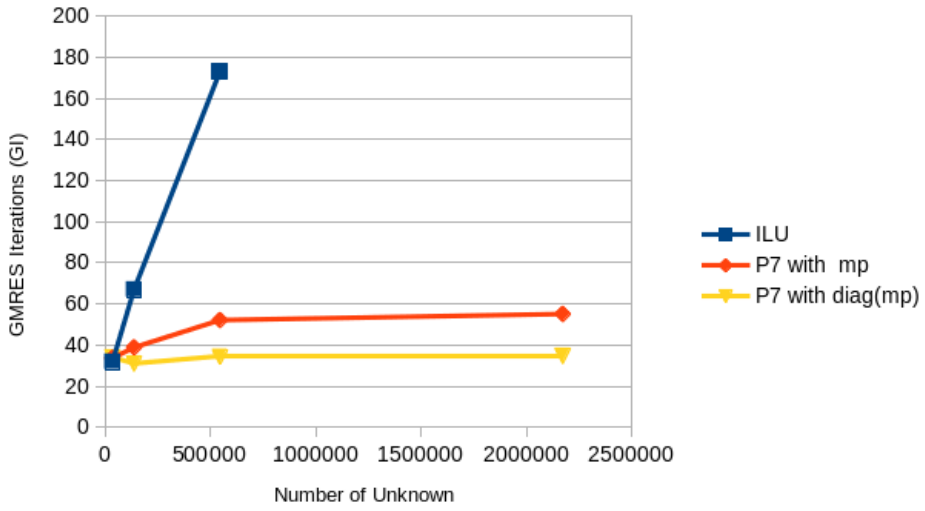


Figure 6.8: Comparison between the number of GMRES iterations using different preconditioners: ILU, P7 with mp and P7 with diag(mp), with the GMRES relative tolerance $1e-3$, the results are taken from Table 6.19, Table 6.20 and Table 6.21.



6.3.2.2 Unstructured Grids

From the previous subsection we concluded that, the preconditioner P7 with the diagonal of mp achieved the best performance. However all of the examples considered these were on regular grids on a square domain. In [57] the computational domains are approximately circular and the grids are completely unstructured (as illustrated in Figure 5.6). So, in this subsection the linear equations systems are solved on unstructured grids, covering a circular domain, using the best preconditioner, P7 with the diagonal of mp, and compared with the MUMPS solver.

We start with the same grid from [57] that has the number of unknowns 76237. We then apply uniform mesh refinements to obtain a sequence of finer and finer grids, each with approximately four times more unknowns than the previous.

Table 6.23 presents the running time that is needed to solve the linear system using MUMPS. As mentioned above, MUMPS can not solve the large problems when we sequentially halve the grid spacing. Moreover the computational time grows by factor of 7. This means that the solution required at best $\mathcal{O}(NlogN)$ complexity. By replacing the solver to P7 preconditioned GMRES, where the relative tolerance is $1e-3$, the performance of the solution improved significantly and that is shown by comparing Table 6.23 and Table 6.24. In Table 6.24 the running time scales linearly with problem size. Also, the number of iterations in this domain is almost fixed. When we reduced the tolerance to $1e-6$. the computational time behave almost optimally, however, the number of GMRES iterations slightly increased (see Table 6.25).

Table 6.23: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using MUMPS on fully unstructured grids: N is the number of unknowns.

N	average Time per step (sec.)
76237	9.01380657199
302252	64.09243820
1203643	-

Table 6.24: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using P7 (with mp=diag(mp)) preconditioned GMRES on fully unstructured grids: GI is GMRES iterations (the relative tolerance for GMRES is $tol = 1e-3$), N is the number of unknowns.

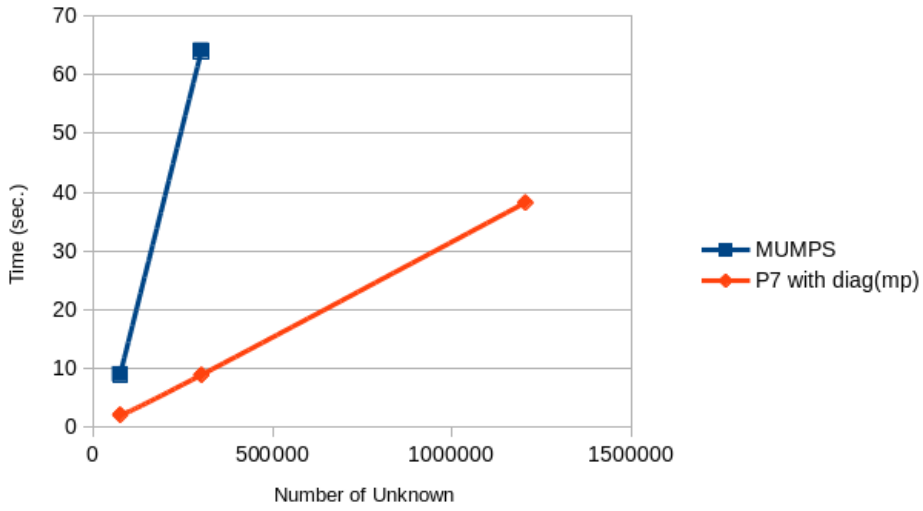
N	average GI	average Time per step (sec.)
76237	35.138	2.12991429
302252	39.811	8.9229566389
1203643	41.901	38.244552119

Table 6.25: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system using P7 (with mp=diag(mp)) preconditioned GMRES on fully unstructured grids: GI is GMRES iterations (the relative tolerance for GMRES is $tol = 1e-6$), N is the number of unknowns.

N	average GI	average Time per step (sec.)
76237	88.933	3.492189506
302252	101.864	15.40491242999
1203643	117.727	76.8860694099

To conclude, P7 preconditioned GMRES is the best solver, achieving almost optimal solution times on unstructured grids. Moreover, the new solution scheme reduced the memory that is required to solve the problem. The graph in Figure 6.9 shows the improvement in the running time when we replaced the sparse direct solver MUMPS by our new preconditioner P7 with GMRES solver.

Figure 6.9: Comparison between running times using different solvers: MUMPS (blue line) and P7 preconditioned GMRES (red line) with relative tolerance $1e-3$, the results are taken from Table 6.23 and Table 6.24.



6.4 Discussion

The achievements of this chapter are twofold: firstly, to develop and apply a new block-preconditioned algorithm to efficiently solve the discrete algebraic system resulting from Taylor-Hood FEM approximation of the momentum balance equations in [57]. This allows us to replace the MUMPS sparse direct solver used in [57], and therefore solve systems of larger size and with greater efficiency. Secondly, it is observed experimentally that the preconditioner that has been developed and implemented (i.e. P7) shows close to optimal convergence behaviour.

From the results presented in the previous section, we note that the MUMPS solver is much slower than our preconditioned GMRES. Moreover, MUMPS is unable to solve very large problems due to its less efficient use of memory. The next chapter includes further computational experiments to assess the computational and memory costs for the whole tumour model with our proposed improvements to the solver.

Chapter 7

Analysis of the Complete Model

In Chapter 5 and Chapter 6 we presented algorithms for the efficient solution of the algebraic systems that arise from discretization of the nonlinear reaction diffusion equation system and linear momentum balance equations system, respectively, from the mathematical model, which is discussed in Chapter 2 Section 2.2. Our solver achieved optimal convergence behaviour for the nonlinear system by using an AMG preconditioned GMRES method at each Newton step, and also achieved almost optimal convergence behaviour for the linear system by using our new preconditioner, P7, with the GMRES method.

In this chapter, we consider the solution of the complete multiphase tumour model, incorporating each of our algorithmic improvements. Validation of our results against existing results in [57] is presented in Section 7.1. Following that selected simulations, with description of our results, for regular grids and unstructured grids are shown in Section 7.2. In Section 7.3, the accuracy of our solutions is discussed. The computational time and the memory requirement for solving the whole model are presented in Section 7.4.

7.1 Validation Against Hubbard and Byrne

This section presents the validation of our results against the existing results in [57]. We validate our results, which are obtained using preconditioned iterative methods for the linear systems, by comparing with the results that are obtained using the sparse direct solution package MUMPS in [57].

It is worth mentioning at this point that we found some errors in the Fortran code written by Hubbard and Byrne. These errors are found in the block structure of the matrix for the linear system after discretization of the momentum equation. Figure 7.1 and Figure 7.2 show,

respectively, the block structure of the matrix with the errors and the block structure of the matrix after the corrections. It can be seen that the errors come from mistakes in the location of the coefficients relating to the y-velocity of the tumour phase. In the first block of rows the coefficients multiplying the y-velocity of the tumour phase should be zero whereas in the second block of rows these coefficients should not be zero. Also, there are corresponding errors in the third and fourth blocks of rows which relate to the transpose blocks. These errors were verified by the original author [55]. Figure 7.3 shows the original results in the first row, which are the same results as in Figure 5 in [57], and the results obtained after the corrections in the second row. There are clear qualitative differences in the results. Our new results are shown in the third and fourth rows with GMRES tolerance $1e-6$ and $1e-3$, respectively, which are using to solve both momentum equation and the reaction-diffusion equation. It can be seen that overall, our results behave similarly to the results that are obtained using MUMPS. The value of the GMRES tolerance has a clear effect on the results. When the smaller tolerance is used the solution is much closer to the solutions arising from using MUMPS solver.

Figure 7.1: Structure of sparse matrix A with errors

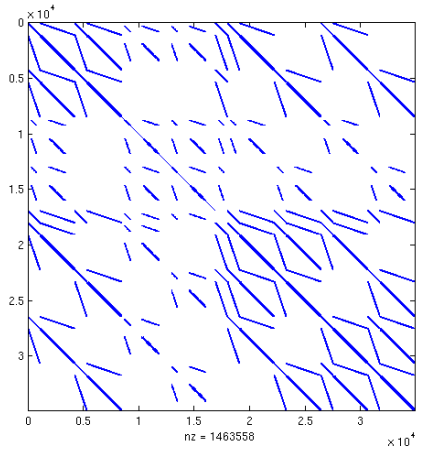


Figure 7.2: Structure of sparse matrix A after the correction

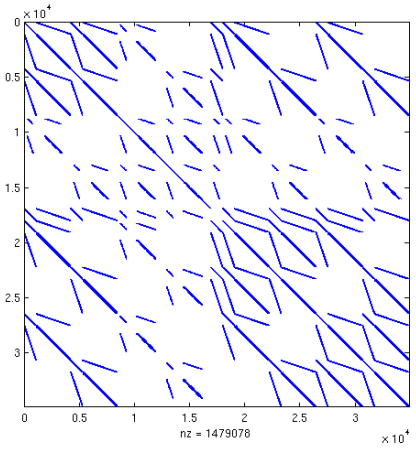


Figure 7.3: Comparison between different solutions for growth of the volume fraction of tumour cells θ_2, arranged from the top row to the bottom row: MUMPS in [57], MUMPS (corrected), GMRES with $tol = 1e-6$ and GMRES with $tol = 1e-3$. Time increases from left to right, t=100, 200 and 300.

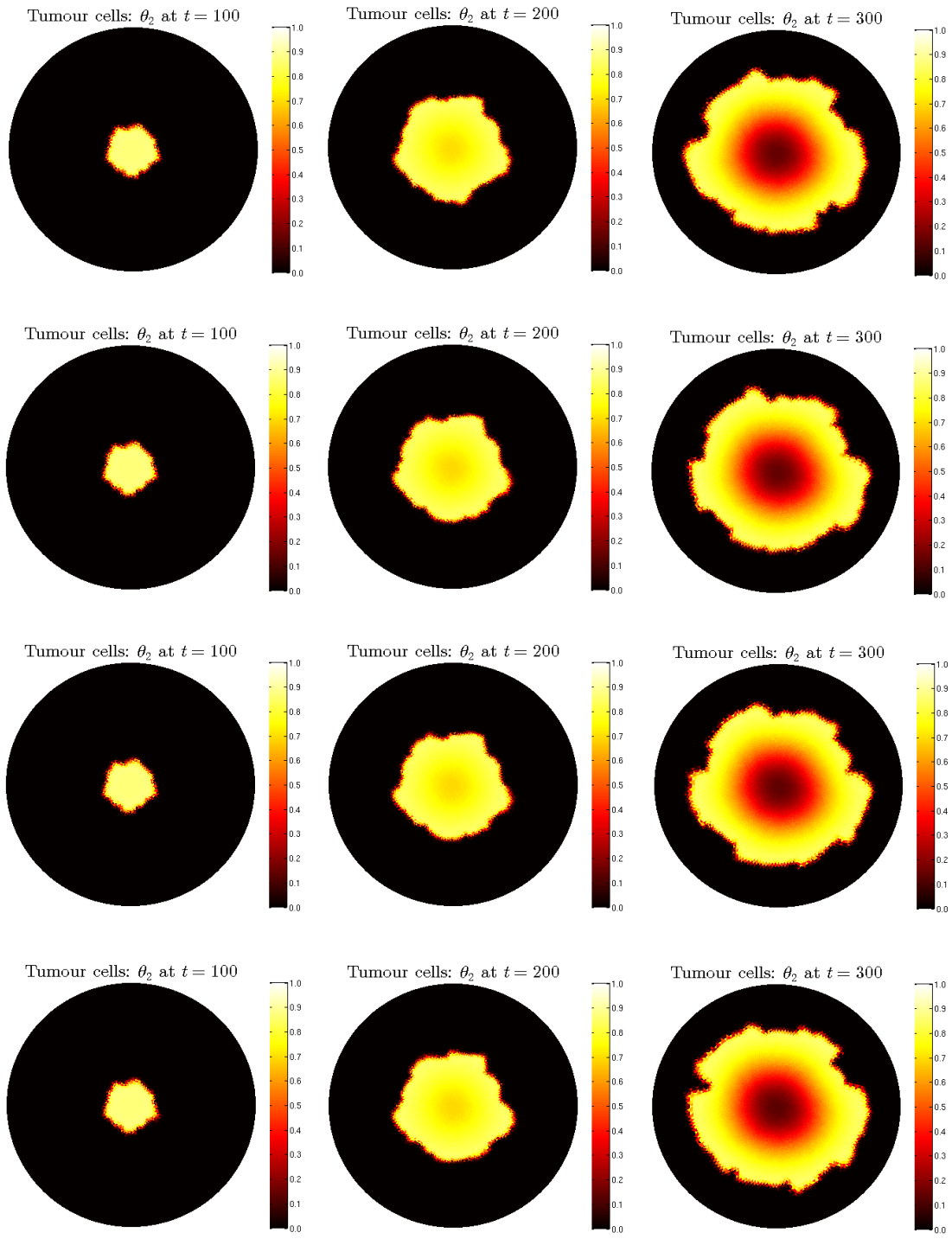


Table 7.1: Comparison between the values of the volume fractions of all phases on unstructured domain and at t=300, using different solvers.

volume-fractions	original result (MUMPS)	new result (MUMPS)	our result(1e-6)	our result(1e-3)
θ_1	0.0012498	0.0012467	0.0012467	0.0012498
θ_2	0.8810535	0.8810567	0.8810565	0.8812581
θ_3	0.0003866	0.0003868	0.0003868	0.0003878
θ_4	0.1173101	0.1173099	0.1173100	0.1171043

For further validation Table 7.1 presents the values of the volume fractions for each phase using different methods. These values are at the centre of the domain after a simulation to $t = 300$. From this table it can be observed that our results with GMRES tolerance $1e-6$ are almost identical to the results that are obtained using MUMPS. The maximum value of the tumour volume fraction is in the range 0.8 and 0.9 and this corresponds to what was observed in [57].

7.2 Further Numerical Results

As mentioned in Section 4.3, we focus our discussion on one of the three scenarios that are used in [57] to simulate tumour growth, which is seeding a single cluster of tumour cells in the centre of the computational domain. This scenario is considered on two different computational domains: a square with a regular triangular grid (as shown in Figure 5.3) and a circle with an unstructured triangular grid (as shown in Figure 5.6). The boundary conditions, the initial conditions and the parameter values that are used to simulate this model are as introduced in Section 4.3. Also, selected original results from [57] are presented in Subsection 4.3.2.

The numerical simulations in Figure 7.5 and Figure 7.9 show typical volume fractions of each phase on the square and circle domains respectively. In Subsection 4.3.2 we described the original results physically. The behaviour of our results physically is qualitatively the same as for the original results. It can be noted however the tumour has a different shape depending on the structure of the domains. In particular, the orientation of the triangles in the regular grid on the square domain has a noticeable influence on the spread direction of the tumour. When the irregular grid is used on the circular domain the tumour has an incomplete circle shape since the grid is not radially symmetric. However, overall the tumour growth has similar behaviour in these two domains. The volume fraction of tumour cells reaches a maximum value in the early time, which is in the range between 0.8 and 0.9 (see Figure 7.4), and then the tumour cells in the centre start dying, leaving a ring of tumour to spread far from the centre of the domain.

Figure 7.6, Figure 7.7 and Figure 7.8 illustrate the evolution of the phase fluxes, pressures and the nutrient concentration on regular square grid, respectively. In Figure 7.10, Figure 7.11 and Figure 7.12 the evolutions of the phase fluxes, pressures and the nutrient concentration on

an unstructured circle grid are presented.

As mentioned in Subsection 4.3.2, the proliferation and death rates of the tumour cells are assumed to be double and half the values of the proliferation and death rates of the healthy cells respectively (see Table 4.1). So, the tumour cells grow faster and die slower than the healthy cells. In this case, the tumour cells absorb the extracellular material during their growth. This leads to a fall in the extracellular material θ_4 and therefore leads to decreasing the healthy cells' birth rate. Also, when the tumour cells grow faster than the healthy cells that leads to the tumour cells pushing the healthy cells in front and replacing the volume by the extracellular material (as illustrated in Figure 7.6 on a regular grid and Figure 7.10 on an unstructured grid). Furthermore, the high tumour cells density generate high pressures which leads to the occlusion of the blood vessels and therefore restricts the supply of the nutrient inside the tumour (cf. Figure 7.8 on a regular grid and Figure 7.12 on an unstructured grid).

Figure 7.4: The evolution of the tumour cells on the square domain along $y = 0$ with increasing time from lift to right, t=100, 200 and 300. The grid size is 33×33 with the number of unknowns in the momentum system equal to 34889.

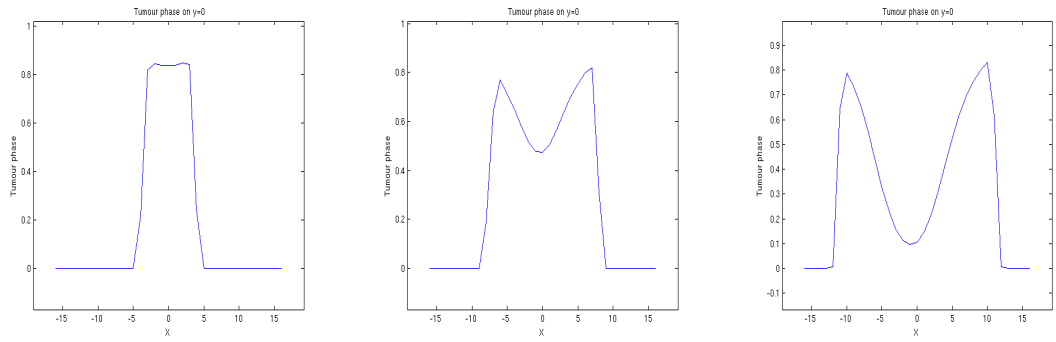


Figure 7.5: Evolution of the volume fraction for each phase arranged from the top row to the bottom row: healthy cells θ_1, tumour cells θ_2, blood vessels θ_3 and extracellular material θ_4, with increasing time from left to right, t=100, 200 and 300. The grid size is 33×33 with the number of unknowns in the momentum system equal to 34889.

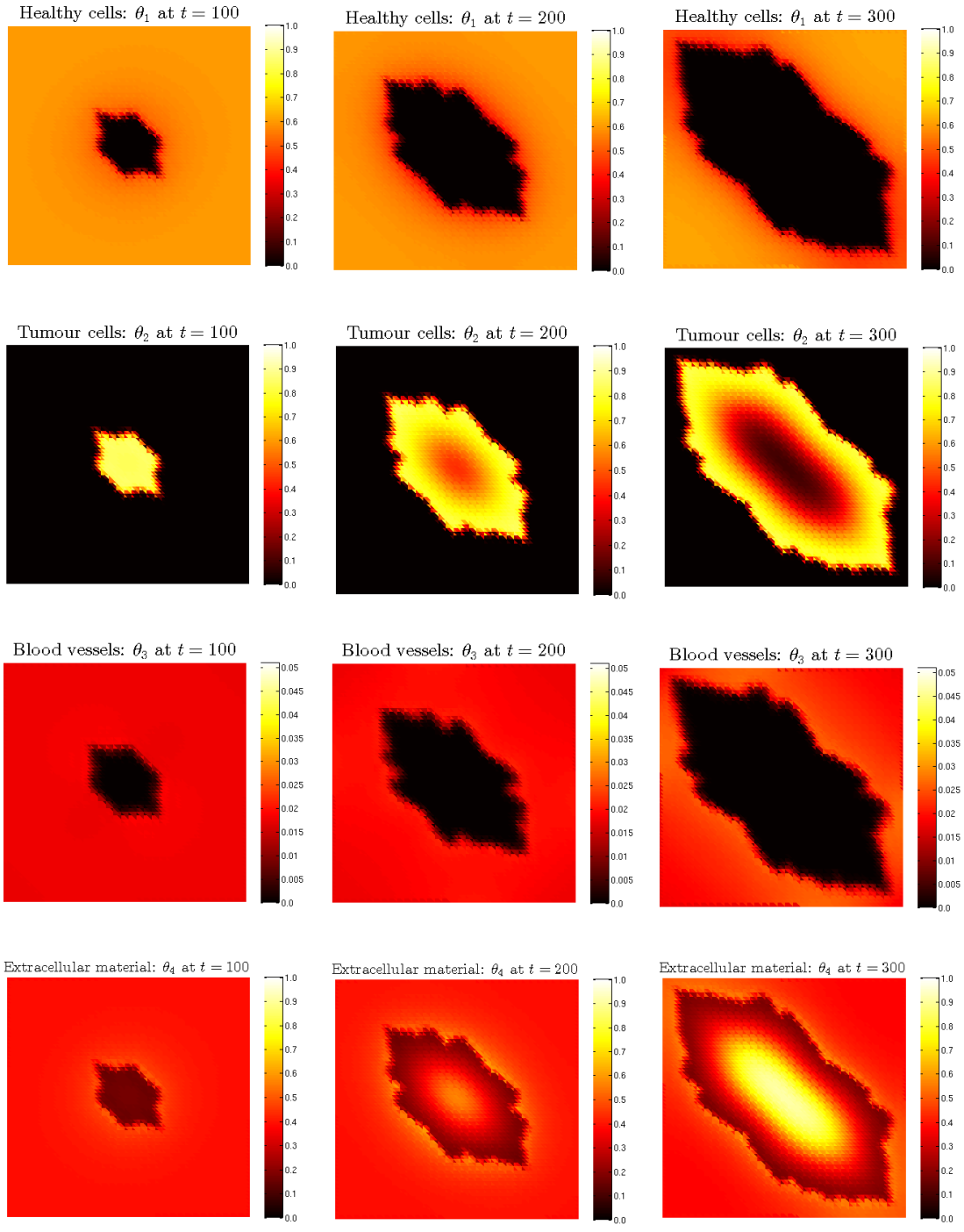


Figure 7.6: Evolution of the fluxes of phases, arranged from the top row to the bottom row: healthy cells $\theta_1\vec{u}_1'$, tumour cells $\theta_2\vec{u}_2'$ and extracellular material $\theta_4\vec{u}_4'$, with increasing time from left to right, t=100, 200 and 300. The grid size is 33×33 with the number of unknowns in the momentum system equal to 34889.

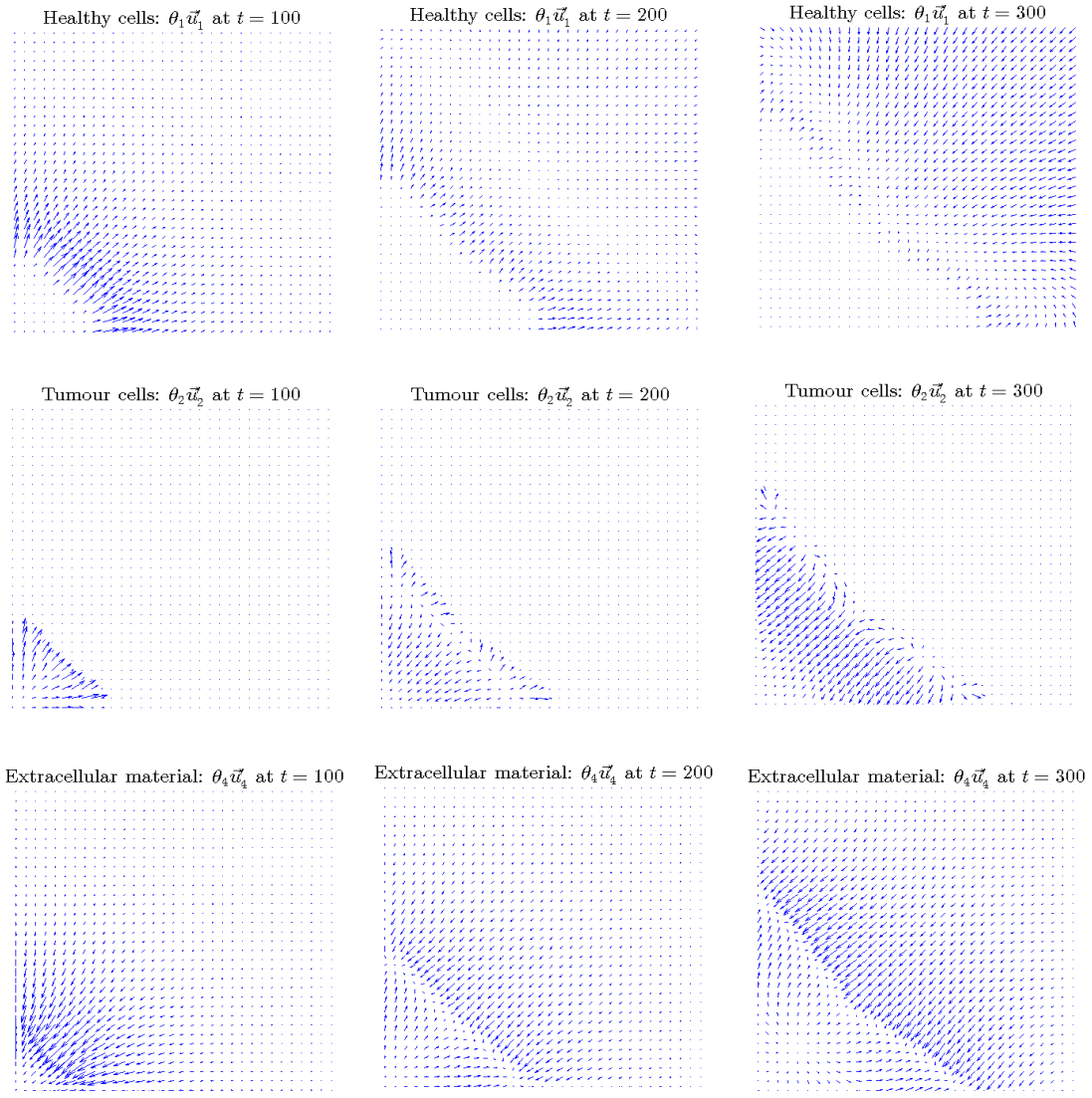


Figure 7.7: Evolution of the pressures arranged from the top row to the bottom row: healthy and tumour cells $p_1 = p_2$ and extracellular material (ECM) p_4, with increasing time from left to right, t=100, 200 and 300. The blood vessels pressure p_3 is zero in this model. The grid size is 33×33 with the number of unknowns in the momentum system equal to 34889.

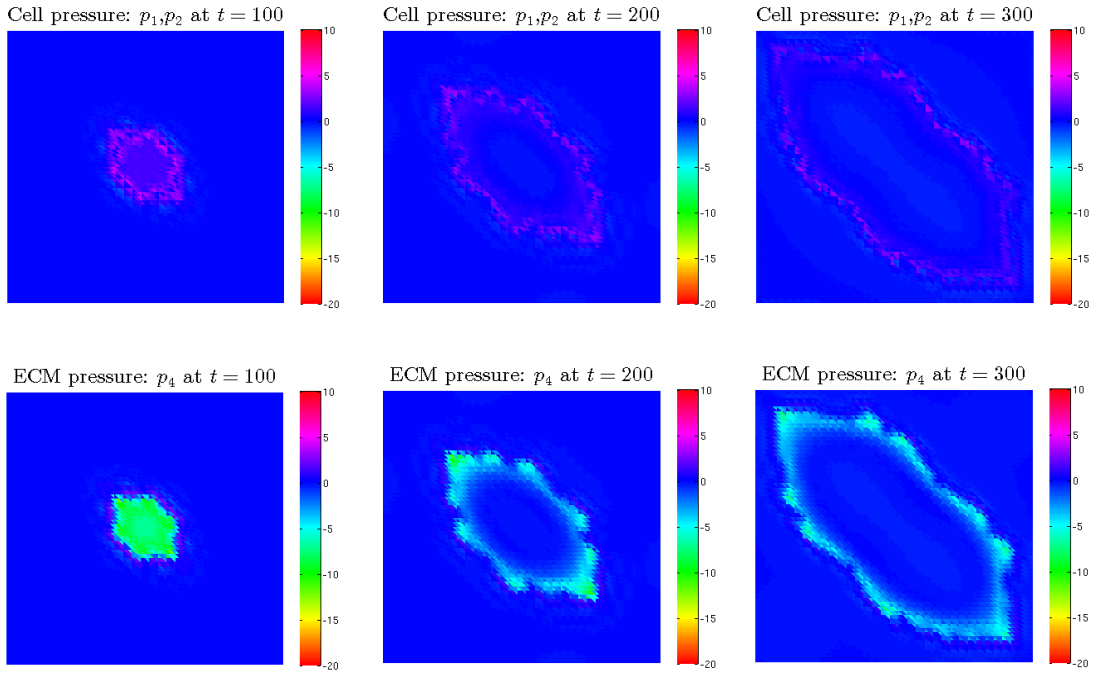


Figure 7.8: Evolution of the nutrient concentration c with increasing time from left to right, t=100, 200 and 300. The grid size is 33×33 with the number of unknowns in the momentum system equal to 34889.

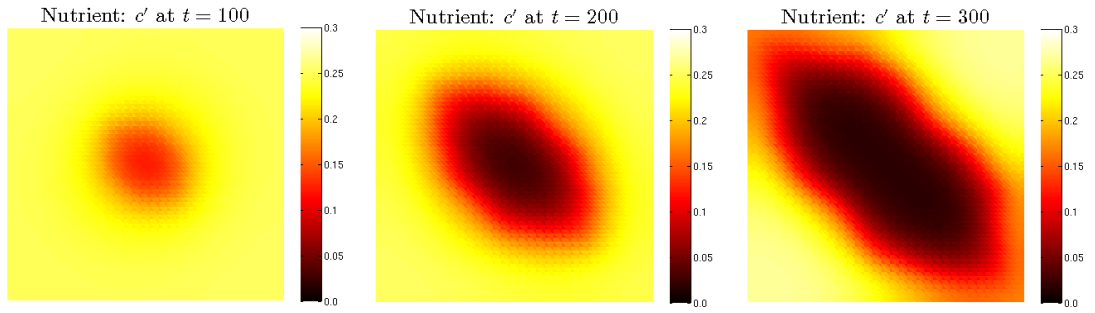


Figure 7.9: Evolution of the volume fraction for each phase arranged from the top row to the bottom row: healthy cells θ_1, tumour cells θ_2, blood vessels θ_3 and extracellular material θ_4, with increasing time from left to right, t=100, 200 and 300. The number of unknowns in the momentum system equal to 76237.

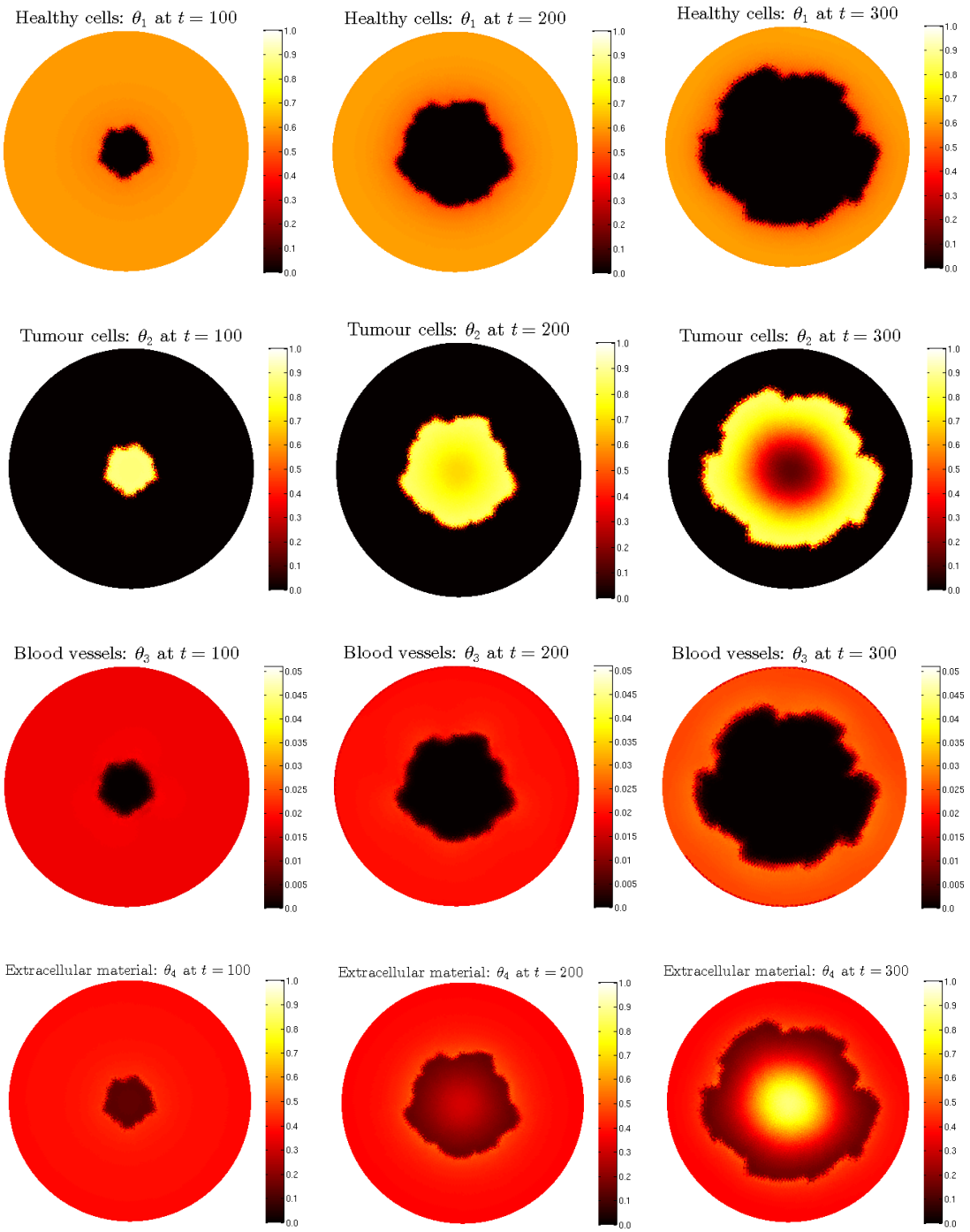


Figure 7.10: Evolution of the fluxes of phases, arranged from the top row to the bottom row: healthy cells $\theta 1\vec{u}_1'$, tumour cells $\theta_2\vec{u}_2'$ and extracellular material $\theta_4\vec{u}_4'$, with increasing the time from left to right, t=100, 200 and 300. The number of unknowns in the momentum system equal to 76237.

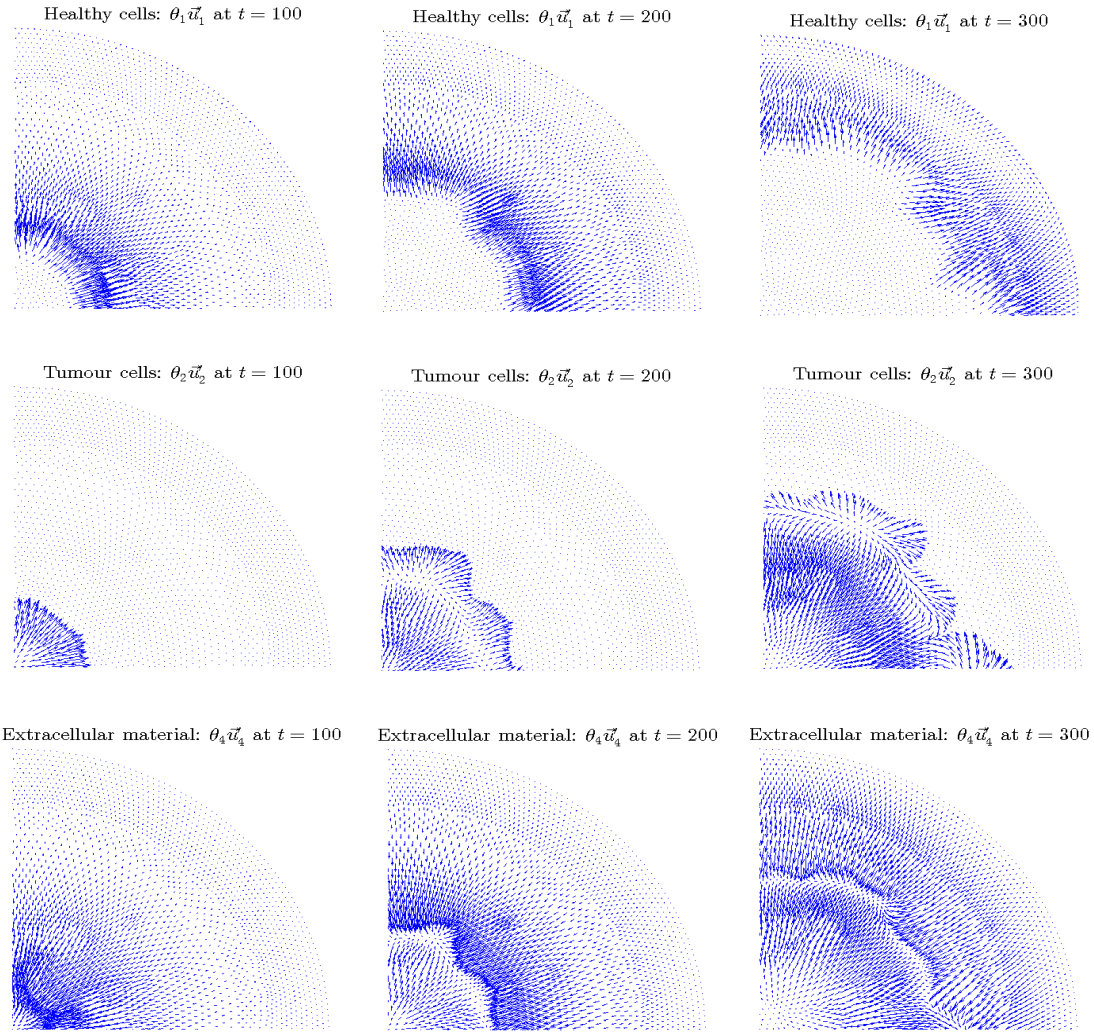


Figure 7.11: Evolution of the pressures arranged from the top row to the bottom row: healthy and tumour cells $p_1 = p_2$ and extracellular material (ECM) p_4, with increasing time from left to right, t=100, 200 and 300. The blood vessels pressure p_3 is zero in this model. The number of unknown in the momentum system equal to 76237.

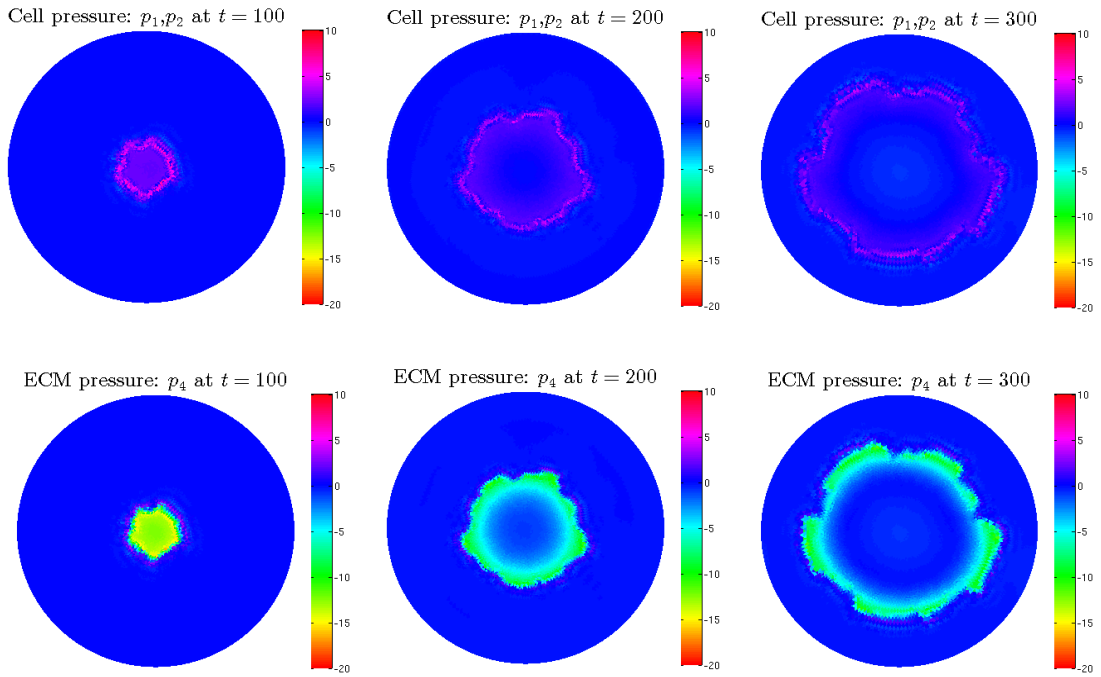
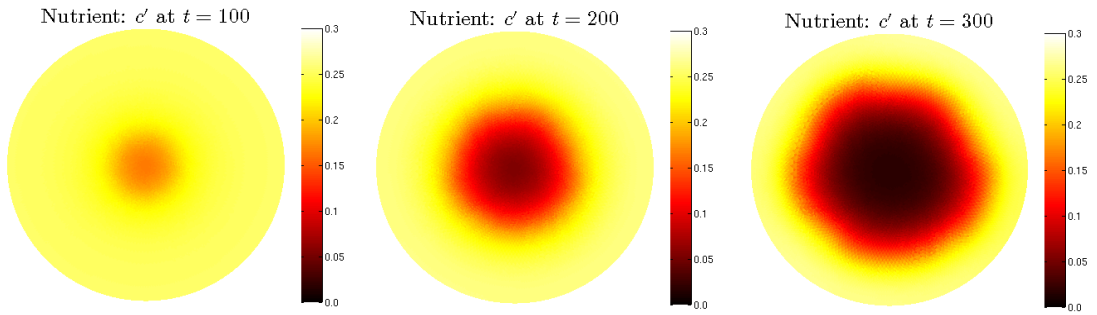


Figure 7.12: Evolution of the nutrient concentration c with increasing time from left to right, t=100, 200 and 300. The number of unknown in the momentum system equal to 76237.



7.3 Accuracy

In [57] the use of the sparse direct solver restricted the mesh size that it was possible to use. Hence no accuracy on mesh convergence tests here undertaken. We no longer have such a restriction therefore in this section the accuracy of our solutions is considered. We use sequences of the regular grids on a square domain to test the mesh convergence of the model and our proposed solver. We suggest new initial condition for θ_2 on the regular grids rather than the existing initial condition in (4.15) to ensure we are solving a comparable problem with identical initial condition on all grids. This initial condition can be written as:

$$\theta_2(x, y, t = 0) = \begin{cases} 0.05 \quad for \quad r \lesssim 1 \\ 0 \qquad\qquad\quad otherwise, \end{cases} \tag{7.1}$$

in which $r = max(\mid x \mid, \mid y \mid)$. So, the new initial condition is suggested to ensure that all grids of 33^2 or finer start initially with exactly the same value of θ_2 as illustrated in Table 7.2.

Table 7.2: Initial values of θ_2 with new initial condition in (7.1)

grid	N	$\sum \theta_2$
33^2	34889	0.2
65^2	137353	0.2
129^2	545033	0.2

Table 7.3 and Table 7.4 present our convergence results on sequences of regular grids. On each grid we can see the results converge as $\Delta t \longrightarrow 0$. However, between grids the convergence is less clear: by the time we reach 257^2 resolution $\max(\theta_2)$ appears to have converged, however the total mass of tumour cells may not yet be fully converged. Table 7.3 consider the maximum value of θ_2 and we can drawer similar conclusions. Note that the results in Table 7.3 and Table 7.4 are obtained using the GMRES tolerance $1e-3$, however using the GMRES tolerance $1e-6$ does not make a qualitative difference to these mesh convergence results. For example when $\Delta t = 0.015625$ in the grid 65^2, the total of θ_2 is equal 1.6466 and 1.6464 with the GMRES tolerances $1e-3$ and $1e-6$, respectively. Also, for the same grid and value of Δt, the maximum value of θ_2 is equal 0.4012 with both of the GMRES tolerances. Another example is taken in the grid 129^2 when using $\Delta t = 0.00390625$. The total values of θ_2 is 1.6418 and 1.6415 when using respectively the GMRES tolerances $1e-3$ and $1e-6$. In the same cases the maximum values of θ_2 is 0.4045 and 0.4046 when using respectively the GMRES tolerances $1e-3$ and $1e-6$.

Table 7.3: The convergence test on the regular grids, N is the number of unknowns, the relative GMRES tolerance is $1e-3$.

grid	N	Δt	step	$\max(\theta_2)$
33^2	34889	0.5	20	0.38232003
		0.25	40	0.39050775
		0.125	80	0.39461319
		0.0625	160	0.39666500
65^2	137353	0.125	80	0.39753173
		0.0625	160	0.39964982
		0.03125	320	0.40070918
		0.015625	640	0.40123892
129^2	545033	0.03125	320	0.40372176
		0.015625	640	0.40419875
		0.0078125	1280	0.40438966
		0.00390625	2560	0.40448361
257^2	2,171,401	0.0078125	1280	0.40355711
		0.00390625	2560	0.40369154
		0.00195312	5120	0.40375748
		0.00097656	10240	0.40379115

Table 7.4: The convergence test on regular grids, N is the number of unknowns, the relative GMRES tolerance is $1e-3$, $\sum \theta_2$ is the total of the volume fraction of tumour cells.

Grid	N	Δt	step	$\sum \theta_2$
33^2	34889	0.5	20	1.56920246
		0.25	40	1.609637076
		0.125	80	1.63047227
		0.0625	160	1.64105226
65^2	137353	0.125	80	1.62850286
		0.0625	160	1.63880276
		0.03125	320	1.64398629
		0.015625	640	1.64658639
129^2	545033	0.03125	320	1.63750621
		0.015625	640	1.63998837
		0.0078125	1280	1.64123225
		0.00390625	2560	1.64185302
257^2	2,171,401	0.0078125	1280	1.63480270
		0.00390625	2560	1.63540101
		0.00195312	5120	1.63569484
		0.00097656	10240	1.6358447

What we are able to concluded from these numerical experiments is that the mesh resolutions used in [57] are not sufficient to ensure mesh independence of their results. Their use of sparse direct linear algebra means that the CPU and memory requirements to remedy this would have been prohibitive, however the optimal iterative approaches developed in this work do allow much finer resolution.

7.4 Analysis of Totel Computational Cost

This section considers the computational and memory costs for the whole tumour model with our proposed approach based on preconditioned GMRES with AMG preconditioning for solving the discrete system from the reaction diffusion equation system (2.6) and our new preconditioning, P7, for solving the discrete system from the momentum equations system (2.4).

In the following subsections all numerical experiments use a time step size of 0.25, and the number of time steps taken is 1000. We refer back to Section 4.3, which defines the boundary conditions, the initial conditions and the parameter values that are used.

7.4.1 Regular Grids

This subsection shows a selection of results for solving the whole model on sequences of regular square grids (as illustrated in Figure 5.3). Each table includes the number of unknowns, the memory percentage (as reported by top) and the computational time (in minutes) required.

Table 7.5 presents CPU time and the memory requirement for solving the model using the sparse direct solver MUMPS. It can be seen that, both the time and the memory usage increased rapidly when the number of unknowns increased and required at best $\mathcal{O}(NlogN)$ complexity. As mentioned before, this model is unable to solve large problems of 257^2 or greater, when using MUMPS solver, on the test computer within the available memory.

From Table 7.6 it can be observed that using the ILU preconditioned GMRES solver for both momentum and reaction-diffusion equations (with drop tolerance of $1e-2$ and relative convergence tolerance $1e-3$) leads to reducing the memory and the CPU time compared with MUMPS solver. However, it is still requires relatively large memory due to the memory requirement of the incomplete factorization.

The cost of our solver is presented in Table 7.7 and Table 7.8. In both the cases, the memory required for solving the model using P7 (with mp) or P7 (with diagonal of mp) preconditioned GMRES (with relative tolerance $1e-3$) are similar, while the CPU time is better when P7 (with diagonal of mp) is used. We can confidently say that our solver leads to dramatic saving of computational time and memory compared with MUMPS solver. Furthermore, the time and the memory requirement behave almost optimally as $\mathcal{O}(N)$.

For further investigation, the tolerance of the GMRES solver is reduced to $1e-6$ and the results are shown in Table 7.9. By compare this table and Table 7.8 it can be seen that, when the value of the GMRES tolerance is reduced the memory and the CPU time is slightly increased. However, the time and the memory requirements still scale almost linearly as $\mathcal{O}(N)$.

Table 7.5: The CPU time and percentage of memory cost for solving the whole model on regular grids using the MUMPS solver.

Grid	MEM (%)	Time
33^2	1.5	49m42s
65^2	7.3	301m12s
129^2	35.8	2293m7s
257^2	-	-

Table 7.6: The CPU time and percentage of memory cost for solving the whole model on regular grids using ILUd(10^{-2}) preconditioned GMRES (relative tolerance $1e-3$).

Grid	MEM (%)	Time
33^2	1.4	13m30s
65^2	4.1	55m33s
129^2	17.2	577m19s
257^2	-	-

Table 7.7: The CPU time and percentage of memory cost for solving the whole model on regular grids using preconditioned GMRES: the linear system solved using p7 with mp, and the nonlinear system solved using AMG preconditioning, with GMRES relative tolerance $1e-3$.

Grid	MEM (%)	Time
33^2	1	15m23s
65^2	3.5	62m51s
129^2	12.5	285m13s
257^2	48.5	1194m31s

Table 7.8: The CPU time and percentage of memory cost for solving the whole model on regular grids using preconditioned GMRES: the linear system solved using p7 with diagonal of mp, and the nonlinear system solved using AMG preconditioning, with GMRES relative tolerance $1e-3$.

Grid	MEM (%)	Time
33^2	1	15m16s
65^2	3.4	58m22s
129^2	12.7	246m53s
257^2	48.5	998m10s

Table 7.9: The CPU time and percentage of memory cost for solving the whole model on regular grids using preconditioned GMRES: the linear system solved using p7 with diagonal of mp, and the nonlinear system solved using AMG preconditioning, with GMRES relative tolerance $1e-6$.

Grid	MEM (%)	Time
33^2	1.1	25m9s
65^2	3.7	98m7s
129^2	13.8	444m17s
257^2	52.7	1985m17s

Figure 7.13 shows execution times for a series of problems of different size solved using the four solvers. Our preconditioiner P7 (with diagonal of mp) shows the best results, followed by p7 (with mp), then ILU preconditioned GMRES method with the MUMPS solver being the slowest. Overall, using our solver has achieved the best performance in terms of CPU time.

As shown in Figure 7.14 the MUMPS solver also has significantly greater memory usage than the other three solvers. With MUMPS and ILU preconditioned GMRES solvers 129^2 is largest problem we are able to solve on a standard desktop computer when we sequentially halve the grid spacing. However, with our new solver we are able to solve the larger problem 257^2 due to the much reduced memory requirement.

Figure 7.13: Comparison between the CPU time required for solving the model using different methods: MUMPS, ILU preconditioned GMRES and our solver: the results are taken from Table 7.5, Table 7.6, Table 7.7 and Table 7.8.

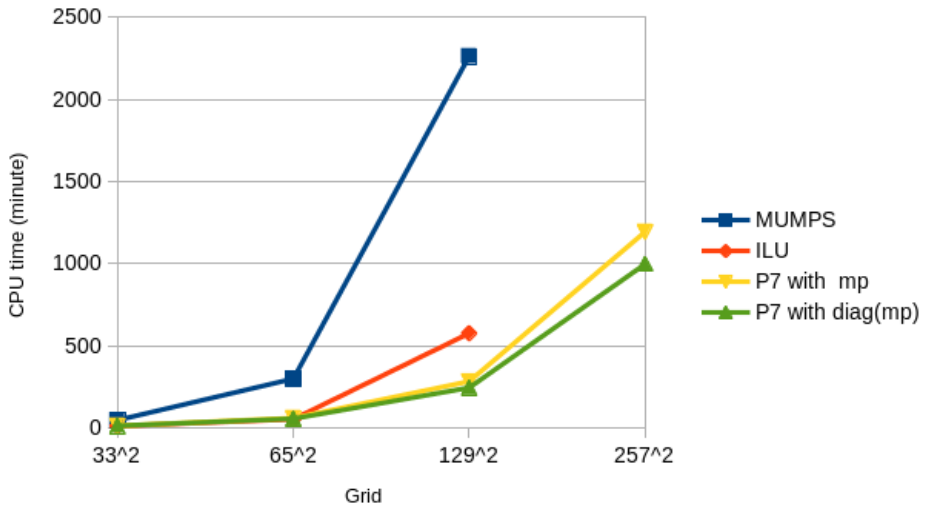
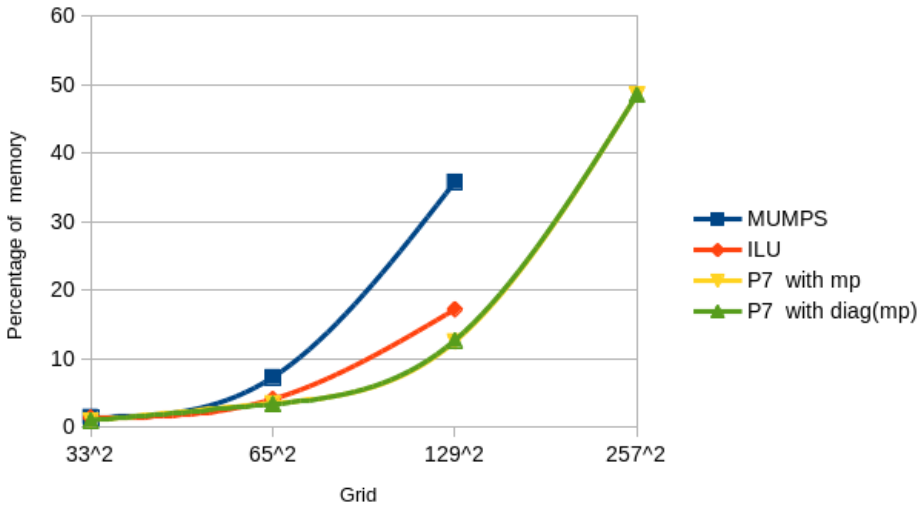


Figure 7.14: Comparison between the memory requirement for solving the model using different methods: MUMPS, ILU preconditioned GMRES and our solver: the results are taken from Table 7.5, Table 7.6, Table 7.7 and Table 7.8.



7.4.2 Unstructured Grids

This subsection presents the cost of our solver compared with the MUMPs solver on unstructured grids (as shown in Figure 5.6). As mentioned before, we begin with the same grid from [57], on which the number of unknowns is 76237. Then, uniform mesh refinements are applied to obtain a sequence of finer grids.

The MUMPS method required the CPU times and memory as shown in Table 7.10. As expected, this solver dramatically increased the time and the memory usage when the grid size increased. As mentioned above, MUMPS can not be used to solve large problems when we sequentially halve the grid spacing.

Table 7.11 and Table 7.12 illustrate the computational time and the memory cost of our solver with the GMRES tolerances $1e-3$ and $1e-6$ respectively. It can be seen that these achieved much faster time and much less memory compared the results using MUMPS in Table 7.9. Also, our solution method enables us to solve larger problems than when using the MUMPS solver.

Table 7.10: The CPU time and percentage of memory cost for solving the whole model on fully unstructured grids using the MUMPS solver: N is the number of unknowns in the discrete momentum system.

N	MEM (%)	Time
76237	3.3	194m58s
302252	18.4	1084m4s
1203643	-	-

Table 7.11: The CPU time and percentage of memory cost for solving the whole model on fully unstructured grids using preconditioned GMRES: the linear system solved using p7 with diagonal of mp, and the nonlinear system solved using AMG preconditioning, with GMRES relative tolerance $1e-3$: N is the number of unknowns in the discrete momentum system.

N	MEM (%)	Time
76237	2.4	37m22s
302252	8	156m47s
1203643	27.5	672m28s

Table 7.12: The CPU time and percentage of memory cost for solving the whole model on fully unstructured grids using preconditioned GMRES: the linear system solved using p7 with diagonal of mp, and the nonlinear system solved using AMG preconditioning, with GMRES relative tolerance $1e-6$: N is the number of unknowns in the discrete momentum system.

N	MEM (%)	Time
76237	2.6	59m48s
302252	8.9	263m16s
1203643	29.7	1319m47s

Figure 7.15 illustrates the CPU time required for solving the model using the MUMPS solver and our solver with using P7 (with the diagonal of mp). It is clear that our solver has achieved the best performance in terms of CPU time and significantly reduced the time required for solving the model compared with MUMPS.

Figure 7.16 shows that the memory needed for solving the model using the MUMPS solver is much more than the memory needed when using our solver. Because of that the MUMPS solver is unable to solve on the larger grid when we sequentially halve the grid spacing. Our solver is able to solve the larger problem due to its much reduced memory requirements for solving the model.

Figure 7.15: Comparison between the CPU time requirement for solving the model using MUMPS our solver: the results are taken from Table 7.9 and Table 7.11.

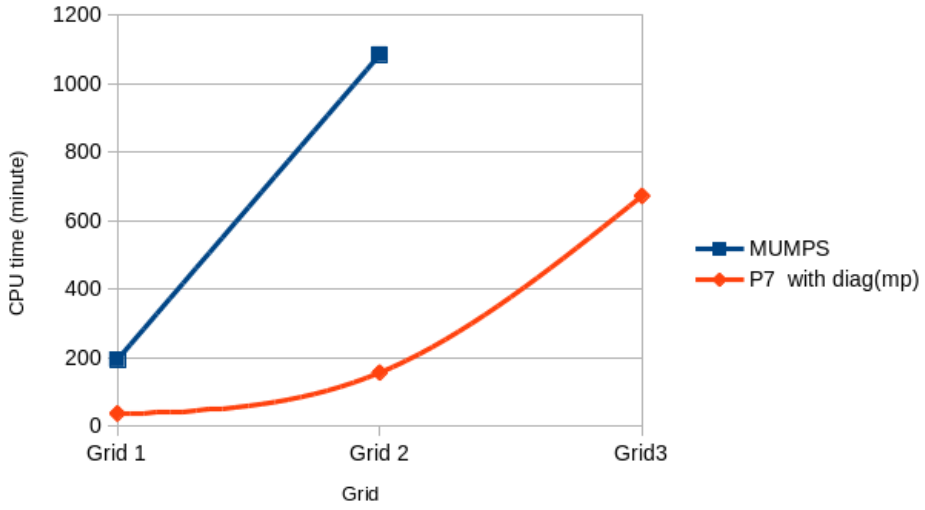
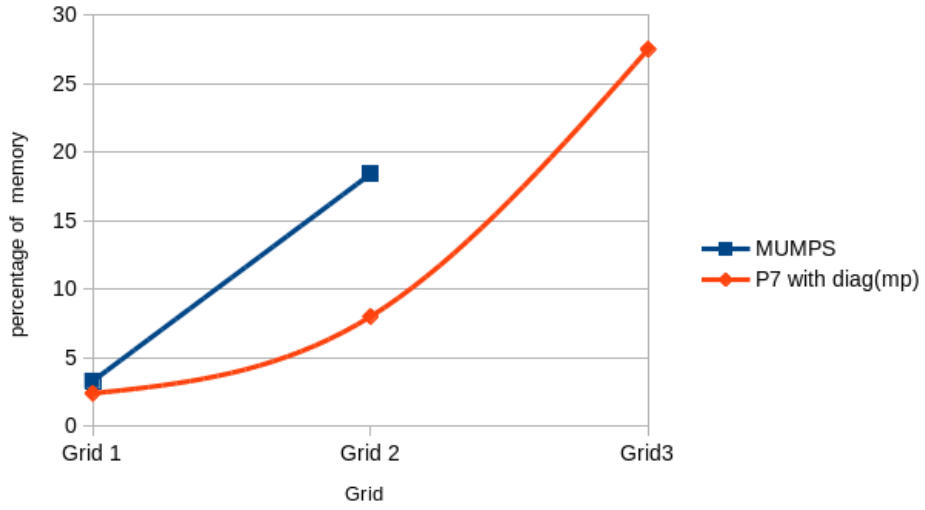


Figure 7.16: Comparison between the memory requirement for solving the model using MUMPS and our solver: the results are taken from Table 7.9 and Table 7.11.



7.5 Discussion

In this chapter the validation of our results against the existing results in [57] have been presented. We concluded that our model provided an accurate solution, compared with the existing codes. Further simulation results are shown and discussed in Section 7.2, before the accuracy and efficiency of the solver were considered in Section 7.3 and Section 7.4 respectively. We concluded that our results were computed with a dramatic saving of computational time and memory usage compared with the MUMPS solver. Moreover, we demonstrated that the additional levels of mesh resolution that are permitted as a result of this improved efficiency do indeed lead to more accurate simulations.

In the following chapter this basic multiphase tumour model is now extended to consider drug delivery and the inclusion of additional phases. We then show that our preconditioning strategy may be extended to these models with more than four phases.

Chapter 8

Generalization to the Model and the Numerical Methods

In the previous chapters the multiphase modelling of vascular tumour growth were studied, based upon the model introduced in [57]. This model includes four phases: healthy cells, tumour cells, blood vessels and extracellular material. In this chapter we modify the mathematical model described in Section 2.2 by adding a new governing equation representing the delivery and diffusion of a drug. We then extend the model further by increasing the number of phases from four to five. We begin with modelling the drug diffusion, and study the impact of the drug on the tumour growth in Section 8.1. Then, in Section 8.2, the modification of the multiphase model to include an additional phase is presented. The main objective in this chapter is extending our momentum equation preconditioner, P7, which is discussed in detail in the previous chapters, and examining its efficiency when generalized to this five phase system.

8.1 Drug Delivery

The tumour is a cluster of out-of-control cell growth in healthy tissue that can be benign or malignant. Many tumour treatments are based upon chemotherapy, and the effectiveness of these treatments depends on the type, and the biological properties, of the tumour as well as the pharmacology of the drug [49, 81]. Furthermore, the efficacy of tumour treatments also depends on the drug concentration. The aim of delivering drug to the tumour cells is to kill enough tumour cells to eradicate the tumour or to prevent cell division of the tumour cells.

Generally, the drug delivery in a tumour can be divided into three major stages: supply, flux (or movement) and consumption [77]. In the first stage, supply of drug determines the drug

delivery via the blood vessels in the tumours. This supply depends on the dose of the drug and on the pharmacokinetics (PK). In the second stage the drug is transported via diffusion through the mass of the tumour. The third stage is consumption, which involves the metabolism of the drug, binding to cellular material and subsequent sequestration [49, 77].

Mathematical and computational models of drug delivery may be used to predict how the drug is transported to tumour cells by blood vessels and to understand these bio-transport mechanisms [58, 100]. Moreover, models can investigate the effect of the drug on the size of the tumour. In recent years, there has been an increasing amount of research that studies the properties of delivering the drug to the tumours such as [100], [49], [58], [81] and [102].

In [59], Jackson and Byrne developed a mathematical model that describes a spherically-symmetric vascular tumour and studied the response, and the reduction of vascular tumour, to the drug, which is described as blood-borne chemotherapeutic. The mathematical model contains a system of partial differential equations (PDEs) that describe the drug concentration and two different types of tumour cells: one with high drug susceptibility and the other with lower susceptibility to the drug. The principle of conservation of mass is applied to obtain these governing equations.

Groh et al. in [49] have developed mathematical and computational models to describe the delivery of drug from blood vessel to a tumour cord. Three different models are studied, two of them in one-dimension (which are a radially symmetric compartment model and a radially symmetric continuum model) and the other in two-dimensions (a discrete cell-centred model). Each of these models has the same drug binding model. Also three different pharmacokinetic (PK) profiles of the supplied drug are used to investigate their impact on the drug distribution. The radially symmetric compartment model is modified to two-dimension in [58] by Hubbard et al. In that paper an *in silico* model of the drug transport is developed, which models intra-tumoural drug penetration in a tumour cord. Moreover, they investigated the influence of changing the PK profiles and the parameters on the numerical results.

In the following subsection, a modification of the mathematical model of [57] to include drug delivery and diffusion is presented.

8.1.1 Mathematical Model

In this subsection we modify the mathematical model that is presented in Section 2.2. In addition to the three governing equations that are introduced in Section 2.2, we introduce a new governing equation (of similar form to the nutrient equation) that is included to simulate drug concentration, which is called d'. So, the reaction diffusion equation for the drug concentration is written as

$$\begin{aligned} D_d^* \nabla'^2 d' &= \theta_3(dv(t) - d') - k d_{6,1}^* \theta_1 d' - k d_{6,2}^* \theta_2 d' \\ &- k d_{7,1}^* \theta_1 \theta_4 \left(\frac{d'}{d_p^* + d'} \right) - k d_{7,2}^* \theta_2 \theta_4 \left(\frac{d'}{d_p^* + d'} \right) \end{aligned} \tag{8.1}$$

in which

$$D_d^* = \frac{D_d}{kd_5 L_0^2}, \quad kd_{6,1}^* = \frac{kd_{6,1}}{kd_5}, \quad kd_{6,2}^* = \frac{kd_{6,2}}{kd_5},$$
$$kd_{7,1}^* = \frac{kd_{7,1}}{d_v kd_5}, \quad kd_{7,2}^* = \frac{kd_{7,2}}{d_v kd_5}. \tag{8.2}$$

Here D_d^* is the drug diffusion coefficient, (kd_5, $kd_{6,1}$, $kd_{6,2}$, $kd_{7,1}$, $kd_{7,2}$) are pre-defined rate constants. d_p^* denotes a drug concentration parameter, and d_v is the drug concentration within the blood vessels. This equation is a nonlinear equation with a single variable d'. In equation (8.1), the drug, d', is supplied by the vasculature and consumed by the healthy cells and tumour cells. The drug supply to the tumour cells depends on pharmacokinetics (PK): dv(t) is a pharmacokinetic (PK) term which can be defined as

$$\text{dv(t)}=\begin{cases} 0 & \text{if } t < t_0 \\ \frac{(t-t_0)}{(t_{max}-t_0)}\ dmax & \text{if } t < t_{max} \\ \frac{(t_1-t)}{(t_1-t_{max})}\ dmax & \text{if } t > t_{max} \\ 0 & \text{if } t > t_1. \end{cases} \tag{8.3}$$

This term simulates the introduction of the drug to the system from $t = t_0$, with linear increase to a level dmax at $t = t_{max}$ and then linear reduction to zero by $t = t_1$ The drug is assumed to adversely affect the tumour phase through causing a decrease in the tumour cells birth rate and/or increase in the tumour cells death rate. This may be modelled via the following mass balance equation for θ_2:

$$\frac{\partial \theta_2}{\partial t'} + \vec{\nabla}'.(\theta_2 \vec{u}_2') = \underbrace{k_{1,2}^* \theta_2 \theta_4 (1-\alpha_1 d') \left(\frac{c'}{c_p^* + c'}\right)}_{\text{cell birth}} - \underbrace{k_{2,2}^* \theta_2 (1+\alpha_2 d') \left(\frac{c_{c_1}^* + c'}{c_{c_2}^* + c'}\right)}_{\text{cell death}} \tag{8.4}$$

in which α_1 and α_2 are pre-defined rate constants.

8.1.2 Extension of the numerical model

This subsection introduces the numerical scheme for solving the new governing equation (8.1). This equation is a steady nonlinear PDE. It is approximated using the same approach as for the nutrient equation, based upon a Galerkin finite element scheme with linear elements, to get:

$$\int_\Omega \omega^{l_f} D_d^* \vec{\nabla}^2 d \ d\vec{x} - \int_\Omega \omega^{l_f} q_d d\vec{x} = 0, \tag{8.5}$$

in which D_d^* denotes the drug diffusion coefficient, ω^{l_f} is a linear test function, defined on the finer mesh (i.e. the mesh that is used to update the volume fractions of the phases), and q_d is a source term, which is a nonlinear function of the the drug concentration d (i.e. the right-hand side of (8.1)).

Using integration by parts leads to

$$\int_{\partial\Omega} \omega^{l_f} D_d^* \vec{\nabla} c.\vec{n} ds - \int_{\Omega} \vec{\nabla}\omega^{l_f}.D_d^*\vec{\nabla} d d\vec{x} - \int_{\Omega} \omega^{l_f} q_d d\vec{x} = 0. \tag{8.6}$$

The drug concentration d is written using the trial functions as:

$$d \approx \sum_{k=1}^{N_{l_f}} d_k \omega_k^{l_f}. \tag{8.7}$$

Since we use the same scheme that is used to solve the nutrient diffusion equation (2.6), this FE approach leads to another nonlinear system of equations with a sparse Jacobian. As before, this is solved using Newton's method to linearise it, and AMG-preconditioned GMRES at each Newton iteration.

8.1.3 Numerical Method

AMG-preconditioned GMRES is used for solving the linearised discrete system that arises from the new equation (8.1). Furthermore, we used the same initial conditions, boundary conditions and the parameter values (in Table 4.1) that are presented in Section 4.3. For the new variable drug concentration the initial condition is chosen to be $d' = 0$ and the Neumann boundary condition for the system of equation (8.1) is imposed on the whole of the boundary Γ:

$$\vec{\nabla}' d'.\vec{n} = 0,$$

where $\vec{n}$ is the unit of outward-pointing normal to Γ. The additional parameter values used are shown in Table 8.1.

Table 8.1: The nondimensional parameter values used in equation (8.1).

Parameter	Symbol	Values
Healthy cell baseline (drug consumption rate)	$kd_{6,1}^*$	0.01
Tumour cell baseline (drug consumption rate)	$kd_{6,2}^*$	0.01
Healthy cell birth (drug consumption rate)	$kd_{7,1}^*$	0.1
Tumour cell birth (drug consumption rate)	$kd_{7,2}^*$	0.2
Cell birth rate dependence on drug	d_p^*	0.25
Drug diffusion coefficient	D_d^*	1.0

8.1.4 Numerical Results

In this subsection, we present typical numerical results using the new model, with drug delivery, to show the impact of the drug on the tumour cells growth. As noted above, the AMG-preconditioned GMRES method is used to solve the discrete system. For this method, we still make use of the software implementation that is available in Harwell Subroutine Library (HSL) [1]. This includes: HSL-MI20 for the AMG method and HSL-MI24 for the GMRES method.

Figure 8.1 and Figure 8.2 present the volume fractions of the tumour cells on the regular square and the unstructured circular grids, respectively. These figures show the difference between the tumour size with and without drug uptake. The parameters α_1 and α_2 in equation (8.4) are varied to study the impact of the drug on the tumour cells. We can observe that the model with the parameters $\alpha_1 = \alpha_2 = 1$ significantly reduces the tumour size compared with the tumour without drug uptake, when $\alpha_1 = \alpha_2 = 0$. Furthermore, we can investigate the difference between a drug that works by accelerating cell death compared to one that prevents cell division /birth (see for example when $\alpha_1 = 1, \alpha_2 = 0$ and compared with when $\alpha_1 = 0, \alpha_2 = 1$).

Figure 8.3 illustrates the total value of θ_2 with and without drug uptake. It can be seen that the solution value of θ_2 increases with the time when θ_2 is not affected by the drug ($\alpha_1 = \alpha_2 = 0$). The effect of the drug when using the PK terms given by equation (8.3) is also shown in this figure. With moderate drug uptake ($\alpha_1 = \alpha_2 = 0.5$) the rate of increase of the tumour mass reduces shortly after the drug levels starts to reduce from its maximum at $t = 105$. With stronger drug uptake ($\alpha_1 = \alpha_2 = 1.0$) the tumour mass actually reduces once sufficient drug levels are reached, with a subsequent increase after the maximum level has been passed.

Figure 8.1: Evolution of the volume fraction of tumour cells on a regular grid: we vary α_1 and α_2 in equation (8.4). All plots show solutions at the same time, t=300. The grid size is 33×33 with the number of unknowns in the momentum system is equal to 34889.

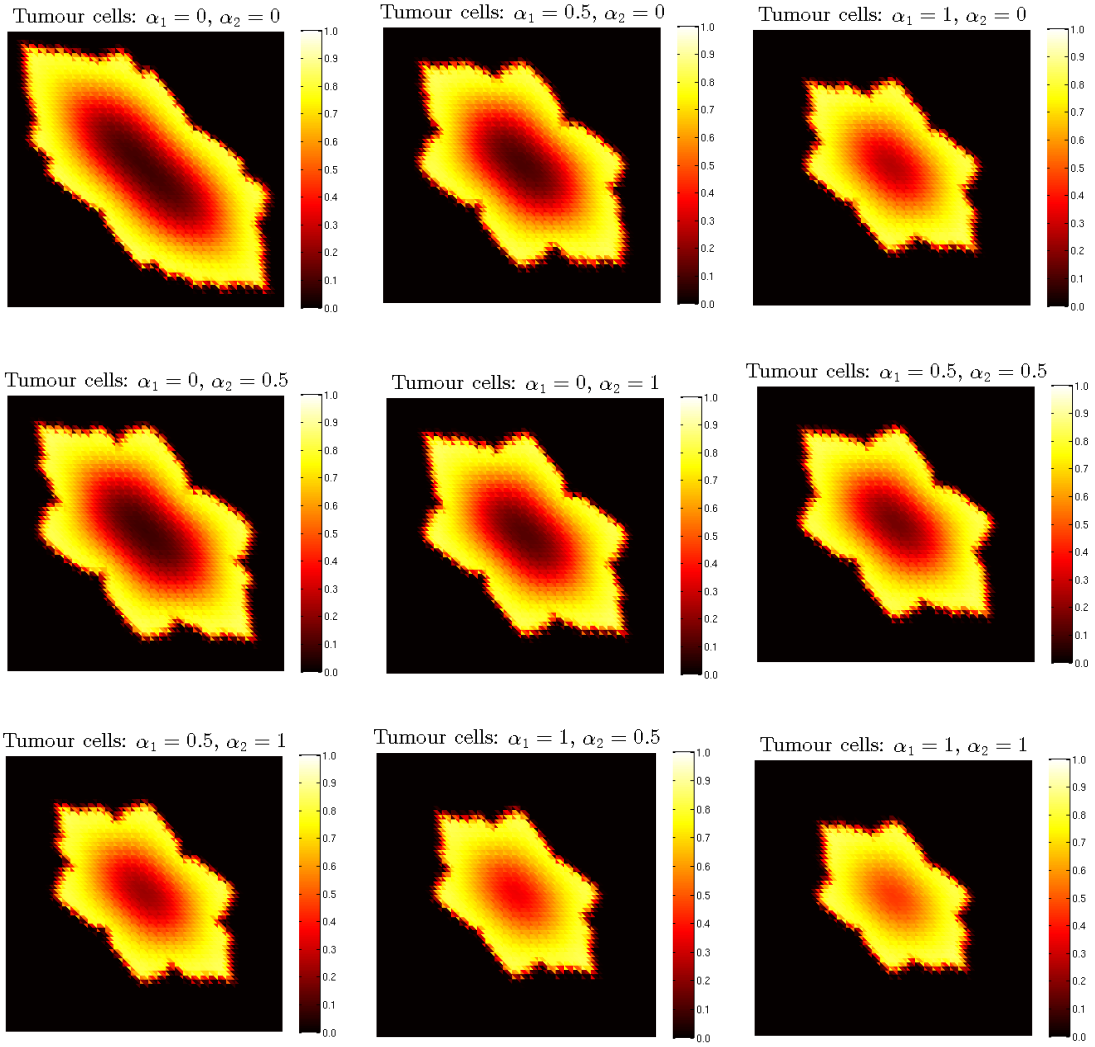


Figure 8.2: Evolution of the volume fraction of tumour cells on an unstructured grid: we vary α_1 and α_2 in equation (8.4). All plots show solutions at the same time, t=300. The number of unknowns in the momentum system equal to 76237.

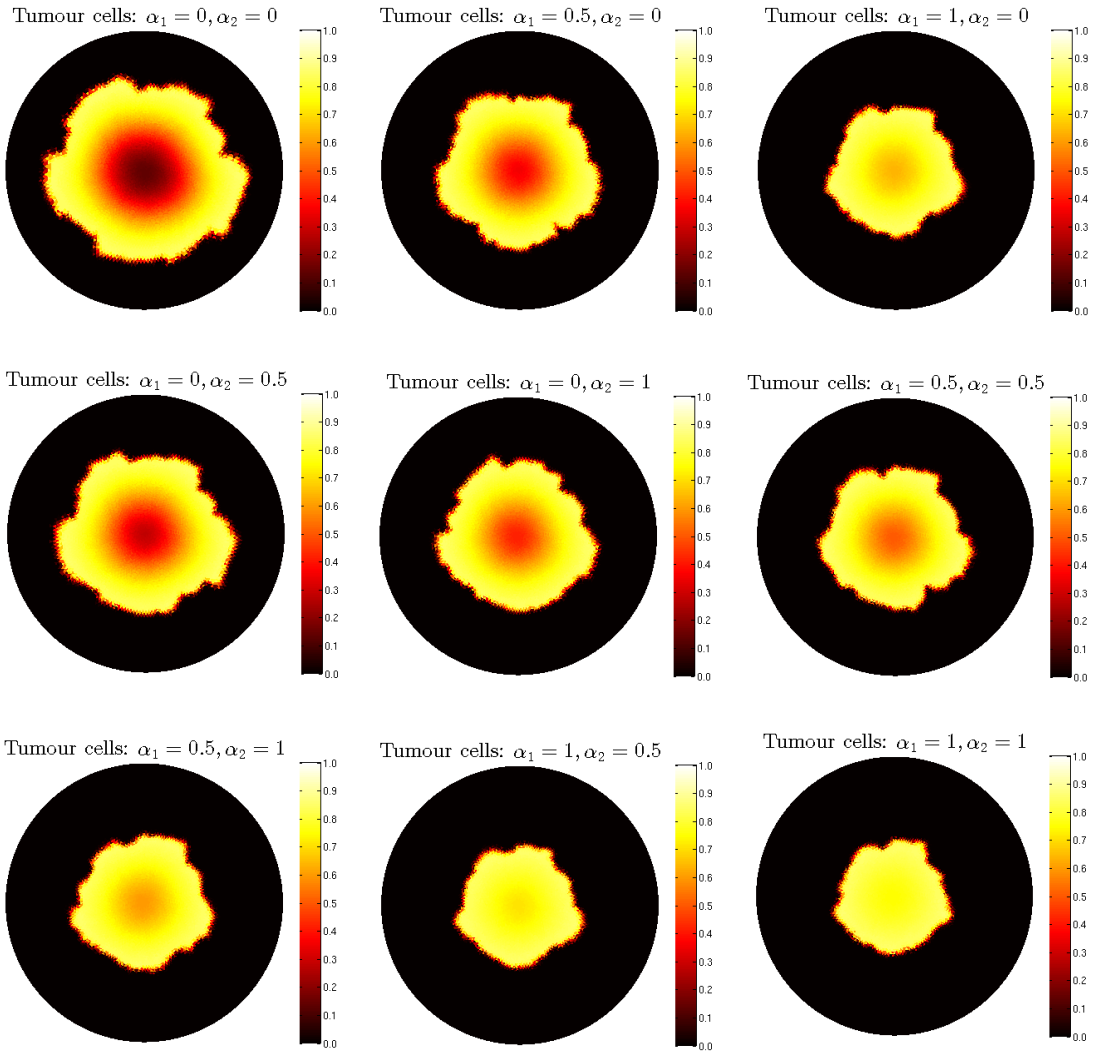
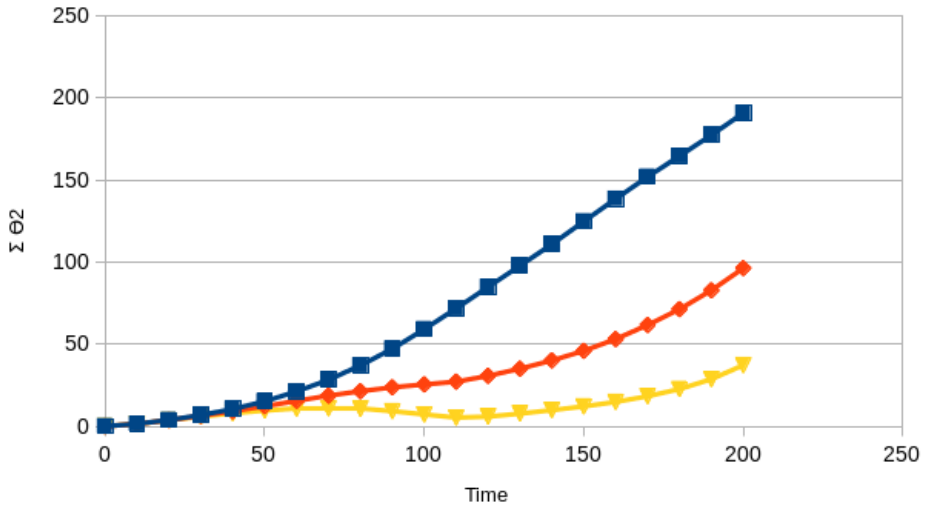


Figure 8.3: Comparison between the total mass of tumour (θ_2) for different solutions: without drug uptake (blue line), with moderate drug uptake ($\alpha_1 = \alpha_2 = 0.5$) (red line) and finally with high drug uptake ($\alpha_1 = \alpha_2 = 1$) (yellow line). This simulations are based upon $t_0 = 10$, $t_{max} = 105$, $t_1 = 200$ and dmax=1 in equation (8.3)



8.2 Additional Tumour Phase

The original idea behind the multiphase fluid model that is considered in this thesis was introduced for the first time by Breward et al. in [26]. In [26] the model includes just two phases, which are tumour cells and extracellular material, and avascular tumour growth simulated in just one-dimension. This model was extended by Breward et al. in [27] to include blood vessels as a third phase, in order to simulate vascular tumour growth. Then the mathematical models from [26, 27] were extended in [57] to include healthy cells as a fourth phase and two space dimensions (see Subsection 2.1.1 for greater detail about the related works). The mathematical model of [57] is further extended in this section to include a fifth phase.

8.2.1 Mathematical Model

The extra phase in our mathematical model is generated by splitting the tumour phase into two different phases: one with high susceptibility to the drug (HS) and other with low susceptibility to the drug (LS). So, the five phases in this model are: healthy cells, LS tumour cells, HS tumour cells, blood vessels and extracellular material.

As in Section 8.1, our model includes four governing systems of equations, which are: mass balance equations for the volume fraction of each phase (θ_i for $i = 1, ..., 5$), momentum balance equations for the flow of each phase (velocities u_i and v_i and pressure p_i for $i = 1, ..., 5$), the

reaction-diffusion equation for the nutrient concentration (c) and the reaction-diffusion equation for the drug concentration (d).

8.2.1.1 Mass Balance Equations

We start with mass balance equations. All phases considered have the same density, so the mass balance for the healthy cells (θ_1), LS tumour cells (θ_2), HS tumour cells (θ_5) and blood vessel (θ_3) volume fractions are given as follows :

$$\begin{aligned}
\frac{\partial \theta_1}{\partial t'} + \vec{\nabla}'.(\theta_1 \vec{u}_1') &= \underbrace{\theta_1\theta_4\left(\frac{c'}{c_p^* + c'}\right)}_{\text{cell birth}} - \underbrace{k_{2,1}^*\theta_1\left(\frac{c_{c_1}^* + c'}{c_{c_2}^* + c'}\right)}_{\text{cell death}} \\
\frac{\partial \theta_2}{\partial t'} + \vec{\nabla}'.(\theta_2 \vec{u}_2') &= \underbrace{k_{1,2}^*\theta_2\theta_4(1-\alpha_1^L d')\left(\frac{c'}{c_p^* + c'}\right)}_{\text{cell birth}} - \underbrace{k_{2,2}^*\theta_2(1+\alpha_2^L d')\left(\frac{c_{c_1}^* + c'}{c_{c_2}^* + c'}\right)}_{\text{cell death}} \\
\frac{\partial \theta_3}{\partial t'} + \vec{\nabla}'.(\theta_3 \vec{u}_3') &= \underbrace{-k_3^*\theta_3\mathcal{H}(\theta_1 p_1' + \theta_2 p_2' + \theta_5 p_5' - p_{crit}^*, \epsilon_3^*)}_{\text{occlusion}} + \underbrace{k_4^*(\theta_1+\theta_2+\theta_5)\theta_3\left(\frac{\theta_4}{\varepsilon+\theta_4}\right)\left(\frac{c'}{(c_a^*+c')^2}\right)}_{\text{angiogenesis}} \\
\frac{\partial \theta_5}{\partial t'} + \vec{\nabla}'.(\theta_5 \vec{u}_5') &= \underbrace{k_{1,2}^*\theta_5\theta_4(1-\alpha_1^H d')\left(\frac{c'}{c_p^* + c'}\right)}_{\text{cell birth}} - \underbrace{k_{2,2}^*\theta_5(1+\alpha_2^H d')\left(\frac{c_{c_1}^* + c'}{c_{c_2}^* + c'}\right)}_{\text{cell death}},
\end{aligned} \tag{8.8}$$

in which

$$k_{1,2}^* = \tfrac{k_{1,2}}{k_{1,1}}, \quad k_{2,1}^* = \tfrac{k_{2,1}}{k_{1,1}}, \quad k_3^* = \tfrac{k_3}{k_{1,1}}, \quad k_4^* = \tfrac{k_4}{c_v k_{1,1}},$$

$$c_p^* = \tfrac{c_p}{c_v}, \quad c_a^* = \tfrac{c_a}{c_v}, \quad c_{c_1}^* = \tfrac{c_{c_1}}{c_v}, \quad c_{c_2}^* = \tfrac{c_{c_2}}{c_v},$$

$$p_{crit}^* = \tfrac{p_{crit}}{\Lambda}, \quad \epsilon_3^* = \tfrac{\epsilon_3}{\Lambda},$$

and the smooth switch function:

$$\mathcal{H}(p,\epsilon) = 0.5(1 + tanh\frac{p}{\epsilon}), \quad \epsilon \ll 1.$$

As in Subsection 4.2.1, this model is written in dimensionless form and the primes denote the dimensionless variables. θ_1, θ_2, θ_3, θ_4 and θ_5 denote the phase volume fractions of the healthy cells, LS tumour cells, blood vessels, extracellular material and HS tumour cells respectively, c' is the nutrient concentration, p_i' are the pressures in each phase, c_p, c_{c_1}, c_{c_2} denote the nutrient concentration parameters, c_v is the nutrient concentration within the blood vessels, d' is the drug concentration, (α_1^L, α_2^L) and (α_1^H, α_2^H) are the parameters associated with lower and higher susceptibility to the drug, $k_{1,1}$,$k_{1,2}$,$k_{2,1}$,$k_{2,2}$, k_3 and k_4 are pre-defined rate constants, ε is the volume fraction of the extracellular material at half the maximal angiogenesis rate, while c_a is

the nutrient concentration at the maximal angiogenesis rate, and p^*_{crit} is the critical pressure for vessel occlusion.

Equations (8.8) are evolution equations for updating the volume fractions of the cells (θ_1, θ_2, θ_5) and blood vessels (θ_3). To determine the volume fraction for the extracellular phase (θ_4), we use the no-voids condition given by

$$\sum_{i=1}^{5} \theta_i = 1. \tag{8.9}$$

8.2.1.2 Momentum Balance Equations

Following Subsection 4.2.2, the following equations describe momentum balance for the dimensionless phase velocities $\vec{u}'_i$ and pressures p'_i, where $i = 1, ..., 5$:

$$\sum_{j \neq i} d^*_{ij} \theta_i \theta_j (\vec{u}'_j - \vec{u}'_i) - \theta_i \vec{\nabla}'. \, (\Lambda^* p'_i \mathbf{I}) + \vec{\nabla}'.[\theta_i[\mu^*_i(\vec{\nabla}'\vec{u}'_i + (\vec{\nabla}'\vec{u}'_i)^T) + \lambda^*_i(\vec{\nabla}'.\vec{u}'_i)\mathbf{I}]] = 0. \tag{8.10}$$

The incompressibility condition implies that

$$\Sigma_{i=1}^{5} \vec{\nabla}'. \, (\theta_i \vec{u}'_i) = 0,$$

where

$$p'_1 = p'_2 = p'_5 = p'_4 + \Sigma'(\theta), \qquad p'_3 = \frac{p^*_3}{\Lambda}.$$

Furthermore
$d^*_{ij} = \frac{d_{ij}}{d_{12}}$, $\Lambda^* = \frac{\Lambda}{d_{12} k_{1,1} L_0^2}$,
$\mu^*_i = \frac{\mu_i}{d_{12} L_0^2}$, $\lambda^*_i = \frac{\lambda_i}{d_{12} L_0^2}$,

and

$$\Sigma'(\theta) = \begin{cases} \frac{(\theta - \theta^*)}{(1-\theta)^2} & \text{if } \theta \geq \theta^* \\ 0 & \text{if } \theta < \theta^*, \end{cases} \tag{8.11}$$

in which d^*_{ij} is the drag coefficient, $d_{ij} = d_{ji}$ for $i, j = 1, 2, 3, 4, 5$, and $i \neq j$, μ_i denote the dynamic shear and λ_i are the bulk viscosities that are related through $\lambda_i = -\frac{2}{3}\mu_i$. The total volume fraction of a cell is considered as $\theta = \theta_1 + \theta_2 + \theta_5$, and θ^* is called the cells natural density. $\Sigma'(\theta)$ describes the pressure in the cell resulting from cell-cell interaction.

The momentum balance equations (8.10) is linear in $\vec{u_i}'$ and p'_4, and not time dependent. Moreover, p'_3 is assumed to be constant ($p'_3 = \frac{p^*_3}{\Lambda}$), where p^*_3 is the externally-imposed pressure and it is assumed constant. Hence, the equations (8.10) update the velocities $\vec{u_i}$ (five velocities in x-direction and five velocities in y-direction) and only the pressure p'_4.

8.2.1.3 Reaction Diffusion Equations

Following Subsection 4.2.3, the reaction-diffusion equation (quasi-steady-state) for the nutrient concentration c' can be written as

$$\begin{aligned} D_c^*\nabla'^2 c' &= \underbrace{\theta_3(1-c')}_{\text{replenishment}} \underbrace{-k_{6,1}^*\theta_1 c' - k_{6,2}^*\theta_2 c' - k_{6,2}^*\theta_5 c'}_{\text{baseline consumption}} \\ &- \underbrace{k_{7,1}^*\theta_1\theta_4\left(\frac{c'}{c_p^*+c'}\right) - k_{7,2}^*(\theta_2+\theta_5)\theta_4\left(\frac{c'}{c_p^*+c'}\right)}_{\text{consumption due to cell birth}}, \end{aligned} \tag{8.12}$$

in which

$$\begin{aligned} &D_c^* = \tfrac{D_c}{k_5 L_0^2}, \quad k_{6,1}^* = \tfrac{k_{6,1}}{k_5}, \quad k_{6,2}^* = \tfrac{k_{6,2}}{k_5}, \\ &k_{7,1}^* = \tfrac{k_{7,1}}{c_v k_5}, \quad k_{7,2}^* = \tfrac{k_{7,2}}{c_v k_5}, \end{aligned} \tag{8.13}$$

where D_c^* is the diffusion coefficient, (k_5, $k_{6,1}$, $k_{6,2}$, $k_{7,1}$, $k_{7,2}$) are pre-defined rate constants, with assumed $\frac{k_{7,1}}{k_{7,2}} = \frac{k_{1,1}}{k_{1,2}}$. This equation is a nonlinear equation with a single variable c'.

As in Section 8.1, in this model we add another nonlinear equation which is the reaction-diffusion equation for the drug concentration d'. This equation can be written as

$$\begin{aligned} D_d^*\nabla'^2 d' &= \theta_3(dv(t)-d') - kd_{6,1}^*\theta_1 d' - kd_{6,2}^*(\theta_2+\theta_5)d' \\ &- kd_{7,1}^*\theta_1\theta_4\left(\frac{d'}{d_p^*+d'}\right) - kd_{7,2}^*(\theta_2+\theta_5)\theta_4\left(\frac{d'}{d_p^*+d'}\right) \end{aligned} \tag{8.14}$$

in which

$$\begin{aligned} &D_d^* = \tfrac{D_d}{kd_5 L_0^2}, \quad kd_{6,1}^* = \tfrac{kd_{6,1}}{kd_5}, \quad kd_{6,2}^* = \tfrac{kd_{6,2}}{kd_5}, \\ &kd_{7,1}^* = \tfrac{kd_{7,1}}{d_v kd_5}, \quad kd_{7,2}^* = \tfrac{kd_{7,2}}{d_v kd_5}, \end{aligned} \tag{8.15}$$

in which D_d^* is the drug diffusion coefficient, (kd_5, $kd_{6,1}$,$kd_{6,2}$,$kd_{7,1}$,$kd_{7,2}$) are pre-defined rate constants. As before d_p^* and d_v denote the drug concentration parameter and the drug concentration within the blood vessels respectively. This equation is a nonlinear equation with a single variable d'. The drug supply to the tumour cells depends on pharmacokinetics (PK) and dv(t) defines the pharmacokinetic (PK) equation, which is given by (8.3).

8.2.2 Extension of the Numerical Methods

In this subsection, the numerical solution schemes for the new model, presented in the previous subsection, are described. The numerical approach presented in Section 4.2 is extended to include the new phase.

8.2.2.1 Mass Balance Equations

The mass balance equations (8.8) are hyperbolic and time dependent PDEs. An explicit solver in time, with a cell centred finite volume scheme and forward Euler time stepping is used to approximate the equation (8.8). The integral form for the mass balance equation can be written as follows:

$$\int_{\Delta} \frac{\partial \theta_i}{\partial t} d\vec{x} + \int_{\Delta} \vec{\nabla}.(\theta_i \vec{u}_i) d\vec{x} = \int_{\Delta} q_i d\vec{x} \qquad i = 1, 2, 3, 5. \tag{8.16}$$

The discrete system of equations is designed to update the unknowns θ_i, which are the cell-average values. The equation (8.16) is used to update θ_1, θ_2, θ_3 and θ_5 in time, while the no-voids condition (8.9) is used to update θ_4.

By using the Gauss divergence theorem for the flux integrals in the previous equation we obtain

$$\int_{\Delta} \frac{\partial \theta_i}{\partial t}\ d\vec{x} + \oint_{\partial\Delta} (\theta_i \vec{u}_i).\vec{n}\ ds = \int_{\Delta} q_i d\vec{x} \qquad i = 1, 2, 3, 5 \tag{8.17}$$

where Δ is the control volume, q_i are the source/sink terms, $\partial\Delta$ is the boundary of the control volume, and $\vec{n}$ is the outward-pointing unit normal to the boundary. A cell-centred MUSCL approach is used, which is a conservative, upwind, finite volume scheme. The fluxes $\theta_i \vec{u}_i$ are approximated by using a standard upwind scheme [71]. So, the discrete equation for each triangle can be written as

$$\overline{\theta}_i^{n+1} = \overline{\theta}_i^{n} - \frac{\Delta t}{\mid \Delta \mid} \sum_{k=1}^{3} (\theta_i^n \vec{u_i^n})_k^*.\vec{n_k} + \Delta t (q_i^n)^*, i = 1, ..., 5, \tag{8.18}$$

where n is the time level, Δt is the time step size, $\mid \Delta \mid$ is the control volume area, and $\vec{n_k}$ is the outward-pointing unit normal to the edge of the cell opposite the vertex k. The asterisk (*) is used to distinguish the quantities, which have been approximated depending on the variables $\bar{\theta}_i$, which is a cell average.

8.2.2.2 Momentum Balance Equations

The momentum balance equations (8.10) are steady PDEs, and linear in $\vec{u_i}$ and p_4. The weak form is obtained by applying a Galerkin finite element scheme with Taylor-Hood elements (see Section 3.1.1) as

$$\int_\Omega \omega^q \vec{\nabla}.(\theta_i \sigma_i) d\vec{x} + \int_\Omega \omega^q \vec{F}_i d\vec{x} = 0 \quad i = 1, ..., 5,$$

$$\int_\Omega \omega^l \sum_{i=1}^{5} \vec{\nabla}.\ (\theta_i \vec{u}_i)\ d\vec{x} = 0 \tag{8.19}$$

where ω^l and ω^q are the standard linear and quadratic Lagrange test functions respectively, σ_i are the stresses in each individual phase, which can be defined as

$$\sigma_i = -p_i \mathbf{I} + \mu_i(\vec{\nabla}\vec{u}_i + (\vec{\nabla}\vec{u}_i)^T) + \lambda_i(\vec{\nabla}.\vec{u}_i)\mathbf{I} \quad i = 1, ..., 5,$$

and $\vec{F}_i$ are the momentum sources, which can be defined as

$$\vec{F}_i = p_i \mathbf{I} \vec{\nabla} \theta_i + \sum_{j=1, j\neq i}^{5} d_{ij} \theta_i \theta_j (\vec{u}_j - \vec{u}_i) \quad i = 1, ..., 5.$$

Using integration by parts for the first equation from (8.19) leads to

$$\oint_{\partial\Omega} \omega^q \theta_i \sigma_i.\vec{n}\ ds - \int_\Omega \vec{\nabla}\omega^q.\theta_i \sigma_i d\vec{x} + \int_\Omega \omega^q \vec{F}_i d\vec{x} = 0 \quad i = 1, ..., 5. \tag{8.20}$$

The velocities $\vec{u_i}$ and pressures p are approximated by using piecewise polynomials, which are written using the trial function as

$$\vec{u_i} \approx \sum_{k=1}^{N_q} (\vec{u_i})_k \omega_k^q, \qquad p \approx \sum_{k=1}^{N_l} p_k \omega_k^l, \tag{8.21}$$

in which N_q and N_l are, respectively, the numbers of degrees of freedom related to the quadratic and linear Lagrange elements.

The boundary conditions that can be used for the momentum balance equations, are Dirichlet condition on $\vec{u_i}$, or $\sigma_i.\vec{n}$ specified. Here, the boundary conditions imposed are zero normal stress $\sigma_i.\vec{n} = 0$ for four phases: healthy cells, LS tumour cells, HS tumour cells and blood vessels; and zero velocity for the extracellular phase $\vec{u}_4 = 0$.

These choices of boundary conditions for the normal stress lead to the boundary term in (8.20) becoming zero and so (8.19) may be expressed as:

$$\begin{aligned} -\int_\Omega \vec{\nabla}\omega^q.\theta_i \sigma_i d\vec{x} + \int_\Omega \omega^q \vec{F}_i d\vec{x} &= 0, \quad i = 1, ..., 5, \\ \int_\Omega \omega^l \Sigma_{i=1}^5 \vec{\nabla}.\ (\theta_i u_i)\ d\vec{x} &= 0. \end{aligned} \tag{8.22}$$

By applying the integration by parts in the second equation from (8.22), and using (8.21), we

obtain:

$$\sum_{i=1}^{5}\sum_{k=1}^{N_q}\int_\Omega[\omega_m^l\theta_i\frac{\partial\omega_k^q}{\partial x}+\omega_m^l\omega_k^q\frac{\partial\theta_i}{\partial x}](u_i)_k d\vec{x}+$$
$$\sum_{i=1}^{5}\sum_{k=1}^{N_q}\int_\Omega[\omega_m^l\theta_i\frac{\partial\omega_k^q}{\partial y}+\omega_m^l\omega_k^q\frac{\partial\theta_i}{\partial y}](v_i)_k d\vec{x}=0 \tag{8.23}$$

where $m=1,...,N_l$. From the definition of σ_i, the first equation in (8.22) can be written as:

$$\begin{aligned}\sum_{k=1}^{N_q}\int_\Omega\Big\{\big[\mu_i\theta_i(2\tfrac{\partial\omega_k^q}{\partial x}\tfrac{\partial\omega_m^q}{\partial x} &+ \tfrac{\partial\omega_k^q}{\partial y}\tfrac{\partial\omega_m^q}{\partial y})+\lambda_i\theta_i\tfrac{\partial\omega_k^q}{\partial x}\tfrac{\partial\omega_m^q}{\partial x}+\omega_k^q\omega_m^q d_{ij}\theta_i\theta_j\big](u_i)_k\\ &- \big[\omega_k^q\omega_m^q d_{ij}\theta_i\theta_j\big](u_j)_k\\ &+ \big[\mu_i\theta_i\tfrac{\partial\omega_k^q}{\partial y}\tfrac{\partial\omega_m^q}{\partial x}+\lambda_i\theta_i\tfrac{\partial\omega_k^q}{\partial x}\tfrac{\partial\omega_m^q}{\partial y}\big](v_i)_k\Big\}d\vec{x}\\ &- \sum_{k=1}^{N_l}\int_\Omega\Big\{\big[\omega_k^l\theta_i\tfrac{\partial\omega_m^q}{\partial x}-\omega_k^l\omega_m^q\tfrac{\partial\theta_i}{\partial x}\big](p_i)_k\Big\}d\vec{x}=\int_\Omega\omega^q\vec{F}_{ix}d\vec{x},\end{aligned} \tag{8.24}$$

$$\begin{aligned}\sum_{k=1}^{N_q}\int_\Omega\Big\{\big[\mu_i\theta_i(\tfrac{\partial\omega_k^q}{\partial x}\tfrac{\partial\omega_m^q}{\partial x} &+ 2\tfrac{\partial\omega_k^q}{\partial y}\tfrac{\partial\omega_m^q}{\partial y})+\lambda_i\theta_i\tfrac{\partial\omega_k^q}{\partial y}\tfrac{\partial\omega_m^q}{\partial y}+\omega_k^q d_{ij}\theta_i\theta_j\big](v_i)_k\\ &- \big[\omega_k^q\omega_m^q d_{ij}\theta_i\theta_j\big](v_j)_k\\ &+ \big[\mu_i\theta_i\tfrac{\partial\omega_k^q}{\partial x}\tfrac{\partial\omega_m^q}{\partial y}+\lambda_i\theta_i\tfrac{\partial\omega_k^q}{\partial y}\tfrac{\partial\omega_m^q}{\partial x}\big](u_i)_k\Big\}d\vec{x}\\ &- \sum_{k=1}^{N_l}\int_\Omega\Big\{\big[\omega_k^l\theta_i\tfrac{\partial\omega_m^q}{\partial y}-\omega_k^l\omega_m^q\tfrac{\partial\theta_i}{\partial y}\big](p_i)_k\Big\}d\vec{x}=\int_\Omega\omega^q\vec{F}_{iy}d\vec{x},\end{aligned} \tag{8.25}$$

where $i\neq j$, $i,j=1,...,5$, and $m=1,...,N_q$.

8.2.2.3 Reaction Diffusion Equations

The reaction-diffusion equations (8.12) and (8.14) for the nutrient and drug concentrations, respectively, are quasi/steady nonlinear PDEs. These equations are approximated using a standard Galerkin finite element scheme with linear elements to get:

$$\int_\Omega\omega^{l_f}D_c\vec{\nabla}^2c\ d\vec{x}-\int_\Omega\omega^{l_f}q_c d\vec{x}=0, \tag{8.26}$$

and

$$\int_\Omega\omega^{l_f}D_d\vec{\nabla}^2d\ d\vec{x}-\int_\Omega\omega^{l_f}q_d d\vec{x}=0, \tag{8.27}$$

in which D_c and D_d denote to the diffusion coefficient for the nutrient and drug concentrations, respectively, which are assumed to be constant. Also, ω^{l_f} is a linear test function defined on the finer mesh, which is used to update the volume fractions of the phases, q_c and q_d are source terms, which are nonlinear functions of the the nutrient and drug concentrations, respectively (based on the right-hand sides of equations (8.12) and (8.14)).

Using integration by parts leads to

$$\int_{\partial\Omega}\omega^{l_f}D_c\vec{\nabla}c.\vec{n}ds-\int_\Omega\vec{\nabla}\omega^{l_f}.D_c\vec{\nabla}c d\vec{x}-\int_\Omega\omega^{l_f}q_c d\vec{x}=0, \tag{8.28}$$

$$\int_{\partial\Omega} \omega^{l_f} D_d \vec{\nabla} c.\vec{n} ds - \int_{\Omega} \vec{\nabla}\omega^{l_f}.D_d\vec{\nabla} d d\vec{x} - \int_{\Omega} \omega^{l_f} q_d d\vec{x} = 0. \tag{8.29}$$

The nutrient concentration c and the drug concentration d can be written using the trial functions as:

$$c \approx \sum_{k=1}^{N_{l_f}} c_k \omega_k^{l_f}. \tag{8.30}$$

$$d \approx \sum_{k=1}^{N_{l_f}} d_k \omega_k^{l_f}. \tag{8.31}$$

In the following subsection, the initial and boundary conditions for the previous governing equations are introduced

8.2.3 Initial and Boundary Conditions

In this subsection we present the initial and boundary conditions that are used in our experiments. The initial conditions for simulation of a single tumour seeded in the centre of healthy tissue can be given as follows:

–

$$\theta_2(x,y,t=0) = \theta_5(x,y,t=0) = \begin{cases} 0.025 \quad cos^2(\frac{\pi r}{2}) \quad for \quad r \lneq 1 \\ 0 \qquad\qquad\qquad otherwise \end{cases} \tag{8.32}$$

in which $r = \sqrt{x^2+y^2}$.

– $\theta_1(x,y,0) = 0.6 - \theta_2(x,y,0) - \theta_5(x,y,0)$.

– $\theta_3 = 0.0174978$.

– $\theta_4 = 0.3825022$.

– Each phase has zero velocity.

– $p_3 = 0$ and $p_4 = 0$, then $p_1 = p_2 = p_5 = 0$.

– The nutrient concentration $c' = 0.2532031$ everywhere.

– The drug concentration $d' = 0$.

The boundary conditions are chosen as

– the boundary conditions for the system of equations (8.8) are imposed on the inflow section of the boundary Γ_i^{inflow} on which $\vec{u}_i'.\vec{n} < 0$:

$$\theta_i = \theta_i^{\infty}, \quad i = 1, ..., 5.$$

– the boundary conditions for the system of equations (8.10) are imposed on the whole of the boundary Γ

$$\sigma_i'.\vec{n} = 0, \;\; i = 1, 2, 3, 5 \quad and \qquad \vec{u}_4' = 0,$$

where σ_i' are the stresses in each individual phase.

– the boundary conditions for the system of equations (8.12) and (8.14) are imposed on the whole of the boundary Γ

$$\vec{\nabla}' c'.\vec{n} = 0.$$

and

$$\vec{\nabla}' d'.\vec{n} = 0.$$

where $\vec{n}$ is the unit outward-pointing normal to Γ.

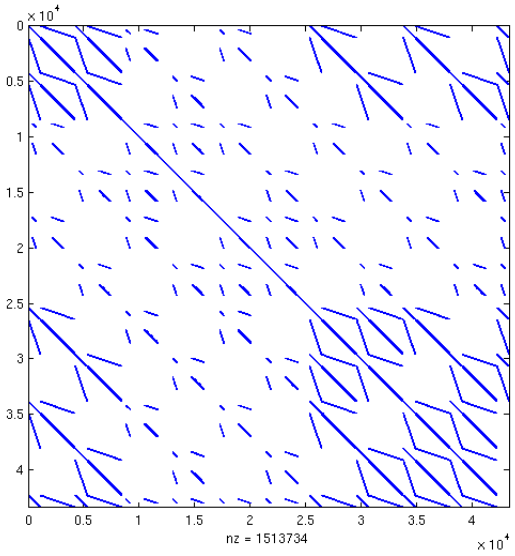
8.2.4 Modifications to the Linear Momentum System

The block-matrix system for the discrete momentum equations in our new model may be expressed in the form:

$$\begin{pmatrix}
k_{xx11} & k_{xy11} & k_{xx12} & 0 & k_{xx15} & 0 & k_{xx14} & 0 & k_{xx13} & 0 & C_{x1} \\
k_{yx11} & k_{yy11} & 0 & k_{yy12} & 0 & k_{yy15} & 0 & k_{yy14} & 0 & k_{yy13} & C_{y1} \\
k_{xx21} & 0 & k_{xx22} & k_{xy22} & k_{xx25} & 0 & k_{xx24} & 0 & k_{xx23} & 0 & C_{x2} \\
0 & k_{yy21} & k_{yx22} & k_{yy22} & 0 & k_{yy51} & 0 & k_{yy24} & 0 & k_{yy23} & C_{y2} \\
k_{xx51} & 0 & k_{xx52} & 0 & k_{xx55} & k_{xy55} & k_{xx54} & 0 & k_{xx53} & 0 & C_{x5} \\
0 & k_{yy51} & 0 & k_{yy52} & k_{yx55} & k_{yy55} & 0 & k_{yy54} & 0 & k_{yy53} & C_{y5} \\
k_{xx41} & 0 & k_{xx42} & 0 & k_{xx45} & 0 & k_{xx44} & k_{xy44} & k_{xx43} & 0 & C_{x4} \\
0 & k_{yy41} & 0 & k_{yy42} & 0 & k_{yy45} & k_{yx44} & k_{yy44} & 0 & k_{yy43} & C_{y4} \\
k_{xx31} & 0 & k_{xx32} & 0 & k_{xx35} & 0 & k_{xx34} & 0 & k_{xx33} & k_{xy33} & 0 \\
0 & k_{yy31} & 0 & k_{yy32} & 0 & k_{yy35} & 0 & k_{yy34} & k_{yx33} & k_{yy33} & 0 \\
B_{x1}^T & B_{y1}^T & B_{x2}^T & B_{y2}^T & B_{x5}^T & B_{y5}^T & B_{x4}^T & B_{y4}^T & B_{x3}^T & B_{y3}^T & 0
\end{pmatrix}
\begin{pmatrix}
u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x5} \\ u_{y5} \\ u_{x4} \\ u_{y4} \\ u_{x3} \\ u_{y3} \\ p_4
\end{pmatrix}
=
\begin{pmatrix}
fx_1 \\ fy_1 \\ fx_2 \\ fy_2 \\ fx_5 \\ fy_5 \\ fx_4 \\ fy_4 \\ fx_3 \\ fy_3 \\ 0
\end{pmatrix}. \tag{8.33}$$

In this model, the coefficient matrix A has an 11×11 block structure, with 10 sets of velocity variables and 1 set of pressure variables. Each two rows express the x and y directions of momentum of each phase, starting from the top healthy phase, LS tumour phase, HS tumour cells, extracellular phase and blood vessels phase. The final block row is obtained from the continuity equation. The sparsity pattern of this coefficient matrix for the regular grids is shown in Figure 8.4. Note that the momentum balance equations (8.10) are approximated by Taylor-Hood FEM, which uses linear and quadratic basis functions for pressure and velocities variables (as shown in Figure 3.2). So, the number of unknowns in this linear system is $N = 10n^q + n^l$ in which n^q is the nodes on the vertices and the edge of the elements and n^l is the node on the vertices of the elements only.

Figure 8.4: The sparsity pattern A for 33^2 grid with unknown N=34889.



The discrete system is nonsymmetric and may be solved using the GMRES method with a suitable preconditioner. The next section shows how we have generalized our preconditioning, P7, to use it for solving this new model.

8.2.5 Preconditioning

Our new preconditioning P7 can be extended to include the extra phase variables as follows:

$$P8 = \begin{pmatrix} k_{xx11_{AMG}} & k_{xy11} & k_{xx12} & 0 & k_{xx15} & 0 & k_{xx14} & 0 & k_{xx13} & 0 & C_{x1} \\ 0 & k_{yy11_{AMG}} & 0 & k_{yy12} & 0 & k_{yy15} & 0 & k_{yy14} & 0 & k_{yy13} & C_{y1} \\ 0 & 0 & k_{xx22_{AMG}} & k_{xy22} & k_{xx25} & 0 & k_{xx24} & 0 & k_{xx23} & 0 & C_{x2} \\ 0 & 0 & 0 & k_{yy22_{AMG}} & 0 & k_{yy25} & 0 & k_{yy24} & 0 & k_{yy23} & C_{y2} \\ 0 & 0 & 0 & 0 & k_{xx55_{AMG}} & k_{xy55} & k_{xx54} & 0 & k_{xx53} & 0 & C_{x5} \\ 0 & 0 & 0 & 0 & 0 & k_{yy55_{AMG}} & 0 & k_{yy54} & 0 & k_{yy53} & C_{y5} \\ 0 & 0 & 0 & 0 & 0 & 0 & k_{xx44_{AMG}} & k_{xy44} & k_{xx43} & 0 & C_{x4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{yy44_{AMG}} & 0 & k_{yy43} & C_{y4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{xx33_{AMG}} & k_{xy33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{yy33_{AMG}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & mp_{CG} \end{pmatrix}.$$

The preconditioner P8 is a block-upper-triangular matrix. As we did in the Chapter 6, the diagonal blocks are solved only approximately, using just one V-cycle of the AMG method, and the diagonal of pressure mass matrix (mp) is solved for the final block. The steps in Algorithm 9 may be followed to solve the system $P8z = r$, in which z and r vectors are defined in block

matrix notation as:

$$z = \begin{pmatrix} zu_1 \\ zv_1 \\ zu_2 \\ zv_2 \\ zu_5 \\ zv_5 \\ zu_4 \\ zv_4 \\ zu_3 \\ zv_3 \\ z_p \end{pmatrix}, \qquad r = \begin{pmatrix} ru_1 \\ rv_1 \\ ru_2 \\ rv_2 \\ ru_5 \\ rv_5 \\ ru_4 \\ rv_4 \\ ru_3 \\ rv_3 \\ r_p \end{pmatrix}.$$

Algorithm 9 Solving P8z=r in step 3 from GMRES Algorithm 3

1: Solve $diag(mp)z_p = r_p$ using CG.
2: Solve $k_{yy33}zv_3 = rv_3$ using AMG.
3: Update $ru_3 = ru_3 - k_{xy33}zv_3$
4: Solve $k_{xx33}zu_3 = ru_3$ using AMG.
5: Update $rv_4 = rv_4 - k_{yy43}zv_3 - C_{y4}z_p$
6: Solve $k_{yy44}zv_4 = rv_4$ using AMG.
7: Update $ru_4 = ru_4 - k_{xy44}zv_4 - k_{xx43}zu_3 - C_{x4}z_p$
8: Solve $k_{xx44}zu_4 = ru_4$ using AMG
9: Update $rv_5 = rv_5 - k_{yy54}zv_4 - k_{yy53}zv_3 - C_{y5}z_p$
10: Solve $k_{yy55}zv_5 = rv_5$ using AMG.
11: Update $ru_5 = ru_5 - k_{xy55}zv_5 - k_{xx54}zu_4 - k_{xx53}zu_3 - C_{x5}z_p$
12: Solve $k_{xx55}zu_5 = ru_5$ using AMG
13: Update $rv_2 = rv_2 - k_{yy25}zv_5 - k_{yy24}zv_4 - k_{yy23}zv_3 - C_{y2}z_p$
14: Solve $k_{yy22}zv_2 = rv_2$ using AMG.
15: Update $ru_2 = ru_2 - k_{xy22}zv_2 - k_{xx25}zu_5 - k_{xx24}zu_4 - k_{xx23}zu_3 - C_{x2}z_p$
16: Solve $k_{xx22}zu_2 = ru_2$ using AMG
17: Update $rv_1 = rv_1 - k_{yy12}zv_2 - k_{yy15}zv_5 - k_{yy14}zv_4 - k_{yy13}zv_3 - C_{y1}z_p$
18: Solve $k_{yy11}zv_1 = rv_1$ using AMG.
19: Update $ru_1 = ru_1 - K_{xy11}zv_1 - k_{xx12}zu_2 - k_{xx15}zu_5 - k_{xx14}zu_4 - k_{xx13}zu_3 - C_{x1}z_p$
20: Solve $k_{xx11}zu_1 = ru_1$ using AMG.

8.2.5.1 Eigenvalues

Before we present the numerical results for our new model, it is worth presenting the eigenvalues resulting from the application of our new preconditioner, P8. Table 8.2 illustrates the maximum and the minimum eigenvalues of the coefficient matrix A and the preconditioned matrix AP^{-1}. From this table we can observe that the eigenvalues of the matrix A are spread in a wide range and this range increases as the grid size increases. Conversely, the eigenvalues of the preconditioned system are bounded in a small range far from zero and infinity, and independent

of the grid size.

Table 8.2: Minimum and maximum eigenvalues of the coefficient matrix A and preconditioned matrix AP^{-1}

Grid	MIN $\lambda(A)$	MAX $\lambda(A)$	MIN $\lambda(AP^{-1})$	MAX $\lambda(AP^{-1})$
9^2	0.0051	124.2008	0.0023	1.0238 + 0.0168i
17^2	4.5407e-05	131.0706	0.0023	1.0166 + 0.0181i
33^2	3.5960e-07	139.3251	0.0022	1.0159 + 0.0002i

Figure 8.5 and Figure 8.6 present the N eigenvalues of the matrix A and the preconditioned matrix, AP^{-1}, respectively on the grid size 9^2. From both figures it can be observed that the eigenvalues of A are not all real, however the maximum and the minimum values are real. Moreover, using the preconditioning technique improves the results and makes the majority of the eigenvalues cluster in a small range around 1 in the complex plane, with just a small number of isolated eigenvalues near to the origin.

Figure 8.5: The eigenvalues of the coefficient matrix A on the grid size 9^2.

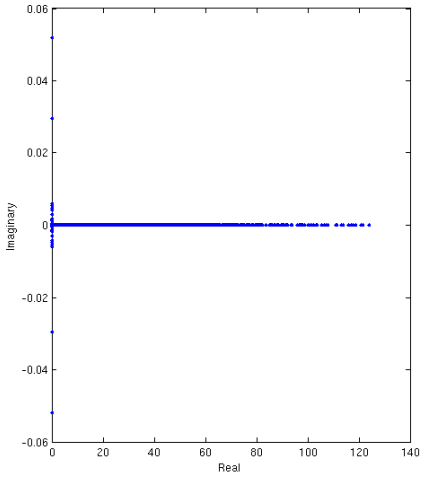
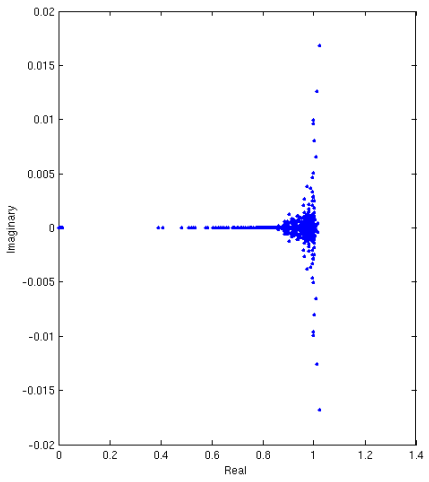


Figure 8.6: The eigenvalues of the preconditioned matrix AP^{-1} on the grid size 9^2.



8.2.6 Numerical Results

In this subsection we introduce numerical results for the solution of the new model that is presented in Section 8.2.1. In these numerical experiments the time step size $\Delta t = 0.25$ and the number of time steps 1000 are used to assess the performance of our proposed preconditioner.

As mentioned before, AMG preconditioned GMRES method is used for finding the solution of the reaction-diffusion equations, and our new preconditioner is applied with the GMRES method to find the solution of the discretised momentum balance equations. We make use of the software implementation that is available in Harwell Subroutine Library (HSL) [1]. This includes: HSL-MI20 for the AMG method, HSL-MI21 for the CG method and HSL-MI24 for the GMRES method.

8.2.6.1 Regular Grids

We start with the application of the new model on a square domain using regular triangular grids such as in Figure 5.3. We first undertake the experiments with chosen values of $\alpha_{L1} = \alpha_{L2} = 0.1$ and $\alpha_{H1} = \alpha_{H2} = 1$ in equation (8.8). Then later we test other values to further investigates.

The solutions of the nonlinear reaction-diffusion system (8.12) for the nutrient concentration are reported in Table 8.3. From this table it can be seen that the outer iterations (Newton iterations) and the inner iterations (GMRES iterations) are still almost fixed and independent of the grid size. Moreover the time clearly scales linearly as we would expect. Table 8.4 presents the corresponding results from obtained solving the discrete nonlinear reaction-diffusion equation system (8.14) for the drug concentration. Also in this table the Newton iterations and the GMRES iterations are nearly fixed in the all cases. The computational time that is required behaves as $\mathcal{O}(N)$. Overall, in this model the solvers for the nonlinear systems still have $\mathcal{O}(N)$ complexity.

Table 8.3: The average running times (in seconds), over 1000 time steps, required for solving the linear discrete system from equation (8.12) using AMG preconditioned GMRES on regular girds: N is the number of unknowns, NI is Newton iterations, GI is GMRES iterations at each NI and the relative tolerance for GMRES is $tol = 1e-3$. The tolerance for Newton is $1e-12$.

Grid	N	average NI	average GI per NI	average time per step (sec.)
33^2	4225	3.9790	2.0572	0.046609948
65^2	16641	3.9281	2.2215	0.187927939
129^2	66049	3.9790	2.2553	0.739807022
257^2	263169	3.6084	2.4092	2.7783027529

Table 8.4: The average running times (in seconds), over 1000 time steps, required for solving the linear discrete system from equation (8.14) using AMG preconditioned GMRES on regular girds: N is the number of unknowns, NI is Newton iterations, GI is GMRES iterations at each NI and the relative tolerance for GMRES is $tol = 1e-3$. The tolerance for Newton is $1e-12$.

Grid	N	average NI	average GI per NI	average time per step (sec.)
33^2	4225	3.2777	2.3721	0.073539967
65^2	16641	3.2687	2.3154	0.295210102
129^2	66049	3.2687	2.3839	1.132743667
257^2	263169	2.5205	2.7983	3.6129077077

For the solution of the discretized linear momentum equation system (8.10), combined with the incompressibility condition, the numerical results are shown in Table 8.5. From this table it can be observed that using our proposed approach, based on preconditioned GMRES with P8 preconditioning, for solving the linear algebra system (8.33) achieves almost optimal solution time behaviour. Indeed, the number of GMRES iterations are almost fixed for all the problem sizes.

Table 8.5: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system (8.33) using P8 preconditioned GMRES on regular grids: GI is GMRES iterations (the relative tolerance for GMRES is $tol = 1e-3$), N is the number of unknowns.

Grid	N	average GI	average Time per step (sec.)
33^2	43339	29.8012	1.043650373
65^2	170635	29.6104	4.196122836
129^2	677131	29.1489	15.985175519
257^2	2697739	28.8761	63.189456249

The computational time and memory costs for the whole new model on a sequence of regular domains is shown in Table 8.6. It is clear that the CPU time and the percentage of memory used both increase by factor of nearly 4 as the number of unknowns N quadruples. Consequently, the results scale linearly as $\mathcal{O}(N)$.

Table 8.6: The CPU time and percentage of memory cost for the whole model on regular grids using preconditioned GMRES: the linear system (8.33) is solved using p8, and the nonlinear systems solved using AMG preconditioning, with GMRES relative tolerance $1e-3$.

Grid	MEM (%)	Time
33^2	1	19m26s
65^2	4.2	78m21s
129^2	13.2	297m12s
257^2	41.1	1154m57s

For further investigation, and to ensure that our results are not parameter dependent, we

chose $\alpha_{L1} = \alpha_{L2} = 0.2$ and $\alpha_{H1} = \alpha_{H2} = 0.5$ in equation (8.8). From the following tables we again see that in this experiment, the number of Newton iterations and the GMRES iterations that are required for solving the nonlinear systems are still almost fixed (see Table 8.7 and Table 8.8). Also, the computational time achieves the optimal linear complexity. By comparing these results in Table 8.7 and Table 8.8 with the previous results, that are presented respectively in Table 8.3 and Table 8.4, we can observed that the computational time and the number of Newton and GMRES iterations required in both tests are almost identical.

Table 8.7: The average running times (in seconds), over 1000 time steps, required for solving the linear discrete system from equation (8.12) using AMG preconditioned GMRES on regular girds: N is the number of unknowns, NI is Newton iterations, GI is GMRES iterations and the relative tolerance for GMRES is $tol = 1e-3$. Here $\alpha_{L1} = \alpha_{L2} = 0.2$ and $\alpha_{H1} = \alpha_{H2} = 0.5$ in equation (8.8)

Grid	N	average NI	average GI per NI	average time per step (sec.)
33^2	4225	3.9810	2.0570	0.04512942483
65^2	16641	3.9840	2.1991	0.1790750852
129^2	66049	3.9810	2.2529	0.738551417
257^2	263169	3.6603	2.4116	2.808772492

Table 8.8: The average running times (in seconds), over 1000 time steps, required for solving the linear discrete system from equation (8.14) using AMG preconditioned GMRES on regular girds: N is the number of unknowns, NI is Newton iterations, GI is GMRES iterations and the relative tolerance for GMRES is $tol = 1e-3$. Here $\alpha_{L1} = \alpha_{L2} = 0.2$ and $\alpha_{H1} = \alpha_{H2} = 0.5$ in equation (8.8)

Grid	N	average NI	average GI per NI	average time per step (sec.)
33^2	4225	3.2777	2.3724	0.0712010817
65^2	16641	3.2707	2.3002	0.277762508
129^2	66049	3.2697	2.3819	1.13359706
257^2	263169	2.5225	2.7941	3.689646612

The solution results of the linear momentum system using our preconditioner P8 with GMRES are reported in Table 8.9. The GMRES iterations in this case slightly increased when the grid size increased. However, the running time still almost optimal. Furthermore, the GMRES required slightly more iterations to converge and therefore the computational time slightly increased compared with the previous experiment (see Table 8.5).

Table 8.9: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system (8.33) using P8 preconditioned GMRES on regular grids: GI is GMRES iterations (the relative tolerance for GMRES is $tol = 1e-3$), N is the number of unknowns. Here $\alpha_{L1} = \alpha_{L2} = 0.2$ and $\alpha_{H1} = \alpha_{H2} = 0.5$ in equation (8.8)

Grid	N	average GI	average Time per step (sec.)
33^2	43339	29.9980	0.995721336
65^2	170635	33.4995	4.0404617139
129^2	677131	32.6883	16.60307104
257^2	2697739	41.7562	73.0833657

The computational cost and the memory requirement for solving the whole model in this second test are presented in Table 8.10. Both the CPU time and the memory requirement grow linearly with N. This experiment required slightly larger CPU time and memory than the previous experiment (see Table 8.6).

Table 8.10: The CPU time and percentage of memory cost for the whole model on regular grid using preconditioned GMRES: the linear system (8.33) is solved using P8, and the nonlinear systems solved using AMG preconditioning, with GMRES relative tolerance $1e-3$. Here $\alpha_{L1} = \alpha_{L2} = 0.2$ and $\alpha_{H1} = \alpha_{H2} = 0.5$ in equation (8.8)

Grid	MEM (%)	Time
33^2	1.4	18m36s
65^2	4.8	75m16s
129^2	15.9	309m2s
257^2	64.5	1324m54.344s

8.2.6.2 Unstructured Grids

In addition to the numerical results that are shown in the previous subsection, we obtained the results on a circle domain with unstructured triangular grid (as shown in Figure 5.6) in this subsection. The parameter values of α_{L1}, α_{L2} α_{H1} and α_{H2} in equation (8.8) are chosen to be $\alpha_{L1} = \alpha_{L2} = 0.1$ and $\alpha_{H1} = \alpha_{H2} = 1$.

Table 8.11 and Table 8.12 show the results that are obtained from solving the discrete reaction-diffusion equation systems for the nutrient and drug consecrations, respectively. In both tables the number of Newton iterations and the GMRES iterations are almost fixed for all grids sizes. The computational time on unstructured grids also behaves optimally.

Table 8.11: The average running times (in seconds), over 1000 time steps, required for solving the linear discrete system from equation (8.12) using AMG preconditioned GMRES on unstructured girds: N is the number of unknowns, NI is Newton iterations, GI is GMRES iterations and the relative tolerance for GMRES is $tol = 1e-3$. The tolerance for Newton is $1e-12$.

N	average NI	average GI per NI	average time per step (sec.)
9236	3.5415	2.8962	0.117828948
36627	3.7343	2.9904	0.52736337
145877	3.8322	3.2607	2.184051092

Table 8.12: The average running times (in seconds), over 1000 time steps, required for solving the linear discrete system from equation (8.14) using AMG preconditioned GMRES on unstructured girds: N is the number of unknowns, NI is Newton iterations, GI is GMRES iterations and the relative tolerance for GMRES is $tol = 1e-3$. The tolerance for Newton is $1e-12$.

N	average NI	average GI per NI	average time per step (sec.)
9236	3.2777	2.7742	0.1871657893
36627	3.2777	2.8574	0.75767909
145877	3.2777	2.9942	3.04032486

Table 8.13 presents the running time and the GMRES iterations that are required to solve the linear momentum system (8.33). The computational time again increased by factor of nearly 4 as N is quadrupled, which means that the solver requires approximately $\mathcal{O}(N)$ complexity. Moreover, The GMRES iterations in this case are again nearly independent of the grid size.

Table 8.13: The average running times (in seconds), over 1000 time steps, required for solving the linear algebraic system (8.33) using P8 preconditioned GMRES on fully unstructured grids: GI is GMRES iterations (the relative tolerance for GMRES is $tol = 1e-3$), N is the number of unknowns.

N	average GI	average Time per step (sec.)
94709	41.9560	2.7443692667
375506	44.5614	11.91469505
1495397	42.1758	46.629512359

The cost of the whole new model solution on fully unstructured grids is reported in Table 8.14. It can be observed that the CPU time and the memory requirement for the whole solvers scale approximately linearly and so behave as $\mathcal{O}(N)$.

Table 8.14: The CPU time and percentage of memory cost for the whole model on unstructured grids using preconditioned GMRES: the linear system (8.33) is solved using P8, and the nonlinear systems solved using AMG preconditioning, with GMRES relative tolerance $1e-3$, N is the number of unknowns.

N	MEM (%)	Time
94709	3.2	50m58s
375506	10.3	220m35s
1495397	34.4	867m25s

8.2.7 Further Numerical Results

In the previous chapter we presented the numerical simulations of the four-phase model. In a similar way the new five-phase model is considered in this subsection. We begin with a small cluster of the tumour cells that is seeded in the centre of the unstructured domain. The initial and boundary conditions are introduced in Subsection 8.2.3. In addition, the parameter values are shown in Table 4.1 and Table 8.1. Also, in this experiment we chose $t_0 = 10$, $t_{max} = 105$, $t_1 = 200$ and $dmax = 1$ in equation (8.3).

The numerical simulations in Figure 8.7 illustrates how the volume fraction of all the five phases develop with time on the circular domain. Figure 8.8, Figure 8.9 and Figure 8.10 present the evolution of the phase fluxes, pressures and the nutrient and drug concentrations, respectively, which also generate the characteristic tumour growth pattern. Initially, the simulation starts with a seed cluster of tumour cells in the middle of healthy tissue (not shown). In this model the drug starts to affect the tumour cells at time $t = 10$ and it reaches to the maximum value when $t = 105$, which is approximately 0.7 when $dmax = 1$ in equation (8.3), then the drug value goes down to 0 (see Figure 8.10). So, after $t = 200$ the tumour cells are no longer under the influence of the drug, which leads to rapid tumour growth after this time (see Figure 8.7). Moreover, from Figure 8.7 it can be seen that all tumour cells are susceptible to the drug and that this drug has the ability to impact on the tumour size. Since HS tumour cells are more susceptible to this drug than LS tumour cells, this has an obvious effect on the size of the tumour for each of them. So, LS tumour cells grow faster that HS tumour cells. Moreover, LS tumour cells also grow faster than the healthy cells since the proliferation and death rates of the LS tumour cells are assumed to be double and half the values of the proliferation and death rates of the healthy cells, respectively. In this case, LS tumour cells absorb the extracellular material more than HS tumour cells during their growth. This leads to a fall in the extracellular material θ_4 and therefore leads to decreasing the healthy cells' birth rate (see Figure 8.8).

Figure 8.7: Evolution of the volume fraction for each phase arranged from the top row to the bottom row: health cells θ_1, tumour cells θ_2, blood vessels θ_3, extracellular material θ_4 and tumour cells θ_5, with increasing time from left to right, t=100, 200 and 300. $\alpha_{L1} = \alpha_{L2} = 0.2$ and $\alpha_{H1} = \alpha_{H2} = 0.3$ in equation (8.8). The number of unknowns in the momentum system equal to 76237.

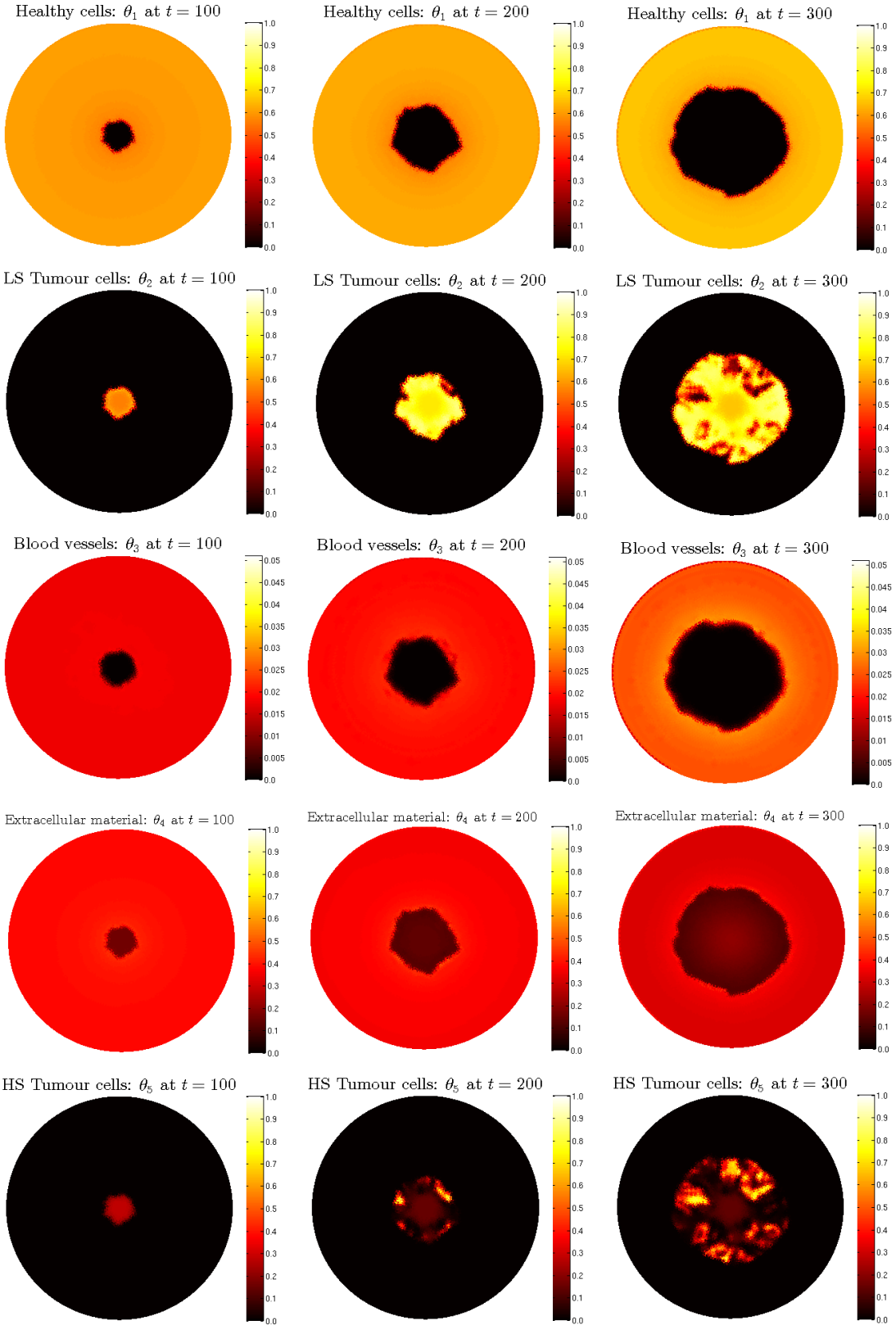


Figure 8.8: Evolution of the fluxes of phases, arranged from the top row to the bottom row: healthy cells $\theta 1\vec{u}_1'$, tumour cells $\theta_2\vec{u}_2'$ and extracellular material $\theta_4\vec{u}_4'$, with increasing the time from left to right, t=100, 200 and 300. The number of unknowns in the momentum system equal to 76237.

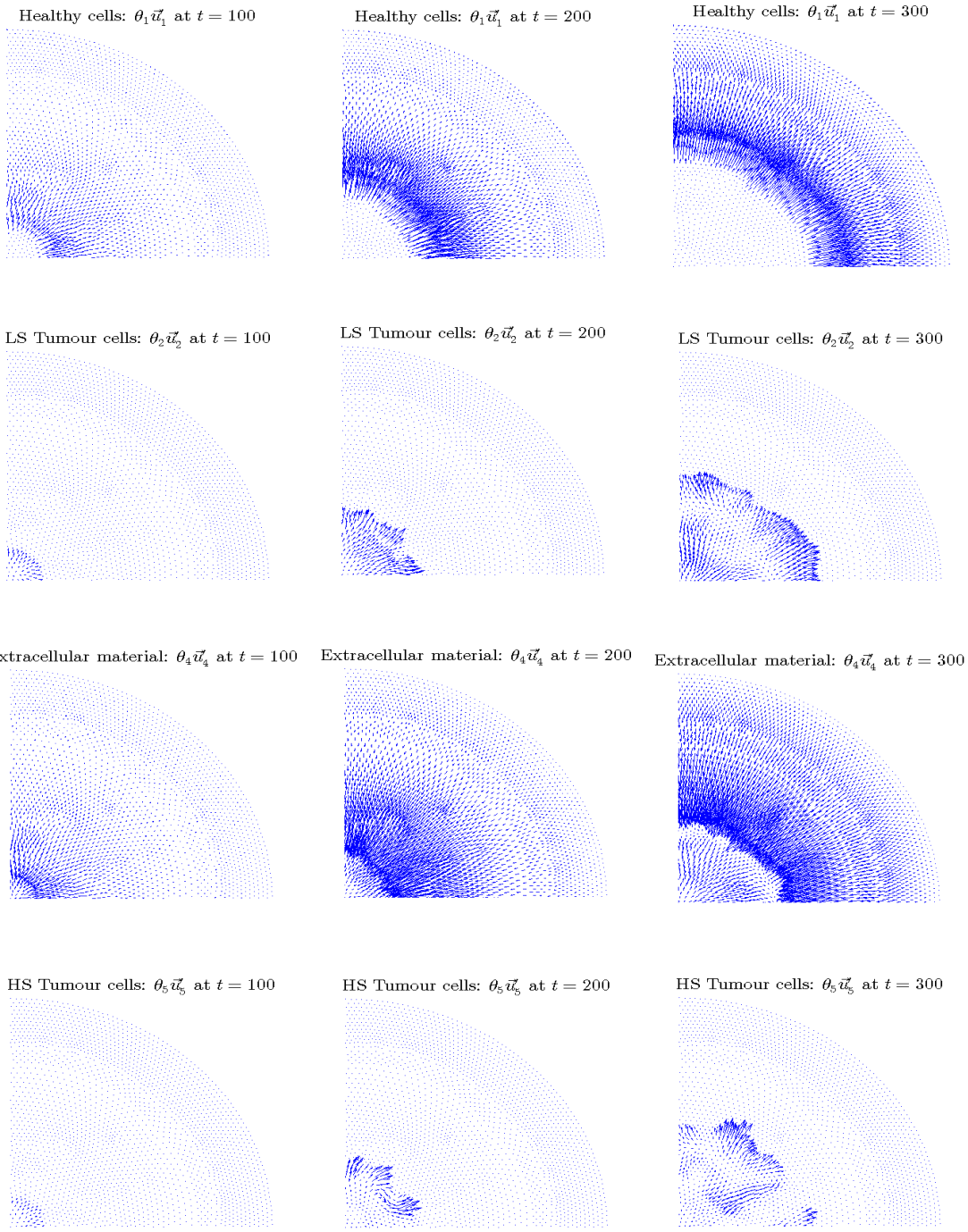


Figure 8.9: Evolution of the pressures arranged from the top row to the bottom row: healthy cells, LS tumour cells and HS tumour cells $p_1 = p_2 = p_5$ and extracellular material (ECM) p_4, with increasing time from left to right, t=100, 200 and 300. The blood vessels pressure p_3 is zero in this model. The number of unknowns in the momentum system equal to 76237.

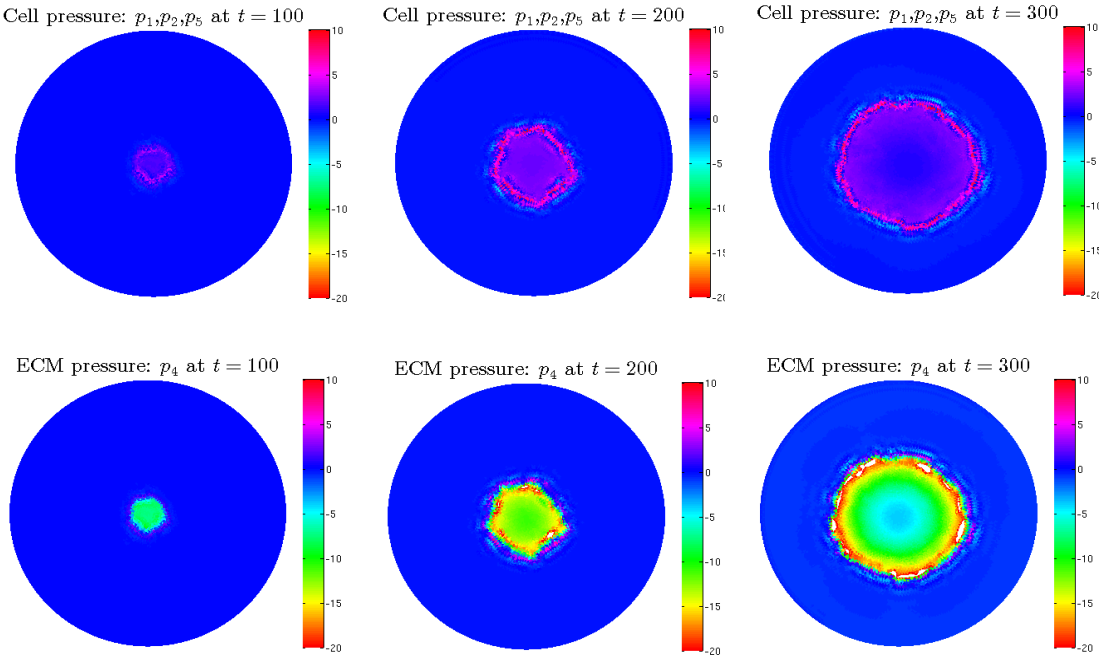
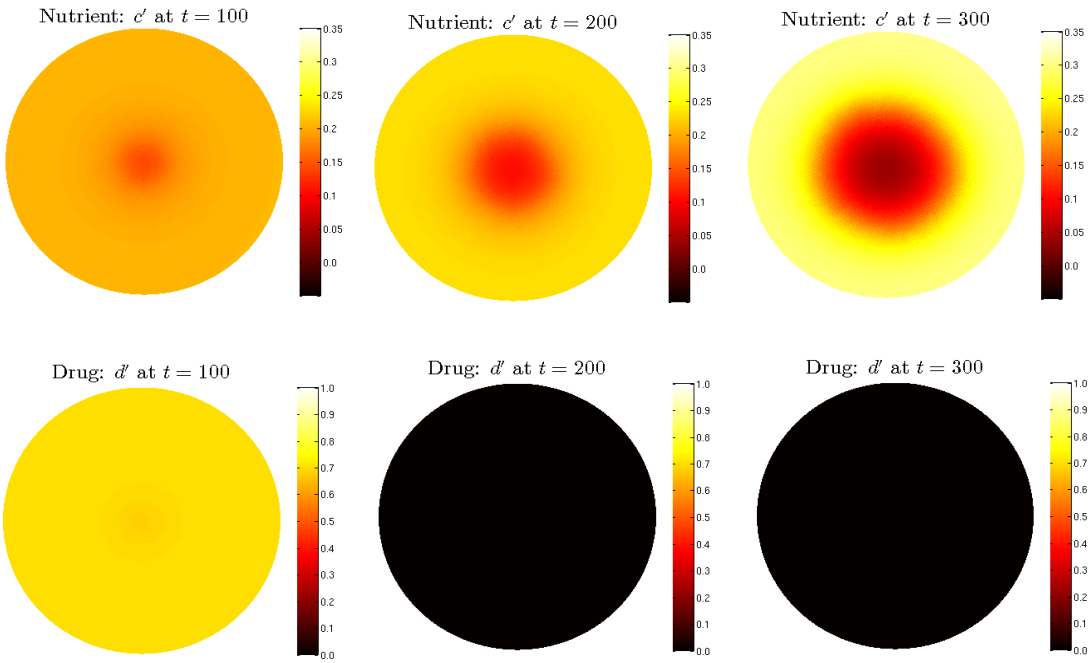
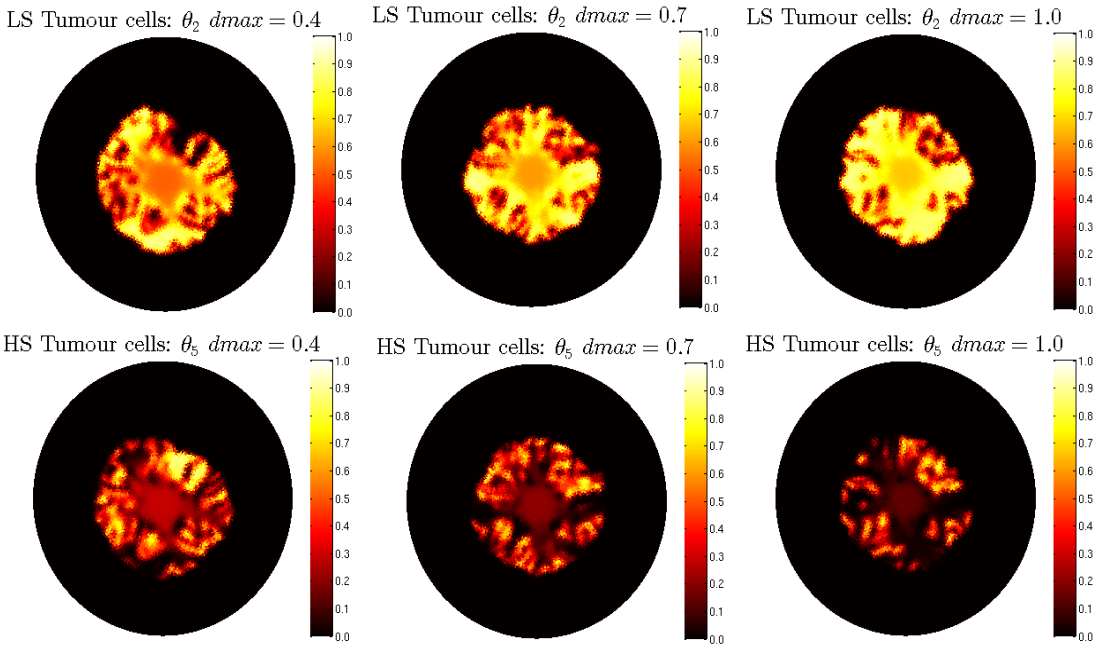


Figure 8.10: Evolution of the nutrient and drug concentrations with increasing time from left to right, t=100, 200 and 300. The number of unknowns in the momentum system equal to 76237.



For further investigation, we test different values of dmax in equation (8.3) and the results are presented in Figure 8.11. From this figure we can observe that when the value of dmax increases the tumour cells with high susceptibility to the drug decrease. This is because the value of the drug increases with the value of dmax. When the value of the drug is small the impact of the drug on the tumour size is very weak and this leads to the LS tumour and the HS tumour being of much closer sizes, while the difference between the two sizes appears more clearly when the value of the drug is large.

Figure 8.11: Evolution of the volume fraction of tumour cells on an unstructured grid: we vary dmax in equation (8.3). All plots show solutions at the same time, t=300. The number of unknowns in the momentum system equal to 76237.



8.3 Discussion

In this chapter we demonstrate that the optimal preconditioning approach that we developed for the four-phase model with a nutrient may be naturally extended to more general multiphase models. This is achieved by considering two extensions of [57]. Firstly, the mathematical model of vascular tumour growth is extended to include drug delivery, diffusion and uptake. Secondly, the mathematical model is extended to simulate five phases, which are: healthy cells, LS tumour cells, HS tumour cells, blood vessels and extracellular material.

We were able to demonstrate very clearly that the optimal performance of the AMG preconditioned Newton-Krylov solver for the nutrient also worked optimally for the drug diffusion. Perhaps more significantly, we were able to show that preconditioner P7, extended in the obvi-

ous way to P8 for the five-phase problem, also continued to perform optimally. We hypothesize that extending the model to even greater number of phases would not present any difficulties for this preconditioning approach.

In the final chapter of this thesis we summarize our findings and conclusions and suggest related extensions of this research.

Chapter 9

Conclusions and Future Work

In the final chapter of this thesis, we present conclusions of our work and discussion some possible areas for future research.

9.1 Conclusions

In this thesis, we have studied the multiphase model of vascular tumour growth in two-dimensions, which is presented by Hubbard and Byrne [57]. The objective of our thesis was to develop efficient computational algorithms for the numerical approximation of PDE systems describing multiphase flow. In particular to improve the efficiency of the numerical methods used in [57] in order to reach optimal efficiency.

Chapter 1 provides a brief introduction of the research area also included the achievements of this thesis and overview of the thesis chapters.

In Chapter 2 different types of models for tumour growth are discussed: continuum, discrete and hybrid representations. The review of the related work is focused mainly on continuum mathematical models of tumour growth. Furthermore, the specific mathematical model chosen for this research is introduced in Section 2.2, which includes three governing equation systems: the mass balance equations (2.2) for the volume fraction of each phase, the momentum balance equations (2.4) for the flow of each phase (velocities and pressures) and finally a nonlinear reaction diffusion equation (2.6) for the nutrient concentration. This model constitutes a four-phase model of vascular tumour growth in two spatial dimensions.

In Chapter 3 some scientific computing tools are introduced, which are related to our work. Discretization methods for PDEs focused on the Finite Element Method (FEM) and the Finite Volume Method (FVM). Also, the solution of linear and nonlinear systems is described in detail

in Section 3.2 and Section 3.4. Section 3.3 of this chapter introduced preconditioning methods.

In Chapter 4 we provided a review of some related works that have used the numerical schemes for solving multiphase models, especially multiphase models of tumour growth. Furthermore, the numerical schemes for the chosen model, which is presented in Chapter 2, are discussed in Section 4.2. The initial conditions, the boundary conditions and the parameter values, that are a requirement for the validation solutions, are presented in Section 4.3

In Chapter 5 an optimally preconditioned Newton-GMRES algorithm is developed, with a preconditioner based on algebraic multigrid (AMG), to efficiently solve the discrete system resulting from the FEM approximation of the nonlinear reaction-diffusion equation. The MUMPS sparse direct solver, used in [57], is replaced using our algorithm which allows us to solve systems of larger size and with greater efficiency. We concluded from this chapter that the MUMPS solver is much slower than our algorithm based on AMG preconditioned GMRES. Moreover, MUMPS is unable to solve very large problems due to its memory requirements.

In Chapter 6 we developed and applied a new block-preconditioned algorithm to efficiently solve the discrete algebraic system resulting from Taylor-Hood FEM approximation of the momentum balance equations. The MUMPS sparse direct solver used in [57] is replaced using our new preconditioner which again allows us to solve systems of larger size and with greater efficiency. Our novel preconditioner takes a block-upper-triangular form, inspired by [114], which is created by dropping all the entries under the block diagonal of the coefficient matrix, and using the pressure mass matrix (mp) for the final diagonal block. The diagonal blocks of this matrix are solved using one V-cycle of the AMG method, and mp is solved trivially. Furthermore, it is observed experimentally that the preconditioner that has been developed and implemented (i.e. P7 using the diagonal of mp) shows close to optimal convergence behaviour. Also in this chapter we noted that the MUMPS solver is much slower than our new preconditioned GMRES, and is unable to solve very large problems due to its less efficient use of memory.

In Chapter 7 the validation of our results against the existing results in [57] have been presented. Our model provided an accurate solution when compared with the existing codes. This chapter also includes simulation results (see Section 7.2) demonstrating the accuracy and efficiency of the solver (see Section 7.3 and Section 7 respectively). We concluded that our results were computed with a dramatic saving of computational time and memory usage compared with the MUMPS solver. Moreover, we demonstrated that the additional levels of mesh resolution that are permitted as a result of this improved efficiency do indeed lead to more accurate simulations.

In Chapter 8 the optimal preconditioning approach that we developed in Chapter 6 for the four-phase model with a nutrient has been extended to a more general multiphase model. Here, the mathematical model of vascular tumour growth is extended to include drug delivery, diffusion and uptake (see Section 8.1), and to simulate five phases, which are: healthy cells, LS tumour cells, HS tumour cells, blood vessels and extracellular material (see Section 8.2). We concluded that the optimal performance of the AMG preconditioned Newton-Krylov solver for

the nutrient also worked optimally for the drug diffusion. Moreover, we were able to show that preconditioner P7, extended in the obvious way to P8 for the five-phase problem, also continued to perform optimally. We hypothesize that extending the model to even greater numbers of phases would not present any difficulties for this preconditioning approach.

9.2 Future Work

We believe that this work opens possibilities for future research for those interested in this area. There are two main possible further works:

Firstly, exploiting the multiphase models in 2D:

- Using greater range of simulations across different initial conditions. There are two more sets of initial conditions in [57] that we have not considered in this thesis.

- Using different numbers of phases: it is possible to model multiphase problems with many more phases, possibly up to as many as ten [55].

- Modelling different drug performance, for example attempting to use the pharmacology of a drug such as in [49]. It is also possible to vary the delivery mechanism; in this thesis the drug is assumed to delivered to the tumour uniformly over a fixed interval of fixed time. It is possible to model the drug being delivered to the centre of the tumour or to the boundary of the tumour, etc.

- Using different 2D domains. In our work the domains of the numerical simulation were a circular domain with unstructured triangular elements and a square domain with regular triangular elements. It would be easy to extend to an unstructured mesh covering a different, more complex, geometry.

In summary, the future work in 2D would be to exploit the capability of the numerical solver to investigate the strengths and limitations of this family of multiphase models.

Secondly, it is possible extend the existing model to 3D, still using our efficient algorithms. The important issues are extending the computational algorithms to 3D problems, and making them efficient enough to be practical for real computations. In three-dimensions, for the four-phase model, the mass balance equation and the reaction-diffusion equation would be relatively straightforward to extend to 3D using tetrahedral meshes. The momentum balance equations will need to include three velocities for each phase, (u,v,w), and one pressure variable. Moreover the same discretization schemes of these PDEs can be used in 3D (i.e quadratic velocities and linear pressures).

The block-matrix system for the discrete momentum equations in three-dimensions may be

expressed in the form:

$$\begin{pmatrix} k_{xx11} & k_{xy11} & k_{xz11} & k_{xx12} & 0 & 0 & k_{xx14} & 0 & 0 & k_{xx13} & 0 & 0 & C_{x1} \\ k_{yx11} & k_{yy11} & k_{yz11} & 0 & k_{yy12} & 0 & 0 & k_{yy14} & 0 & 0 & k_{yy13} & 0 & C_{y1} \\ k_{zx11} & k_{zy11} & k_{zz11} & 0 & 0 & k_{zz12} & 0 & 0 & k_{zz14} & 0 & 0 & k_{zz13} & C_{z1} \\ k_{xx21} & 0 & 0 & k_{xx22} & k_{xy22} & k_{xz22} & k_{xx24} & 0 & 0 & k_{xx23} & 0 & 0 & C_{x2} \\ 0 & k_{yy21} & 0 & k_{yx22} & k_{yy22} & k_{yz22} & 0 & k_{yy24} & 0 & 0 & k_{yy23} & 0 & C_{y2} \\ 0 & 0 & k_{zz21} & k_{zx22} & k_{zy22} & k_{zz22} & 0 & 0 & k_{zz24} & 0 & 0 & k_{zz23} & C_{z2} \\ k_{xx41} & 0 & 0 & k_{xx42} & 0 & 0 & k_{xx44} & k_{xy44} & k_{xz44} & k_{xx43} & 0 & 0 & C_{x4} \\ 0 & k_{yy41} & 0 & 0 & k_{yy42} & 0 & k_{yx44} & k_{yy44} & k_{yz44} & 0 & k_{yy43} & 0 & C_{y4} \\ 0 & 0 & k_{zz41} & 0 & 0 & k_{zz42} & k_{zx44} & k_{zy44} & k_{zz44} & 0 & 0 & k_{yy43} & C_{z4} \\ k_{xx31} & 0 & 0 & k_{xx32} & 0 & 0 & k_{xx34} & 0 & 0 & k_{xx33} & k_{xy33} & k_{xz33} & 0 \\ 0 & k_{yy31} & 0 & 0 & k_{yy32} & 0 & 0 & k_{yy34} & 0 & k_{yx33} & k_{yy33} & k_{yz33} & 0 \\ 0 & 0 & k_{zz31} & 0 & 0 & k_{zz32} & 0 & 0 & k_{zz34} & k_{zx33} & k_{zy33} & k_{zz33} & 0 \\ B_{x1}^T & B_{y1}^T & B_{z1}^T & B_{x2}^T & B_{y2}^T & B_{z2}^T & B_{x4}^T & B_{y4}^T & B_{z4}^T & B_{x3}^T & B_{y3}^T & B_{z3}^T & 0 \end{pmatrix} \begin{pmatrix} u_{x1} \\ u_{y1} \\ u_{z1} \\ u_{x2} \\ u_{y2} \\ u_{z2} \\ u_{x4} \\ u_{y4} \\ u_{z4} \\ u_{x3} \\ u_{y3} \\ u_{z3} \\ p_4 \end{pmatrix} = \begin{pmatrix} fx_1 \\ fy_1 \\ fz_1 \\ fx_2 \\ fy_2 \\ fz_2 \\ fx_4 \\ fy_4 \\ fz_4 \\ fx_3 \\ fy_3 \\ fz_3 \\ 0 \end{pmatrix}.$$

Our preconditioner in 3D would then be:

$$P = \begin{pmatrix} k_{xx11} & k_{xy11} & k_{xz11} & k_{xx12} & 0 & 0 & k_{xx14} & 0 & 0 & k_{xx13} & 0 & 0 & C_{x1} \\ 0 & k_{yy11} & k_{yz12} & 0 & k_{yy12} & 0 & 0 & k_{yy14} & 0 & 0 & k_{yy13} & 0 & C_{y1} \\ 0 & 0 & k_{zz11} & 0 & 0 & k_{zz12} & 0 & 0 & k_{zz14} & 0 & 0 & k_{zz13} & C_{z1} \\ 0 & 0 & 0 & k_{xx22} & k_{xy22} & k_{xz22} & k_{xx24} & 0 & 0 & k_{xx23} & 0 & 0 & C_{x2} \\ 0 & 0 & 0 & 0 & k_{yy22} & k_{yz22} & 0 & k_{yy24} & 0 & 0 & k_{yy23} & 0 & C_{y2} \\ 0 & 0 & 0 & 0 & 0 & k_{zz22} & 0 & 0 & k_{zz24} & 0 & 0 & k_{zz23} & C_{z2} \\ 0 & 0 & 0 & 0 & 0 & 0 & k_{xx44} & k_{xy44} & k_{xz44} & k_{xx43} & 0 & 0 & C_{x4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{yy44} & k_{yz44} & 0 & k_{yy43} & 0 & C_{y4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{zz44} & 0 & 0 & k_{zz43} & C_{z4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{xx33} & k_{xy33} & k_{xz33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{yy33} & k_{yz33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{zz33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & mp \end{pmatrix}.$$

It would be necessary to implement and test this in 3D however we would still expect to see close to optimal performance when the diagonal blocks (other than mp) are approximated by the application of a single AMG V-cycle.

Bibliography

[1] Hsl. a collection of fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk/,.

[2] J. A. Adam. A simplified mathematical model of tumor growth. *Mathematical biosciences*, 81(2):229–244, 1986.

[3] J. A. Adam. A mathematical model of tumor growth. II. Effects of geometry and spatial nonuniformity on stability. *Mathematical Biosciences*, 86(2):183–211, 1987.

[4] J. A. Adam. A mathematical model of tumor growth. III. Comparison with experiment. *Mathematical biosciences*, 86(2):213–227, 1987.

[5] J. A. Adam and S. Maggelakis. Mathematical models of tumor growth. IV. Effects of a necrotic core. *Mathematical biosciences*, 97(1):121–136, 1989.

[6] D. Ambrosi and L. Preziosi. On the closure of mass balance models for tumor growth. *Mathematical Models and Methods in Applied Sciences*, 12(05):737–754, 2002.

[7] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer methods in applied mechanics and engineering*, 184(2):501–520, 2000.

[8] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[9] A. R. A. Anderson and M. A. J. Chaplain. Continuous and discrete mathematical models of tumor-induced angiogenesis. *Bulletin of Mathematical Biology*, 60(5):857–899, 1998.

[10] R. P. Araujo and D. L. S. McElwain. A history of the study of solid tumour growth: the contribution of mathematical modelling. *Bulletin of Mathematical Biology*, 66(5):1039–1091, 2004.

[11] O. Axclsson and P. S. Vassilevski. Algebraic multilevel preconditioning methods. i. *Numerische Mathematik*, 56(2-3):157–177, 1989.

[12] O. Axelsson and P. S. Vassilevski. Algebraic multilevel preconditioning methods, ii. *SIAM Journal on Numerical Analysis*, 27(6):1569–1590, 1990.

[13] N. S. Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Mathematics and Mathematical Physics*, 6(5):101–135, 1966.

[14] R. E. Bank, T. F. Dupont, and H. Yserentant. The hierarchical basis multigrid method. *Numerische Mathematik*, 52(4):427–458, 1988.

[15] E. Barkanov. Introduction to the finite element method. Technical report, Institute of Materials and Structures Faculty of Civil Engineering Riga Technical University, 2001.

[16] T. Barth and M. Ohlberger. Finite volume methods: Foundation and analysis. *Encyclopedia of Computational Mechanics*.

[17] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[18] J. Boyle, M. Mihajlovic, and J. Scott. Hsl mi20: an efficient amg preconditioner. Technical report, Citeseer, 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. In *Preliminary Proceedings for International Multigrid Conference*, Copper Mountain, Colorado, 1983.

[21] A. Brandt. Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(1):23–56, 1986.

[22] A. Brandt, S. MacCormick, and J. Ruge. Algebraic multigrid (AMG) for Automatic Multigrid Solutions with Application to Geodetic Computations. Technical report, Institute for Computational Studies, Fort Collins, CO, 1983.

[23] A. Brandt, S. McCoruick, and J. Huge. *Algebraic Multigrid (AMG) for sparse matrix equations*. CUP Archive, 1985.

[24] C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel. *Boundary element techniques: theory and applications in engineering*. Springer Science & Business Media, 2012.

[25] C. E. Brennen. *Fundamentals of Multiphase Flows*. Cambridge University Press, 2005.

[26] C. J. W. Breward, H. M. Byrne, and C. E. 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.

[27] C. J. W. Breward, H. M. Byrne, and C. E. Lewis. A multiphase model describing vascular tumour growth. *Bulletin of Mathematical Biology*, 65(4):609–640, 2003.

[28] W. L. Briggs, S. F. McCormick, et al. *A multigrid tutorial*. SIAM, 2nd edition, 2000.

[29] P. N. Brown and Y. Saad. Hybrid krylov methods for nonlinear systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 11(3):450–481, 1990.

[30] H. M. Byrne. Dissecting cancer through mathematics: from the cell to the animal model. *Nature reviews. Cancer*, 10(3):221, 2010.

[31] H. M. Byrne, T. Alarcon, M. R. Owen, S. D. Webb, and P. K. Maini. Modelling aspects of cancer dynamics: a review. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 364(1843):1563–1578, 2006.

[32] H. M. Byrne, J. R. King, D. L. S. McElwain, and L. Preziosi. A two-phase model of solid tumour growth. *Applied Mathematics Letters*, 16(4):567–573, 2003.

[33] H. M. Byrne and L. Preziosi. Modelling solid tumour growth using the theory of mixtures. *Mathematical Medicine and Biology*, 20(4):341–366, 2003.

[34] C. Canuto, M. Y. Hussaini, A. M. Quarteroni, A. Thomas Jr, et al. *Spectral methods in fluid dynamics*. Springer Science & Business Media, 2012.

[35] P. G. Ciarlet. *The finite element method for elliptic problems*. SIAM, Philadelphia, PA, 2002.

[36] W. Dahmen and L. Elsner. Algebraic multigrid methods and the schur complement. In *Robust Multi-Grid Methods*, volume 23 of *Notes on Numerical Fluid Mechanics*, chapter Algebraic multigrid methods and the Schur complement, pages 58–68. Vieweg+Teubner Verlag, 1989.

[37] A. Das, D. Lauffenburger, H. Asada, and R. D. Kamm. A hybrid continuum-discrete modelling approach to predict and control angiogenesis: analysis of combinatorial growth factor and matrix effects on vessel-sprouting morphology. 368(1921):2937–2960, 2010.

[38] J. W. Demmel. Superlu users' guide. *Lawrence Berkeley National Laboratory*, 2011.

[39] J. Dongarra and F. Sullivan. Guest editors introduction: The top 10 algorithms. *Computing in Science & Engineering*, 2(1):22–23, 2000.

[40] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers : with applications in incompressible fluid dynamics*. Numerical mathematics and scientific computation. Oxford University Press, Oxford, New York, 2005.

[41] M. Embree. How descriptive are GMRES convergence bounds? Technical report, 99/08, Oxford University Computing Laboratory, Oxford, UK, 1999.

[42] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Computational Mathematics and Mathematical Physics*, 1(4):1092–1096, 1962.

[43] R. P. Fedorenko. The speed of convergence of one iterative process. *USSR Computational Mathematics and Mathematical Physics*, 4(3):227–235, 1964.

[44] 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.

[45] A. Gillman. *Fast direct solvers for elliptic partial differential equations*. PhD thesis, University of Colorado, 2011.

[46] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, 2nd edition, 1989.

[47] N. I. M. Gould, J. A. Scott, and Y. Hu. A numerical evaluation of sparse direct solvers for the solution of large sparse, symmetric linear systems of equations. Technical report, Council for the Central Laboratory of the Research Councils, Oxfordshire, UK, 2005.

[48] H. P. Greenspan. Models for the growth of a solid tumor by diffusion. *Studies in Applied Mathematics*, 51(4):317–340, 1972.

[49] C. M. Groh, M. E. Hubbard, P. F. Jones, P. M. Loadman, N. Periasamy, B. D. Sleeman, S. W. Smye, C. J. Twelves, and R. M. Phillips. Mathematical and computational models of drug transport in tumours. *Journal of The Royal Society Interface*, 11(94):20131173, 2014.

[50] A. Guermouche and J.-Y. L'excellent. Constructing memory-minimizing schedules for multifrontal methods. *ACM Transactions on Mathematical Software (TOMS)*, 32(1):17–32, 2006.

[51] L. d. N. Guimarães, L. M. Costa, E. A. Santos, A. P. Costa, and I. D. d. S. PONTES FILHO. Multiphase flow of water and oil in heterogeneous reservoir. *MECOM 2002*, pages 670–682, 2002.

[52] M. H. Gutknecht. A brief introduction to Krylov space methods for solving linear systems. In *Frontiers of Computational Science*, pages 53–62. Springer Berlin Heidelberg, 2007.

[53] C. S. Hogea, B. T. Murray, and J. A. Sethian. Simulating complex tumor dynamics from avascular to vascular growth using a general level-set method. *Journal of Mthematical Biology*, 53(1):86–134, 2006.

[54] W. Z. Huang. Convergence of algebraic multigrid methods for symmetric positive definite matrices with weak diagonal dominance. *Applied Mathematics and Computation*, 46(2):145–164, 1991.

[55] M. E. Hubbard. Private communication. 2015.

[56] M. E. Hubbard. Multidimensional slope limiters for MUSCL-type finite volume schemes on unstructured grids. *Journal of Computational Physics*, 155(1):54–74, 1999.

[57] M. E. Hubbard and H. M. Byrne. Multiphase modelling of vascular tumour growth in two spatial dimensions. *Journal of Theoretical Biology*, 316(16):70–89, 2013.

[58] M. E. Hubbard, M. Jove, P. M. Loadman, R. M. Phillips, C. J. Twelves, and S. W. Smye. Drug delivery in a tumour cord model: a computational simulation. *Royal Society Open Science*, 4(5):170014, 2017.

[59] T. L. Jackson and H. M. Byrne. A mathematical model to study the effects of drug resistance and vasculature on the response of solid tumors to chemotherapy. *Mathematical Biosciences*, 164(1):17–38, 2000.

[60] J. Jeon, V. Quaranta, and P. T. Cummings. An off-lattice hybrid discrete-continuum model of tumor growth and invasion. *Biophysical journal*, 98(1):37–47, 2010.

[61] P. K. Jimack. The finite element method sectionii: two-dimensional problems and incompressible flow. University of Leeds.

[62] R. K. J.M. Hyman and J. Scovel. High order finite volume approximations of differential operators on nonuniform grids. *Physica D.,*, 60(1-4):112–138,, 1992.

[63] J. J. Kaluarachchi and J. C. Parker. An efficient finite element method for modeling multiphase flow. *Water Resources Research*, 25(1):43–54, 1989.

[64] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia, 1995.

[65] C. T. Kelley. *Solving Nonlinear Equations with Newton's Method*. SIAM, Philadelphia, PA, 2003.

[66] A. Klawonn and G. Starke. Block triangular preconditioners for nonsymmetric saddle point problems: field-of-values analysis. *Numerische Mathematik*, 81(4):577–594, 1999.

[67] 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.

[68] D. Kopriva. *Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers*. Springer Science & Business Media, 2009.

[69] Y. u. Kuznetsov. Overlapping domain decomposition methods for FE-problems with elliptic singular perturbed operators. In G. M. J. P. . R. Glowinski, Y. A. Kuznetsov

and e. O. Widlund, editors, *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, Proceedings in Applied Mathematics 51*, Philadelphia, PA, 1991. SIAM.

[70] Y. u. A. Kuznetsov. Algebraic multigrid domain decomposition methods. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 4(5):351–380, 1989.

[71] R. J. LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge University Press, 2002.

[72] J. S. Lowengrub, H. B. Frieboes, F. Jin, Y. L. Chuang, X. Li, P. Macklin, S. M. Wise, and V. Cristini. Nonlinear modelling of cancer: bridging the gap between cells and tumours. *Nonlinearity*, 23(1):R1, 2010.

[73] P. Macklin, S. McDougall, A. R. A. Anderson, M. A. J. Chaplain, V. Cristini, and J. Lowengrub. Multiscale modelling and nonlinear simulation of vascular tumour growth. *Journal of Mathematical Biology*, 58(4-5):765–798, 2009.

[74] S. Maggelakis and J. Adam. Mathematical model of prevascular growth of a spherical carcinoma. *Mathematical and Computer Modelling*, 13(5):23–38, 1990.

[75] P. Mascheroni, DP.Boso, C. Stigliano, M. Carfagna, L. Preziosi, P. Decuzzi, and BA.Schrefler. A parametric study of multiphase porous media model for tumor spheroids and envoronment interactions. *ECCOMAS Congress 2016, Crete Island, Greece*, 2016.

[76] P. Mascheroni, C. Stigliano, M. Carfagna, D. P. Boso, L. Preziosi, P. Decuzzi, and B. A. Schrefler. Predicting the growth of glioblastoma multiforme spheroids using a multiphase porous media model. *Biomechanics and Modeling in Mechanobiology*, 15(5):1215–1228, 2016.

[77] A. I. Minchinton and I. F. Tannock. Drug penetration in solid tumours. *Nature Reviews Cancer*, 6(8):583–592, 2006.

[78] R. F. Mudde and O. Simonin. Two-and three-dimensional simulations of a bubble plume using a two-fluid model. *Chemical Engineering Science*, 54(21):5061–5069, 1999.

[79] 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.

[80] S. Nagrath. *Adaptive stabilized finite element analysis of multi-phase flows using level set approach*. PhD thesis, Rensselaer Polytechnic Institute Troy, New York, 2004.

[81] H. Namazi, V. V. Kulish, A. Wong, and S. Nazeri. Mathematical based calculation of drug penetration depth in solid tumors. *BioMed research international*, 2016, 2016.

[82] Y. Notay. An aggregation-based algebraic multigrid method. *Electronic transactions on numerical analysis*, 37(6):123–146, 2010.

[83] M. Orme and M. Chaplain. A mathematical model of vascular tumour growth and invasion. *Mathematical and Computer Modelling*, 23(10):43–60, 1996.

[84] J. Osborne, R. D. O'Dea, J. Whiteley, H. Byrne, and S. Waters. The influence of bioreactor geometry and the mechanical environment on engineered tissues. *Journal of Biomechanical Engineering*, 132(5):051006, 2010.

[85] J. M. Osborne and J. P. Whiteley. A numerical method for the multiphase viscous flow equations. *Computer Methods in Applied Mechanics and Engineering*, 199(49-52):3402–3417, 2010.

[86] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[87] S. Patankar and D. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806., 1972.

[88] F. H. Pereira, S. L. Lopes Verardi, and S. I. Nabeta. A fast algebraic multigrid preconditioned conjugate gradient solver. *Applied Mathematics and Computation*, 179(1):344–351, 2006.

[89] C. P. Please, G. J. Pettet, and D. L. S. McElwain. A new approach to modelling the formation of necrotic regions in tumours. *Applied Mathematics Letters*, 11(3):89–94, 1998.

[90] C. Popa. On smoothing property of the SOR relaxation for algebraic multigrid method. *Studii si Cercetari Matematice*, 41(5):399–406, 1989.

[91] J. N. Reddy. *An introduction to the finite element method*. McGraw-Hill, New York, 3rd edition, 2005.

[92] K. A. Rejniak and A. R. Anderson. Hybrid models of tumor growth. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 3(1):115–125, 2011.

[93] B. Riviere. *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation*. SIAM, 2008.

[94] J. E. Roberts and J. M. Thomas. *Handbook of numerical analysis*, volume 2, chapter Mixed and hybrid methods, pages 523–633. Gulf Professional Publishing, 1991.

[95] Y. Saad. Sparskit: A basic tool kit for sparse matrix computations. 1990.

[96] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, PA, 2nd ed edition, 2003.

[97] Y. Saad and M. Schultz. Gmres :a generalized minimal residual algorithm for solving nonsymmetric linear systems. *Journal on Scientific Computing*, 7(3):856–869, 1986.

[98] Y. Saad and H. A. Van Der Vorst. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1-2):1–33, 2000.

[99] G. Sciumè, S. Shelton, W. G. Gray, C. T. Miller, F. Hussain, M. Ferrari, P. Decuzzi, and B. A. Schrefler. A multiphase model for three-dimensional tumor growth. *New Journal of Physics*, 15(1):015005, 2013.

[100] M. Sefidgar, M. Soltani, K. Raahemifar, M. Sadeghi, H. Bazmara, M. Bazargan, and M. M. Naeenian. Numerical modeling of drug delivery in a dynamic solid tumor microvasculature. *Microvascular Research*, 99:43–56, 2015.

[101] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213, 1994.

[102] M. Steuperaert, G. Falvo D'Urso Labate, C. Debbaut, O. De Wever, C. Vanhove, W. Ceelen, and P. Segers. Mathematical modeling of intraperitoneal drug delivery: simulation of drug distribution in a single tumor nodule. *Drug Delivery*, 24(1):491–501, 2017.

[103] G. Strang and G. J. Fix. *An analysis of the finite element method*, volume 212. Prentice-Hall Englewood Cliffs, NJ, 1973.

[104] K. Stüben. Algebraic multigrid (amg): experiences and comparisons. *Applied Mathematics and Computation*, 13(3):419–451, 1983.

[105] P. Tracqui. Biophysical models of tumour growth. *Reports on Progress in Physics*, 72(5):056701, 2009.

[106] L. N. Trefethen and M. Embree. *Spectra and pseudospectra: the behavior of nonnormal matrices and operators*. Princeton University Press, 2005.

[107] T.Roose, S. J. Chapman, and P. K. Maini. Mathematical models of avascular tumor growth. *SIAM Review*, 49(2):179–208, 2007.

[108] H. A. Van der Vorst. *Iterative Krylov methods for large linear systems*, volume 13. Cambridge University Press, 2003.

[109] P. Vassilevski. Nearly optimal iterative methods for solving finite element elliptic equations based on the multilevel splitting of the matrix. Technical report, 1989.

[110] H. Versteeg and W. Malalasekera. An introduction to computational fluid dynamics,the finite volume method. *Longman Scientific and Technical*, 1995.

[111] C. Y. Wang and P. Cheng. A multiphase mixture model for multiphase, multicomponent transport in capillary porous mediai. model development. *International Journal of Heat and Mass Transfer*, 39(17):3607–3618, 1996.

[112] J. Ward and J. King. Mathematical modelling of avascular-tumour growth ii: modelling growth saturation. *Mathematical Medicine and Biology*, 16(2):171–211, 1999.

[113] A. Wathen. Realistic eigenvalue bounds for the galerkin mass matrix. *IMA Journal of Numerical Analysis*, 7(4):449–457, 1987.

[114] A. J. Wathen. Preconditioning and fast solvers for incompressible flow. Technical report, Oxford University Computing Laboratory, 2004.

[115] A. J. Wathen. Preconditioning. *Acta Numerica*, 24:329–376, 2015.

[116] 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.

[117] Y. X. Xiao, P. Zhang, and S. Shu. An algebraic multigrid method with interpolation reproducing rigid body modes for semi-definite problems in two-dimensional linear elasticity. *Journal of Computational and Applied Mathematics*, 200(2):637–652, 2007.

[118] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49(4):379–412, 1986.

[119] X. Zheng, S. M. Wise, and V. Cristini. Nonlinear simulation of tumor necrosis, neovascularization and tissue invasion via an adaptive finite-element/level-set method. *Bulletin of Mathematical Biology*, 67(2):211–259, 2005.