A Numerically Efficient Implementation of the DI-MCRD model in 3D

R.Q. Engelberts

A Numerically Efficient Implementation of the DI-MCRD model in 3D

by

R.Q. Engelberts

to obtain the degree of Master of Science,
in Computer Simulations for Science and Engineering,
at Delft University of Technology and Kungliga Tekniska Högskolan Stockholm,
to be defended publicly August 29th, 2023.

Student number:	4560752	
Course code:	WI5000COSSE	
Project duration:	September 2022 – July 2023	
Date final version:	2023/07/14	
Supervisors:	Dr. Ir. M.B. van Gijzen,	Delft University of Technology
	Dr. A. Jager	Erasmus MC
Thesis Committee:	Dr. Ir. E.G Rens	Second Assessor

Abstract

With breast cancer being the leading cause of death in the Netherlands, while also being expected to have double the amount of cases in the next ten years, it is vital that treatment is optimised. Over the last decade, research has been done to incorporate mathematical modelling in this process, especially in the case of HER2+ breast cancer patients. This aggressive form has poor chances of survival, but responds well to chemotherapy and is expected to be quite predictable. In previous works of N. Oudhof and E. Slingerland, a mechanically coupled reaction-diffusion model with an extension of chemotherapy was implemented in 2D, using three MRI scans to predict tumour response. The first two scans, taken right before and during treatment, are used to find patient-specific parameters corresponding to proliferation, tumour movement and chemotherapy efficacy. This calibrated model is then used to predict the third scan, taken at the end of treatment. Calibrating this model to individual patients takes up to a day, however.

This thesis aims to extend this model to a higher resolution 3D setting, which should also hand doctors predictions within a working day. To increase speed, the linear-elastic sub-problem of finding shear stress due to tissue types and tumour growth was first optimised. With a novel Laplacian preconditioner in the Conjugate Gradient method, using FFT's and a tridiagonal solver, the time needed was vastly improved. Second, the maximum order of error needed for accurate temporal integration was confirmed to be quite high. Hence a state-of-the-art Parareal implementation was made, using Runge-Kutta 4 and Crank Nicholson as the respective fine and course solver, which succeeded in being both faster and more accurate than simple first-order methods. Then it was found that the underdeterminedness of the problem is best tackled using Total Variation Regularisation on the proliferation parameter and Tikhonov Regularisation on the other two parameters. This ensures that unique solutions can be found in reasonable time, that properly reflect expectations of these parameters in practical settings. The best set of parameters were found fastest with Powell's Dog-leg method, for which a novel way of finding the Jacobian analytically was used.

On simulated data, the error in the amount of tumour cells in the third image went down to single-digit percentage rate, with a maximal shape correlation coefficient. This prediction can be made within a few hours, which means that the feasibility of solving this problem in practical settings has been established successfully. On the real data, the calibration succeeded in calibrating the model on the first two scans, but the predictions still have room of improvement. The most important suggestion of this work is that more research must be done in the verification of the suitability of this model to the available real data, using the techniques presented here. Further improvements can be made by exploring more possibilities of using parallelisation, and by obtaining more data. Before obtaining more data, one should investigate the impact of the timing of the scans on the calibration results.

Preface

Seven years, two bachelors, three masters and three internships, with this little paragraph of text marking the end of my studying life. I am grateful for all the experience, knowledge, maturity and friends I have gained along the way, and I am excited to see what the future will bring.

This thesis could not have been finished without the help of many others. My supervisor Martin always made me feel acknowledged with the progress meetings we had every week, and his advice has been vital to the success of our findings. I genuinely hope that there'll be someone to pick up on this wonderful project, and that Martin can be as much of a help to them as he was to me. While there weren't too many meetings with Agnes, I would also like to thanks her for her pivotal insights in oncology that we as mathematicians would never see. If we end up publishing this work, I hope that our minds combined can contribute to modern science and inspire researchers everywhere. Other gratitude goes to the support of my lovely girlfriend Isabelle, my roommates Diederick and Tijn, my parents, my brother, and everyone else I haven't named. When referring to myself as 'we' in this work, note that I am also referring to both myself and my cat Bezem.

P.S. Apologies for the colours used in the figures of this work, which were chosen to make them interpretable with my colourblindness.

R.Q. Engelberts
The Hague, July 2023

Contents

List of Figures

List of Tables

1 Introduction

According to the Integraal Kankercentrum Nederland, the amount of cancer cases in the Netherlands will almost double in the next ten years [2]. This is on top of the fact that it is already the leading cause of death here, as well as one of the leading causes of death worldwide [3]. Of all types, breast cancer is the most commonly diagnosed form of cancer, accounting for almost 700.000 deaths in 2020 [4]. The three standard ways of treating breast cancer are chemotherapy, surgery and radiotherapy, with most of the time a combination of these being used. Chemotherapy causes the tumour to shrink, after which it can more easily be removed surgically. However, chemotherapy also damages the healthy part of the body, which makes it vital to give a patient the right dose. With a hypothetically optimal dose, the tumour shrinks well enough to be easily removed, while also having as little damage to the body as possible.

With mathematical modelling, one can attempt to predict the response of the tumour on the administered drugs. This is a very complex process, and over the last decade more and more research has been done on this subject. In this work we focus on *HER2-positive* breast cancer, which is an aggressive form of breast cancer with relatively poor prognosis of survival. Meanwhile, these tumours do respond well to treatment focusing on reducing the amount of cancer cells, which facilitates tumour removal.

1.1. Project Description

In previous master theses of Nathalie Oudhof [5] and Eva Slingerland [1] a model was implemented to predict patient-specific tumour response with a data-set of MRI scans of patients of Erasmus MC. Specifically, this is the *Mechanically Coupled Reaction Diffusion model* (MCRD), with an extension called the *Drug-Included MCRD* (DI-MRCD). Using MRI scans of patients undergoing chemotherapy, taken right before treatment starts and after a few rounds, we want to predict the outcome of a third MRI scan a few rounds later. This thesis will continue this research, now focusing on an numerically efficient and accurate implementation. Efficiency is key here, as the models present can take over a day to get to their final prediction with 2D data. Most of the time is needed to tune the patient-specific parameters of the model. For usage in hospitals, this needs to be brought down to a couple of hours, and additionally the complete 3D data should be used. Hence an analysis on the bottlenecks of this simulation is crucial to realising a tool which can assist doctors in making a good treatment schedule.

The improvements on the implementation will be done by taking the DI-MCRD model apart, and identifying what steps are done in what way. Aside from the inverse problem of tuning the model to each patient, there is a sub-problem in the form of a linear-elastic equation. Furthermore, the model needs to be evaluated over time, which can be attained with either implicit or explicit methods of varying orders of accuracy. With state-of-the-art methods and clever programming, the goal of substantially reducing the time needed to run this model should be attainable. Research for this can both be done by investigating medical literature, but also literature in mechanical engineering, where they solve similar problems.

1.2. Research Questions

We will put the main questions we want answered in words, with first and foremost the formal research question

How can the 3D DI-MCRD model be implemented more efficiently, in order to tune the model in a few hours and accurately predict tumour response for HER2+ breast cancer patients?

To help us answering this, we have the following sub-questions

- Is CG faster than the direct method for solving the linear-elastic equation at our scale, and do extensions such as preconditioning an recycling speed it up even more?
- Is it faster to use explicit methods such as FE, RK4 for our temporal evaluations or implicit methods such as BE, CN, gen-α?
- What order of local truncation error is necessary to obtain accurate predictions on our time-scale?
- Are (locally) optimal solutions to the inverse problem on simulated data found fastest with LM, TRF, DL, or AS, and what regularisation technique should be used?
- Can we find (locally) optimal solutions to fitting the model to simulated data in a few hours?

If the answer to this last question is yes, then there is a subsequent last question which determines the utility of this project as a whole.

- Are the predictions of the fitted model accurate with respect to the real data?

While the terms ‘sufficient accuracy’ and ’accurate’ are not quantitative, we can impose that our meaning of this is that the relative L^2-errors can be 10^{-5} at most. The fifth sub-question is a sanity-check to see if solving problems of this size is doable at all, before we continue with real data.

1.3. Structure

This thesis starts by providing some background information on the medical details in Chapter 2, where also a description of the data is given. The mathematical model is then introduced in Chapter 3, as well as its discretisation in a numerical setting. The first sub-problem concerning the linear-elastic part of this model is tackled in Chapter 4 with both methodology and results given. Chapter 5 then handles the sub-problem of the evolution of the model over time, also describing methods and findings. The last sub-problem of finding the right parameters is described and analysed in Chapter 6, followed by Chapter 7 which combines the best found methods thus far and tests them on real data. The findings of this work are then summarised in Chapter 8, which concludes this thesis along with some recommendations.

2
Medical Context

In this chapter we elaborate on the necessary concepts and knowledge from medicine to fully understand this work. In Section 2.1 it is explained what type of cancer is considered, with some of its properties. After that the current standard treatment is given in Section 2.2, with an explanation of the pre-processing of MRI data. The data that we will be using is explained in Section 2.3.

2.1. Breast Cancer

We differentiate over 18 different types of breast cancer, which are most commonly located in the *ductal* an *lobular* part of the breast [6, 7]. Ductal carcinomas are formed in the cells lining the milk ducts, while lobular ones are made in the glands that produce the milk. One can subdivide the breast cancer types by the receptor status [6], which is either hormone receptor positive (HR+), HER2 positive (HER2+), or triple negative. A HR+ tumour has receptors for progesterone, estrogen or both, and may depend on the hormone for growth. The HER2+ tumours on the other hand have the 'human epidermal growth factor receptor 2' protein. This receptor is found on all breast cells, and it promotes growth. A lack of these three type of receptors means that the tumour is classified as triple negative. The focus of this work is on HER2+ tumours, which is a rapidly growing [8] and aggressive type of cancer with relatively high mortality rate [9, 10]. However, they tend to respond quite favourably to treatment, and they entail about 20-25% of all breast cancer cases.

As patients' breast tissue is composed of adipose, fibroglandular and tumour tissue, all with different mechanical properties [11–13], growth of the tumour is not straightforward. The stress and deformation caused by the presence of the tumour impedes it expansion, based on the stiffness of surrounding tissue; this is called *mass effect*. It was found that tumour cells seem to proliferate logistically, rather than exponentially which seems intuitive for doubling cells [14]. This means that smaller groups of malignant tumour cells grow faster than larger ones, and that one can not talk about a doubling time in this context.

2.2. Treatment and MRI Scans

Once a diagnosis of cancer has been made, patients are treated with either chemotherapy, radiotherapy, surgery or a combination of these. If chemotherapy is administrated prior to surgery to shrink the tumour for facilitated removal, it is called *neoadjuvant chemotherapy* (NAT). The success of NAT can prevent the need of a mastectomy, i.e. the removal of the breast entirely, and reduce the surgery to a less invasive lumpectomy, where just the tumour is removed [11, 15]. Current NAT treatments at Erasmus MC and other hospitals are not patient-specific but generalised, so no adjustments are made for different responses of patients. The complexity of the growth and response, along with the increased rate of recurrence free-survival of succesfull NAT treatment [16], suggests that optimisation of the treatment schedule can be very lucrative. Success of NAT can be measured with *pathological complete response* (pCR) of the patient, which is the absence of residual tumour cells in sampled breast tissue; this is called *histological analysis*.

Knowing if NAT is effective early on is pivotal to the treatment, as a lack of response can then be tackled by changing treatment strategy or administered substances. One can evaluate this with radiological scans, such as the ones from MRI or ultrasound. We refer to the lack of evident cancerous cells in these radiological scans as the *radiological complete response* (rCR) [17]. In the case of HER2+ patients, rCR seems to correspond quite well to pCR [18]. Newer types of MRI can provide insights earlier in the treatment schedule than the conventional one. For this work *diffusion-weighted MRI* (DW-MRI) and *dynamic contrast-enhanced MRI* (DCE-MRI) were used. The first of these can be used to measure the *apparent diffusion coefficient* (ADC, i.e. the rate of water diffusion in the tissue), and the second can grant clearer contrast between different tissues after inserting a *contrast agent*.

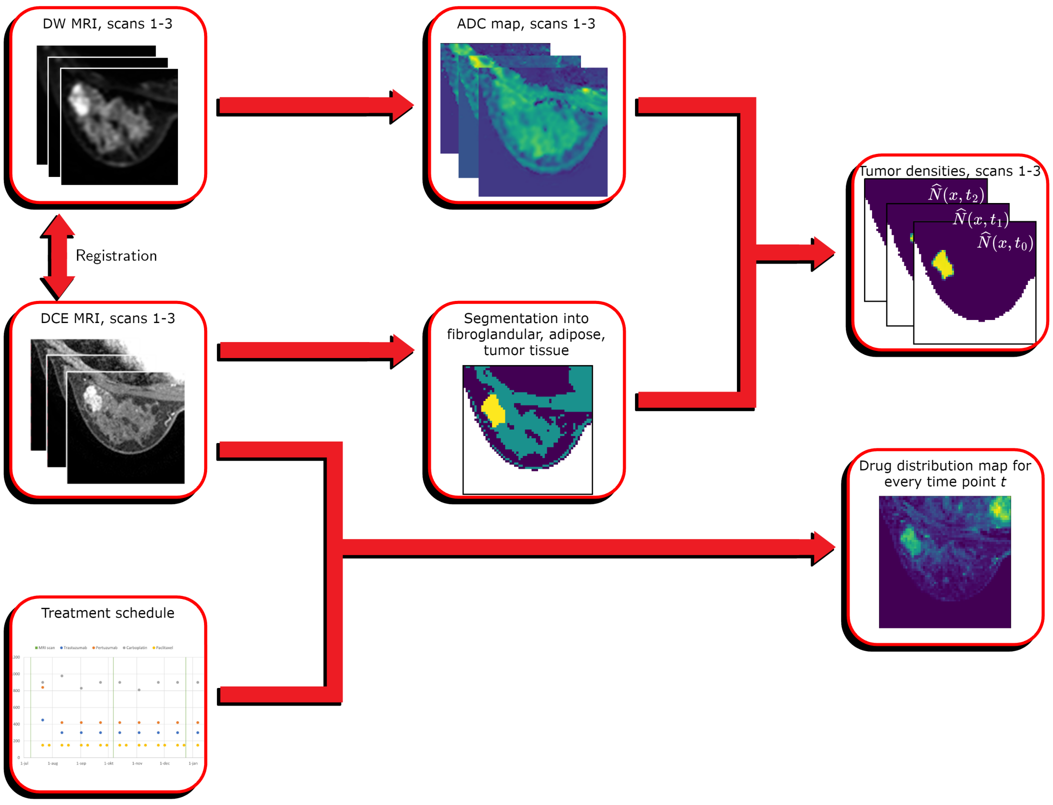

Figure 2.1: Pre-processing pipeline of the MRI data to obtain the tumour densities per voxel and the chemotherapy agent concentration. MRI scans within this figure are taken from [1].

Full details on the data acquisition and pre-processing for this project can be read in [1, 5], and a small overview is given here. For a number of given scans of a patient, first the images (of different sessions and techniques) need to be aligned with each other; this is called *registration*. The different ways of doing this are *rigid registration*, which uses translation and rotation, and *non-rigid registration*, which uses stretching. This step is done with the ITK-ELASTIX python package [19]. Second, the tumour tissue is identified and segmented from the surrounding tissue. For the preceding works this was done with help of a radiologist, namely dr A.I.M Obedijn from Erasmus MC.

After this the remaining tissue is segmented in adipose and fibroglandular tissue, by applying global histogram equalization. Third, the tumour densities are determined using the ADC values and the segmentation. The lower the ADC value of a voxel, the more cells it contains most likely. Lastly, the concentration of the chemotherapy agent is calculated with the DCE-MRI scans and the treatment schedule. A schematic of this process is given in Figure 2.1. With this one can start the calibration and evaluation of the mathematical model.

2.3. Input Data

For three patients we have DCE-MRI and DW-MRI scans at times t_0, t_1, t_2, which are pre-processed as described in the previous Section. The first scan at t_0 is made before the treatment, the second at t_1 during the treatment and the last at t_2 after the treatment, with slightly varying amounts of weeks between the scans between the patients. The first two scans will be used to calibrate the model, while the last is used to evaluate the predictive power. This calibration is done by using the model dynamics to predict what the tumour looks like at t_1, given the information of the tumour at t_0 and some patient-specific parameters. The parameters are then adjusted using information of the error at t_1, until the prediction at t_1 is sufficiently good or a set amount of time has passed. After that the predicted value at t_2 is compared to the real value to evaluate the workings of the model. The three patients have been put forward by dr A. Jager, the involved internist-oncologist from EMC. None of these three patients achieved complete pCR in their treatment. More details on these scans can be found in Appendix A.1.

At the start of this project, these three sets of scans weren't available yet, and hence some simulated data is used in this work as well. To generate this simulated data we either need a publicly usable MRI scan of a breast cancer tumour, or we need to generate it. This could be done with e.g. a simple Gaussian bell shaped tumour in a cube-shaped domain and / or cube-shaped 'breast'. Using this starting image which we set to be at t_0, we commit 'the inverse crime' by running the model forward in time for some known patient-specific parameters to generate the images at t_1 and t_2. This eliminates the model error from the equation, but for the sake of testing the speed of the numerical inversion methods this does not matter. It actually has benefits to use this simulated data, as it gives insight if the model is even solvable in realistic time for simple simulated problems. If it is, then a relatively bad accuracy on real data suggests that the computational methods are likely not at fault, but that the model might not be describing reality as accurate as expected. In this case, modelling assumptions should be revised for further research.

We need to discretise the temporal and spatial domain in order to evaluate our model numerically. The temporal discretisation depends on the method; for explicit methods the stability of the solution depends on the time-step. The spatial discretisation could be done at the resolution of the MRI, though if this is not feasible it might need to be down-scaled. In the previous two works this was done with factor 3 for a single slice. The result of the discretisation is that we have all subdomains of the breast described by distinct voxels $\bar{x}$. We continue to the next Chapter with details on this discretisation and the model itself.

3
Mathematical Model

We introduce the mathematical model in this Chapter, which is to be implemented as efficiently as possible. Specifically, this model is called the *Drug-Included Mechanically Coupled Reaction-Diffusion model* (DI-MCRD). To explain the model fully, we first introduce the regular *Mechanically Coupled Reaction-Diffusion model* (MCRD) in Section 3.1. After that the inclusion of interaction with administered drugs is elaborated on in Section 3.2. We continue with a small note on calibration of this model in Section 3.3, and we finalise this Chapter with a detailed description of the discretisation of this model in Section 3.4.

3.1. MCRD Model

The current most accurate biomechanical model for predicting breast tumour response is the *Mechanically Coupled Reaction-Diffusion model* (MCRD), as opposed to regular reaction-diffusion models. This was shown in detail in [11]. The standard MCRD model is described by the following coupled set of partial differential equations:

$$\frac{\partial N(\bar{x},t)}{\partial t} = \nabla\cdot\big(D(\bar{x},t)\nabla N(\bar{x},t)\big) + k(\bar{x})N(\bar{x},t)\Big(1-\frac{N(\bar{x},t)}{\theta}\Big) \tag{3.1}$$

$$D(\bar{x},t) = D_0 e^{-\gamma\sigma_{vm}(\bar{x},t)} \tag{3.2}$$

$$\nabla\cdot G(\bar{x})\nabla\vec{u}(\bar{x},t) + \nabla\Big(\frac{G(\bar{x})}{1-2v}(\nabla\cdot\vec{u}(\bar{x},t))\Big) - \lambda\nabla N(\bar{x},t) = 0 \tag{3.3}$$

Equation 3.1 here describes the rate of change of the number of tumour cells $N(\bar{x},t)$ in a voxel $\bar{x}$ within the domain at time t (d). The first term after the equality sign is the diffusion term and represents the random tumour cell movement, with tumour cell diffusion $D(\bar{x},t)$. The second term describes the logistic growth in the form of a reaction term, with $k(\bar{x})$ the net tumour proliferation rate within $\bar{x}$ and θ the carrying capacity of the cells, i.e. the maximum amount of tumour cells within a voxel. One can also consider a voxel-dependent carrying capacity by changing θ to $\theta(\bar{x})$, though this parameter would need additional tuning. The boundary conditions here on $N(\bar{x},t)$ are homogeneous Neumann, which means that there is no diffusive flux of tumour cells on the boundary of the domain, as well as on the boundary of the breast. The initial condition $N(\bar{x},0)$ corresponds to the data of the first MRI scan.

The diffusion of the tumour cells is described by equation 3.2, where we use D_0 (mm^3/d) to denote the diffusion without external stress, σ_{vm} (kPa) to denote the von Mises stress and γ (kPa^{-1}) as a coupling constant for it. Von Mises stress expresses the interaction between the tumour changing in size and its surroundings; if there is no stress present the diffusion is equal to D_0, and if the stress is non-zero the diffusion is lower. We calculate the non-negative von Mises stress using Equation 3.3, where $\vec{u} = (u, v, w)$ (mm) are the local deformations in the $x-$, $y-$ and $z-$direction, respectively. One can calculate the normal and shear strain on the tissue with this deformation, which for small displacements is equal to:

$$\begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{pmatrix} = \begin{pmatrix} \partial u/\partial x \\ \partial v/\partial y \\ \partial w/\partial z \\ \partial u/\partial y \\ \partial u/\partial z \\ \partial v/\partial z \end{pmatrix} \tag{3.4}$$

The first three terms are the normal strains, and the last three terms the shear strains. With Hooke's law for linear isotropic materials we can then find the normal and shear stresses as:

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{pmatrix} = \frac{2G(\bar{x})}{1-2\nu} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-2\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-2\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-2\nu \end{bmatrix} \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{pmatrix} \tag{3.5}$$

Here $G(\bar{x}) = \frac{E(\bar{x})}{2(1+\nu)}$ (kPa) is the shear modulus, i.e. an intrinsic mechanical property of the tissue. The multiplication with the factor before the matrix should be done such that the voxels of $G(\bar{x})$ and ε correspond, naturally. This shear modulus is different for the three different types of tissue, and hence it is spatially dependent. For this we use the Young's modulus $E(\bar{x})$ (kPa) and Poisson's ratio ν, which will be taken from literature values. The von Mises stress is then calculated at each voxel as follows:

$$\sigma_{vm} = \sqrt{\frac{1}{2}\left((\sigma_{xx}-\sigma_{yy})^2 + (\sigma_{xx}-\sigma_{zz})^2 + (\sigma_{zz}-\sigma_{yy})^2 + 6(\sigma_{xy}^2+\sigma_{xz}^2+\sigma_{yz}^2)\right)} \tag{3.6}$$

Equation 3.3 is derived from the mechanical equilibrium $\nabla \cdot \sigma_{vm}(\bar{x}, t) - \lambda \nabla N(\bar{x}, t) = 0$ after expressing it in the displacement $\vec{u}(\bar{x}, t)$. We need to assume here that the breast tissue to be linear elastic isotropic. The λ used in the equation is a coupling constant for the displacement vector. Furthermore, there are homogeneous Dirichlet boundary conditions imposed here on $\vec{u}(\bar{x}, t)$; this ensures that there is no tissue displacement on the boundaries.

Evaluating the system forward in time from a starting point can be broken down into six steps for one time-step. Given an initial value $N(\bar{x}, t)$, we first compute its gradient $\nabla N(\bar{x}, t)$ numerically with e.g. central differences. Second, we use this gradient in Equation 3.3 to solve it for $\vec{u}(\bar{x}, t)$ in each voxel. From this we calculate the von Mises stress $\sigma_{vm}(\bar{x}, t)$ at each voxel in the third step. Fourth, we update the value of the tumour cell diffusion $D(\bar{x}, t)$ in each voxel with Equation 3.2. The fifth step is to calculate $N(\bar{x}, t+\Delta t)$ using Equation 3.1 and the chosen temporal integration method, and the sixth step is updating the time parameter to $t := t + \Delta t$. After this the steps are repeated until a desired ending time. A schematic of this process is given in Figure 3.1.

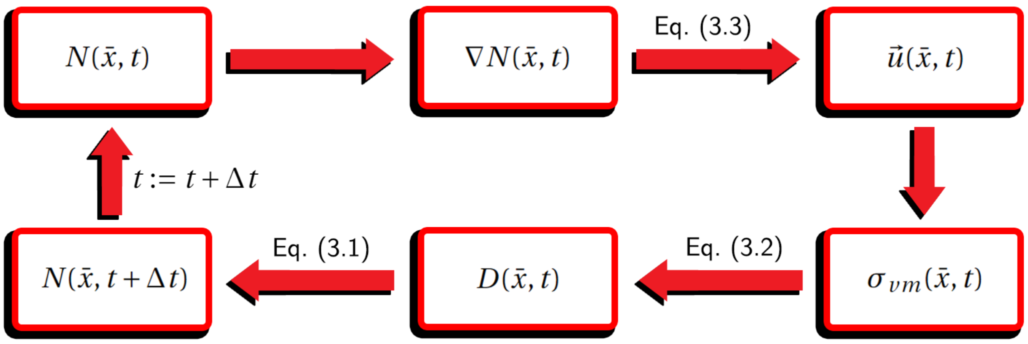

Figure 3.1: The six steps taken when doing a single time-step while evaluating the model.

3.2. Drug Inclusion

After the apparent success of the MCRD model, extensions were made to include chemotherapy and other administrated substances, leading to *Drug-Included MCRD models* (DI-MCRD). In the drug-included model, the reaction diffusion equation 3.1 is changed to include a term reflecting proliferation due to chemotherapy. This looks as follows:

$$\frac{\partial N(\bar{x},t)}{\partial t} = \nabla\cdot\big(D(\bar{x},t)\nabla N(\bar{x},t)\big) + k(\bar{x})N(\bar{x},t)\left(1-\frac{N(\bar{x},t)}{\theta}\right) - \alpha C^{drug}_{tissue}(\bar{x},t)N(\bar{x},t) \tag{3.7}$$

The additional parameter α $((\mu M\cdot d)^{-1})$, called the *drug efficacy* needs to be tuned here as well. The drug concentration $C^{drug}_{tissue}(\bar{x},t)$ (μM) at location $\bar{x}$ and time t here can be defined in multiple ways, and it can be calculated from the DCE-MRI scans. The two ways considered in the previous works are the *extended Kety-Tofts model* (KT) [16, 20] and the *Normalised Blood Volume Map method* (NBVM) [21], which are respectively given by:

$$C^{drug}_{tissue}(\bar{x},t) = K^{trans}(\bar{x})\int_0^t \left(C^{drug}_{plasma}(s)\exp\left(-\frac{K^{trans}(\bar{x})}{v_e(\bar{x})}(t-s)\right)\right)ds + v_p(\bar{x})C^{drug}_{plasma}(t) \tag{3.8}$$

$$C^{drug}_{tissue}(\bar{x},t) = C^{drug}_{tissue}(\bar{x},t^*)e^{-\beta(t-t^*)} \tag{3.9}$$

For the KT model it is assumed that the chemotherapy spreads in the same way as the contrast agent in the DCE-MRI scan. It describes the change in drug concentration between the (breast) tissue and the blood plasma. Hence we have $K^{trans}(\bar{x})$ (d^{-1}) as the volume transfer constant of the contrast agent from the plasma to the *tissue extravascular extracellular space* between the tissue cells and blood vessels. The admitted contrast agent in this context is Gadolinium. Volume fractions of the extravascular extracellular space and the plasma space are respectively given by $v_e(\bar{x})$ and $v_p(\bar{x})$. Note that the ratio $\frac{K^{trans}(\bar{x})}{v_e(\bar{x})}$ then represents the reflux rate of the contrast agent. Lastly, $C^{drug}_{plasma}(t)$ (μM) is a literature-based function for the considered drug.

The NBVM is also calculated from the DCE-MRI scans, but now it subtracts the average pre-contrast signal values from the post-contrast signal values. Of this graph the *area under the curve* (AUC) is calculated, which we normalise with its maximum found value; this is taken as the starting concentration at the time of administering it, and is denoted as $C^{drug}_{tissue}(\bar{x},t^*)$. It is then assumed that the concentration decays exponentially, for which a proper value of the drug decay rate β (d^{-1}) needs to be found. Optimising this value in combination with α might be difficult due to their codependency.

3.3. Calibration

To predict the patient-specific response, we need to calibrate the model to fit the real data as well as possible. The parameters we that we are surely tuning are the values of the proliferation rate $k(\bar{x})$ within our domain, the diffusion parameter D_0 and the efficacy of the drug concentration against the tumour cells α. This way all three driving factors for growth and shrinkage are tuned. Additional parameters that might be tuned, depending on the results, are the carrying capacity θ (possibly varying over the domain), the coupling constant for the von Mises stress γ, the material property ν, the coupling constant for the displacement vector λ and the parameters from the drug inclusion. The optimisation parameters are denoted altogether in the remainder as the vector **P**.

The actual number of tumour cells according to the real or simulated data is denoted as $N_{data}(t)$, though this might only be available at select values of t. We denote the prediction from our model of the amount of tumour cells at time t, given parameters **P**, as $N_{model}(t;\mathbf{P})$. This value is dependent on an initial value $N_{data}(t_0)$ at time t_0. Using this notation, the optimisation objective of finding the best parameter combination such that the L^2−norm at a certain time t_1 is minimised is

$$\mathbf{P}^* = \arg\min_{\mathbf{P}} ||N_{data}(t_1) - N_{model}(t_1;\mathbf{P})||_2^2 \tag{3.10}$$

The norm of the error considers the whole domain here, which will be discretised in the next Section. This problem is underdetermined in this formulation, but in Chapter 6 we explore how we can overcome this challenge, as well as how to solve this inverse problem in general. If there is ample scans per patient, one can minimise the sum of errors at multiple time-points $t_1, t_2, \ldots$ instead. There should be at least one scan per patient which is not considered in the calibration of the model; with this we can validate the predictive performance of the fitted model.

3.4. Discretization

To implement and solve the model we need to discretise the partial differential equations. In this Section the spatial and temporal discretisations used in the model are given for a equidistant grid, as commonly used in Finite Differences. The formulae used and some of the notation convention are taken from [22, 23]. Methods in subsequent chapters which do not use this scheme will be elaborated on as they are introduced later. The schemes are given for 3D systems, though some test might need to be done on 2D systems; the schemes for those are derived analogously and have analogous matrix properties such as symmetry and positive definiteness.

3.4.1. Reaction-diffusion

We start with the reaction diffusion equation as given in 3.7. For this $\bar{x}$ is used to denote a certain cell-centred voxel of the 3D spatial discretisation, taken from a $n_x \times n_y \times n_z$ MRI image with the centres of the pixels as the points. The boundary of the domain is hence where the coordinates are either 0 or $n_i - 1$. We define the total number of points as $n = n_x \cdot n_y \cdot n_z$. Using only derivatives to spatial coordinates and no nabla operators, this equation can be written as

$$\begin{aligned}\frac{\partial N(\bar{x},t)}{\partial t} =& \frac{\partial}{\partial x}\left(D(\bar{x},t)\frac{\partial N(\bar{x},t)}{\partial x}\right) + \frac{\partial}{\partial y}\left(D(\bar{x},t)\frac{\partial N(\bar{x},t)}{\partial y}\right) + \frac{\partial}{\partial z}\left(D(\bar{x},t)\frac{\partial N(\bar{x},t)}{\partial z}\right)\\ &+ k(\bar{x})N(\bar{x},t)\left(1 - \frac{N(\bar{x},t)}{\theta}\right) - \alpha C_{tissue}^{drug}(\bar{x},t)\end{aligned}$$

For convenience, we introduce a shorter notation of the spatially and temporally dependent variables. The voxels are orderly numbered with $\bar{x} := (x_i, y_j, z_k)$ and the time is labelled $t_m := m\Delta t$, with step sizes $\Delta x, \Delta y, \Delta z$ and Δt respectively. With this we can rewrite $N(\bar{x}, t_m) =: N_{i,j,k}^m$, as well as

$D(\bar{x}, t_m) =: D^m_{i,j,k}$, $k(\bar{x}) =: k_{i,j,k}$ and $C^{drug}_{tissue}(\bar{x}, t_m) =: C^m_{i,j,k}$. To avoid confusion, note that subscripts k always refer to the spatial position on the $z-$axis and that a regular k refers to tumour proliferation. The partial spatial derivatives are approximated numerically using central differences, where midpoints are used inbetween the voxels. For a given variable $M^m_{i,j,k}$, the $x-$derivative is then approximated by

$$\frac{\partial M^m_{i,j,k}}{\partial x} = \frac{M^m_{i+1/2,j,k} - M^m_{i-1/2,j,k}}{\Delta x} + \mathcal{O}(\Delta x^2) \tag{3.11}$$

The big-$\mathcal{O}$ notation is used here to show the order of the error of the approximation, and $y-$ and $z-$derivatives are analogous to the $x-$derivative. Using this approximation for the second order derivative yields

$$\frac{\partial}{\partial x}\left(D^m_{i,j,k}\frac{\partial N^m_{i,j,k}}{\partial x}\right) = \frac{D^m_{i+1/2,j,k}\frac{\partial N^m_{i+1/2,j,k}}{\partial x} - D^m_{i-1/2,j,k}\frac{\partial N^m_{i-1/2,j,k}}{\partial x}}{\Delta x} + \mathcal{O}(\Delta x^2)$$

There is still a differential operator present in this form, as well as two midpoints that need to be approximated. A midpoint $M^m_{i+1/2,j,k}$ can be approximated with the two enclosing points, and for its $x-$derivative we use central differences again, now using regular grid points. This looks as follows

$$M^m_{i+1/2,j,k} \approx \frac{M^m_{i+1,j,k} + M^m_{i,j,k}}{2}$$
$$\frac{\partial M^m_{i+1/2,j,k}}{\partial x} = \frac{M^m_{i+1,j,k} - M^m_{i,j,k}}{\Delta x} + \mathcal{O}(\Delta x^2)$$

Combining the previous three expressions gives the following approximation of the diffusion term in the $x-$direction

$$\frac{\partial}{\partial x}\left(D^m_{i,j,k}\frac{\partial N^m_{i,j,k}}{\partial x}\right) \approx \frac{1}{2\Delta x^2}\Big(\big(D^m_{i-1,j,k} + D^m_{i,j,k}\big)N^m_{i-1,j,k} - \big(D^m_{i-1,j,k} + 2D^m_{i,j,k} + D^m_{i+1,j,k}\big)N^m_{i,j,k} + \big(D^m_{i,j,k} + D^m_{i+1,j,k}\big)N^m_{i+1,j,k}\Big)$$

The order of the local truncation error here is also $\mathcal{O}(\Delta x^2)$. The full discretisation of the right-hand side of equation 3.7 is then given by

$$\begin{aligned}\frac{\partial N^m_{i,j,k}}{\partial t} \approx\ & \frac{1}{2\Delta x^2}\Big(\big(D^m_{i-1,j,k} + D^m_{i,j,k}\big)N^m_{i-1,j,k} - \big(D^m_{i-1,j,k} + 2D^m_{i,j,k} + D^m_{i+1,j,k}\big)N^m_{i,j,k} + \big(D^m_{i,j,k} + D^m_{i+1,j,k}\big)N^m_{i+1,j,k}\Big)\\ &+ \frac{1}{2\Delta y^2}\Big(\big(D^m_{i,j-1,k} + D^m_{i,j,k}\big)N^m_{i,j-1,k} - \big(D^m_{i,j-1,k} + 2D^m_{i,j,k} + D^m_{i,j+1,k}\big)N^m_{i,j,k} + \big(D^m_{i,j,k} + D^m_{i,j+1,k}\big)N^m_{i,j+1,k}\Big)\\ &+ \frac{1}{2\Delta z^2}\Big(\big(D^m_{i,j,k-1} + D^m_{i,j,k}\big)N^m_{i,j,k-1} - \big(D^m_{i,j,k-1} + 2D^m_{i,j,k} + D^m_{i,j,k+1}\big)N^m_{i,j,k} + \big(D^m_{i,j,k} + D^m_{i,j,k+1}\big)N^m_{i,j,k+1}\Big)\\ &+ k_{i,j,k}N^m_{i,j,k}\left(1 - \frac{N^m_{i,j,k}}{\theta}\right) - \alpha C^m_{i,j,k}\end{aligned} \tag{3.12}$$

Note that this can only hold for interior points of the data, and that we have homogeneous Neumann conditions at the boundary of the breast. One can discretise the system such that the boundary of the breast is also the boundary of the domain. However, a simpler way is to discretise the MRI data as-is into a bar and put the boundary conditions on 'inner' points, which will result in a lot of zeroes in the matrices and vectors. With central differences, we can then impose the boundaries at $1/2$ and $n_x - 1/2$ since we have a cell-centred domain as follows:

$$\frac{\partial N^m_{1/2,j,k}}{\partial x} = \frac{N^m_{1,j,k} - N^m_{0,j,k}}{\Delta x} + \mathcal{O}(\Delta x^2) = 0 \quad \Longrightarrow \quad N^m_{0,j,k} = N^m_{1,j,k} + \mathcal{O}(\Delta x^3)$$
$$\frac{\partial N^m_{n_x-1/2,j,k}}{\partial x} = \frac{N^m_{n_x,j,k} - N^m_{n_x-1,j,k}}{\Delta x} + \mathcal{O}(\Delta x^2) = 0 \quad \Longrightarrow \quad N^m_{n_x,j,k} = N^m_{n_x-1,j,k} + \mathcal{O}(\Delta x^3)$$

As we have third order error here and we divide by Δx^2 later on, we achieve first order error on the boundary. The boundary conditions in the $y-$ and $z-$direction are analogous. We can also impose that $D^m_{i,j,k} = 0$ for values outside of the domain, as there is no diffusion there. With this the diffusion terms at the boundaries in the $x-$direction (and analogously the $y-$ and $z-$direction) can be approximated as follows

$$\frac{\partial}{\partial x}\Big(D^m_{0,j,k}\frac{\partial N^m_{0,j,k}}{\partial x}\Big) \approx \frac{1}{2\Delta x^2}\Big(-\big(D^m_{0,j,k}+D^m_{1,j,k}\big)N^m_{0,j,k} + \big(D^m_{0,j,k}+D^m_{1,j,k}\big)N^m_{1,j,k}\Big)$$
$$\frac{\partial}{\partial x}\Big(D^m_{n_x-1,j,k}\frac{\partial N^m_{n_x-1,j,k}}{\partial x}\Big) \approx \frac{1}{2\Delta x^2}\Big(\big(D^m_{n_x-2,j,k}+D^m_{n_x-1,j,k}\big)N^m_{n_x-2,j,k} - \big(D^m_{n_x-2,j,k}+D^m_{n_x-1,j,k}\big)N^m_{n_x-1,j,k}\Big)$$

If we conglomerate all the individual elements $N^m_{i,j,k}$ into a vector $\mathbf{N}^m$ (ordered lexicographically first by x, then y, then z), as well as for $k_{i,j,k}$ and $C^m_{i,j,k}$ in respectively $\mathbf{k}$ and $\mathbf{C}^m$, we can write equation 3.12 into an elegant matrix-vector product form as follows:

$$\frac{\partial \mathbf{N}^m}{\partial t} = A\mathbf{N}^m + f(\mathbf{N}^m) \tag{3.13}$$

$$\begin{aligned} A = \; & \frac{1}{2\Delta x^2}\big(\mathbf{D}\odot(T_{n_x}\otimes I_{n_y}\otimes I_{n_z}) + \mathbf{D}_{x+}\odot(U_{n_x}\otimes I_{n_y}\otimes I_{n_z}) + \mathbf{D}_{x-}\odot(L_{n_x}\otimes I_{n_y}\otimes I_{n_z})\big) \\ & + \frac{1}{2\Delta y^2}\big(\mathbf{D}\odot(I_{n_x}\otimes T_{n_y}\otimes I_{n_z}) + \mathbf{D}_{y+}\odot(I_{n_x}\otimes U_{n_y}\otimes I_{n_z}) + \mathbf{D}_{y-}\odot(I_{n_x}\otimes L_{n_y}\otimes I_{n_z})\big) \\ & + \frac{1}{2\Delta z^2}\big(\mathbf{D}\odot(I_{n_x}\otimes I_{n_y}\otimes T_{n_z}) + \mathbf{D}_{z+}\odot(I_{n_x}\otimes I_{n_y}\otimes U_{n_z}) + \mathbf{D}_{z-}\odot(I_{n_x}\otimes I_{n_y}\otimes L_{n_z})\big) \end{aligned}$$

$$\begin{aligned} T_{n_i} &= \text{tridiag}\big(1,-2,1\big) \in \mathbb{R}^{n_i\times n_i} \\ U_{n_i} &= \text{tridiag}\big(0,-1,1\big) \in \mathbb{R}^{n_i\times n_i} \\ L_{n_i} &= \text{tridiag}\big(1,-1,0\big) \in \mathbb{R}^{n_i\times n_i} \\ I_{n_i} &= \text{diag}\big(1,\ldots,1\big) \in \mathbb{R}^{n_i} \end{aligned}$$

$$f(\mathbf{N}^m) = \mathbf{k}\cdot\mathbf{N}^m\cdot\Big(1-\frac{\mathbf{N}^m}{\theta}\Big) - \alpha\mathbf{C}^m\cdot\mathbf{N}^m + \partial\mathbf{A}$$

$$\partial\mathbf{A} = \begin{cases} D_{0,j,k}N^m_{0,j,k} & i=0 \\ D_{n_x-1,j,k}N^m_{n_x-1,j,k} & i=n_x-1 \\ D_{i,0,k}N^m_{i,0,k} & j=0 \\ D_{i,n_y-1,k}N^m_{i,n_y-1,k} & j=n_y-1 \\ D_{i,j,0}N^m_{i,j,0} & k=0 \\ D_{i,j,n_z-1}N^m_{i,j,n_z-1} & k=n_z-1 \\ 0 & \text{otherwise} \end{cases}$$

We use A here as the discretisation matrix of the diffusion part, and $f(\mathbf{N}^m)$ as the nonlinear reaction part, the drug inclusion and an additional term for the homogeneous Neumann boundary conditions $\partial\mathbf{A}$. The matrix A can be decomposed into a sum of nine kronecker products of three terms, splitting the diffusion into the $x-$, $y-$ and $z-$direction. The term in the kronecker products corresponding to the direction it operates on consists of a tridiagonal part, a upper triangular part and a lower triangular part, denoted by T, U, L respectively. These matrices also represent second order central differences, forward differences and (negative) backward differences. Then, $\mathbf{D}_{x\pm}$ is the vector $\mathbf{D}$ with the $x-$indices shifted forward / backward by one, and $\mathbf{D}_{y\pm}, \mathbf{D}_{z\pm}$ are defined similarly. In this notation we use $\mathbf{X} \odot Y$ to indicate that all the rows of the matrix Y are multiplied with the respective elements of the vector $\mathbf{X}$. This is the same as multiplying Y from the left with a matrix which has $\mathbf{X}$ on the diagonal and zeroes elsewhere. The additional term for the boundary conditions is needed if A is expressed in this formulation, as otherwise the boundary points are defined in the same way as the regular points. This decomposition could become handy later on for multiplications with preconditioners, due to the multiplicative properties of kronecker products. Furthermore, the symmetry of the matrix A immediately follows from this formulation, as well as its negative definiteness.

3.4.2. Linear-elastic

Next we discretise the linear elastic equation, using central differences again as formulated in equation 3.11. We want to write equation 3.3 in a usual way for the Lamé equation, for which we rewrite the differential operators again. The result is the following system of three equations

$$
\begin{aligned}
&\frac{2-2\nu}{1-2\nu}\frac{\partial}{\partial x}\Big(G(\bar{x})\frac{\partial u(\bar{x},t)}{\partial x}\Big)+\frac{\partial}{\partial y}\Big(G(\bar{x})\frac{\partial u(\bar{x},t)}{\partial y}\Big)+\frac{\partial}{\partial z}\Big(G(\bar{x})\frac{\partial u(\bar{x},t)}{\partial z}\Big)\\
&\quad+\frac{2\nu}{1-2\nu}\frac{\partial}{\partial x}\Big(G(\bar{x})\Big(\frac{\partial v(\bar{x},t)}{\partial y}\Big)\Big)+\frac{\partial}{\partial y}\Big(G(\bar{x})\Big(\frac{\partial v(\bar{x},t)}{\partial x}\Big)\Big)\\
&\quad+\frac{2\nu}{1-2\nu}\frac{\partial}{\partial x}\Big(G(\bar{x})\Big(\frac{\partial w(\bar{x},t)}{\partial z}\Big)\Big)+\frac{\partial}{\partial z}\Big(G(\bar{x})\Big(\frac{\partial w(\bar{x},t)}{\partial x}\Big)\Big)=\lambda\frac{\partial N(\bar{x},t)}{\partial x}\\
&\frac{\partial}{\partial x}\Big(G(\bar{x})\frac{\partial v(\bar{x},t)}{\partial x}\Big)+\frac{2-2\nu}{1-2\nu}\frac{\partial}{\partial y}\Big(G(\bar{x})\frac{\partial v(\bar{x},t)}{\partial y}\Big)+\frac{\partial}{\partial z}\Big(G(\bar{x})\frac{\partial v(\bar{x},t)}{\partial z}\Big)\\
&\quad+\frac{2\nu}{1-2\nu}\frac{\partial}{\partial y}\Big(G(\bar{x})\Big(\frac{\partial u(\bar{x},t)}{\partial x}\Big)\Big)+\frac{\partial}{\partial x}\Big(G(\bar{x})\Big(\frac{\partial u(\bar{x},t)}{\partial y}\Big)\Big)\\
&\quad+\frac{2\nu}{1-2\nu}\frac{\partial}{\partial y}\Big(G(\bar{x})\Big(\frac{\partial w(\bar{x},t)}{\partial z}\Big)\Big)+\frac{\partial}{\partial z}\Big(G(\bar{x})\Big(\frac{\partial w(\bar{x},t)}{\partial y}\Big)\Big)=\lambda\frac{\partial N(\bar{x},t)}{\partial y}\\
&\frac{\partial}{\partial x}\Big(G(\bar{x})\frac{\partial w(\bar{x},t)}{\partial x}\Big)+\frac{\partial}{\partial y}\Big(G(\bar{x})\frac{\partial w(\bar{x},t)}{\partial y}\Big)+\frac{2-2\nu}{1-2\nu}\frac{\partial}{\partial z}\Big(G(\bar{x})\frac{\partial w(\bar{x},t)}{\partial z}\Big)\\
&\quad+\frac{2\nu}{1-2\nu}\frac{\partial}{\partial z}\Big(G(\bar{x})\Big(\frac{\partial u(\bar{x},t)}{\partial x}\Big)\Big)+\frac{\partial}{\partial x}\Big(G(\bar{x})\Big(\frac{\partial u(\bar{x},t)}{\partial z}\Big)\Big)\\
&\quad+\frac{2\nu}{1-2\nu}\frac{\partial}{\partial z}\Big(G(\bar{x})\Big(\frac{\partial v(\bar{x},t)}{\partial y}\Big)\Big)+\frac{\partial}{\partial y}\Big(G(\bar{x})\Big(\frac{\partial v(\bar{x},t)}{\partial z}\Big)\Big)=\lambda\frac{\partial N(\bar{x},t)}{\partial z}
\end{aligned}
$$

We split the displacement vector here into its $x-$, $y-$ and $z-$ components as $\vec{u} =: (u, v, w)$ respectively. The Lamé coefficients of this equation expressed in the original variables are $\mu(\bar{x}) = G(\bar{x})$, $\lambda(\bar{x}) = \frac{2\nu}{1-2\nu}$ and $\rho(\bar{x}) = 2\mu(\bar{x}) + \lambda(\bar{x}) = \frac{2-2\nu}{1-2\nu}$. As in the reaction-diffusion equation, we look at how we can discretise the individual terms of the equations. We will use the short notation $u(\bar{x}, t_m) =: u^m_{i,j,k}$, $G(\bar{x}) =: G_{i,j,k}$, regular central difference, and central difference as the average of the forward and backward differences for the derivatives in two directions. For this we use δ_x and $\bar{\delta}_x$ as the forward and backward differential operator with respect to x, respectively. This way of determining the derivative is also second order accurate. Second derivatives in the same direction can be written in the same way as in the reaction-diffusion equation, namely

$$\frac{\partial}{\partial x}\Big(G_{i,j,k}\frac{\partial u^m_{i,j,k}}{\partial x}\Big) \approx \frac{1}{2\Delta x^2}\Big(\big(G_{i-1,j,k}+G_{i,j,k}\big)u^m_{i-1,j,k} - \big(G_{i-1,j,k}+2G_{i,j,k}+G_{i+1,j,k}\big)u^m_{i,j,k} + \big(G_{i,j,k}+G_{i+1,j,k}\big)u^m_{i+1,j,k}\Big)$$

When using a second derivative in two directions, we use the average of the forward and backward differences as follows.

$$\begin{aligned}
\frac{\partial}{\partial x}\Big(G_{i,j,k}\frac{\partial v^m_{i,j,k}}{\partial y}\Big) &= \frac{1}{2}\Big(\partial_x\big(G_{i,j,k}\bar{\partial}_y v^m_{i,j,k}\big) + \bar{\partial}_x\big(G_{i,j,k}\partial_y v^m_{i,j,k}\big)\Big)\\
&\approx \frac{1}{2}\Big(\partial_x\Big(G_{i,j,k}\frac{v^m_{i,j,k}-v^m_{i,j-1,k}}{\Delta y}\Big) + \bar{\partial}_x\Big(G_{i,j,k}\frac{v^m_{i,j+1,k}-v^m_{i,j,k}}{\Delta y}\Big)\Big)\\
&\approx \frac{1}{2\Delta x\Delta y}\Big(G_{i-1,j,k}\big(v^m_{i-1,j,k}-v^m_{i-1,j+1,k}\big) + G_{i,j,k}\big(v^m_{i,j-1,k}-2v^m_{i,j,k}+v^m_{i,j+1,k}\big)\\
&\qquad + G_{i+1,j,k}\big(-v^m_{i+1,j-1,k}+v^m_{i+1,j,k}\big)\Big)
\end{aligned}$$

Likewise we can write out all the other derivatives in the first equation, as well as in the second and third equation; for this one simply needs to replace indices, directions and derivatives appropriately. This time we impose homogeneous Dirichlet boundary conditions on the boundary of the breast, which means that there is no tissue displacement there. For a cell-centred grid these conditions are hard to impose, and hence we use a *staggered grid.* The sub-problem of linear elasticity will instead be solved on a vertex-centred grid, containing $n_x - 1 \times n_y - 1 \times n_z - 1$ points. The right-hand side depends on the spatial derivative of N, which for a cell-centred N can be approximated with central differences on a vertex-centred domain. Furthermore, for notation we use a mask function $\partial(\cdot,\cdot,\cdot) : \mathbb{N}^3 \mapsto \{0,1\}$ which indicates if a voxel is on the boundary

$$u^m_{i,j,k} = v^m_{i,j,k} = w^m_{i,j,k} = 0 \quad \text{if} \quad \partial(i,j,z) = 1$$

With these discretisations we can put the system of equations into a form with matrix vector multiplications again. We put the individual discretised elements of u, v, w, G into vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{G}$ again, and the resulting notation is as follows.

$$\begin{aligned}
B_{11}\mathbf{u} + B_{12}\mathbf{v} + B_{13}\mathbf{w} &= g_1(\mathbf{N_m})\\
B_{21}\mathbf{u} + B_{22}\mathbf{v} + B_{23}\mathbf{w} &= g_2(\mathbf{N_m})\\
B_{31}\mathbf{u} + B_{32}\mathbf{v} + B_{33}\mathbf{w} &= g_3(\mathbf{N_m})
\end{aligned} \tag{3.14}$$

$$
\begin{aligned}
B_{11} = \frac{2-2\nu}{1-2\nu}\frac{1}{2\Delta x^2}\big(&\mathbf{G}\odot(T_{n_x}\otimes I_{n_y}\otimes I_{n_z}) + \mathbf{G}_{x+}\odot(U_{n_x}\otimes I_{n_y}\otimes I_{n_z}) + \mathbf{G}_{x-}\odot(L_{n_x}\otimes I_{n_y}\otimes I_{n_z})\big)\\
+\ \frac{1}{2\Delta y^2}\big(&\mathbf{G}\odot(I_{n_x}\otimes T_{n_y}\otimes I_{n_z}) + \mathbf{G}_{y+}\odot(I_{n_x}\otimes U_{n_y}\otimes I_{n_z}) + \mathbf{G}_{y-}\odot(I_{n_x}\otimes L_{n_y}\otimes I_{n_z})\big)\\
+\ \frac{1}{2\Delta z^2}\big(&\mathbf{G}\odot(I_{n_x}\otimes I_{n_y}\otimes T_{n_z}) + \mathbf{G}_{z+}\odot(I_{n_x}\otimes I_{n_y}\otimes U_{n_z}) + \mathbf{G}_{z-}\odot(I_{n_x}\otimes I_{n_y}\otimes L_{n_z})\big)\\
B_{22} = \frac{1}{2\Delta x^2}\big(&\mathbf{G}\odot(T_{n_x}\otimes I_{n_y}\otimes I_{n_z}) + \mathbf{G}_{x+}\odot(U_{n_x}\otimes I_{n_y}\otimes I_{n_z}) + \mathbf{G}_{x-}\odot(L_{n_x}\otimes I_{n_y}\otimes I_{n_z})\big)\\
+\ \frac{2-2\nu}{1-2\nu}\frac{1}{2\Delta y^2}\big(&\mathbf{G}\odot(I_{n_x}\otimes T_{n_y}\otimes I_{n_z}) + \mathbf{G}_{y+}\odot(I_{n_x}\otimes U_{n_y}\otimes I_{n_z}) + \mathbf{G}_{y-}\odot(I_{n_x}\otimes L_{n_y}\otimes I_{n_z})\big)\\
+\ \frac{1}{2\Delta z^2}\big(&\mathbf{G}\odot(I_{n_x}\otimes I_{n_y}\otimes T_{n_z}) + \mathbf{G}_{z+}\odot(I_{n_x}\otimes I_{n_y}\otimes U_{n_z}) + \mathbf{G}_{z-}\odot(I_{n_x}\otimes I_{n_y}\otimes L_{n_z})\big)\\
B_{33} = \frac{1}{2\Delta x^2}\big(&\mathbf{G}\odot(T_{n_x}\otimes I_{n_y}\otimes I_{n_z}) + \mathbf{G}_{x+}\odot(U_{n_x}\otimes I_{n_y}\otimes I_{n_z}) + \mathbf{G}_{x-}\odot(L_{n_x}\otimes I_{n_y}\otimes I_{n_z})\big)\\
+\ \frac{1}{2\Delta y^2}\big(&\mathbf{G}\odot(I_{n_x}\otimes T_{n_y}\otimes I_{n_z}) + \mathbf{G}_{y+}\odot(I_{n_x}\otimes U_{n_y}\otimes I_{n_z}) + \mathbf{G}_{y-}\odot(I_{n_x}\otimes L_{n_y}\otimes I_{n_z})\big)\\
+\ \frac{2-2\nu}{1-2\nu}\frac{1}{2\Delta z^2}\big(&\mathbf{G}\odot(I_{n_x}\otimes I_{n_y}\otimes T_{n_z}) + \mathbf{G}_{z+}\odot(I_{n_x}\otimes I_{n_y}\otimes U_{n_z}) + \mathbf{G}_{z-}\odot(I_{n_x}\otimes I_{n_y}\otimes L_{n_z})\big)
\end{aligned}
$$

$$
\begin{aligned}
B_{12} = \frac{1}{2\Delta x\Delta y}\frac{2}{1-2\nu}\big(&\mathbf{G}\odot(I_{n_x}\otimes T_{n_y}\otimes I_{n_z}) - \mathbf{G}_{x+}\odot(P_{n_x}\otimes L_{n_y}\otimes I_{n_z}) - \mathbf{G}_{x-}\odot(M_{n_x}\otimes U_{n_y}\otimes I_{n_z})\big)\\
+\ \frac{1}{2\Delta x\Delta y}\big(&\mathbf{G}\odot(T_{n_x}\otimes I_{n_y}\otimes I_{n_z}) - \mathbf{G}_{y+}\odot(L_{n_x}\otimes P_{n_y}\otimes I_{n_z}) - \mathbf{G}_{y-}\odot(U_{n_x}\otimes M_{n_y}\otimes I_{n_z})\big)\\
B_{21} = \frac{1}{2\Delta x\Delta y}\frac{2}{1-2\nu}\big(&\mathbf{G}\odot(T_{n_x}\otimes I_{n_y}\otimes I_{n_z}) - \mathbf{G}_{y+}\odot(L_{n_x}\otimes P_{n_y}\otimes I_{n_z}) - \mathbf{G}_{y-}\odot(U_{n_x}\otimes M_{n_y}\otimes I_{n_z})\big)\\
+\ \frac{1}{2\Delta x\Delta y}\big(&\mathbf{G}\odot(I_{n_x}\otimes T_{n_y}\otimes I_{n_z}) - \mathbf{G}_{x+}\odot(P_{n_x}\otimes L_{n_y}\otimes I_{n_z}) - \mathbf{G}_{x-}\odot(M_{n_x}\otimes U_{n_y}\otimes I_{n_z})\big)
\end{aligned}
$$

$$
\begin{aligned}
B_{13} = \frac{1}{2\Delta x\Delta z}\frac{2}{1-2\nu}\big(&\mathbf{G}\odot(I_{n_x}\otimes I_{n_y}\otimes T_{n_z}) - \mathbf{G}_{x+}\odot(P_{n_x}\otimes I_{n_y}\otimes L_{n_z}) - \mathbf{G}_{x-}\odot(M_{n_x}\otimes I_{n_y}\otimes U_{n_z})\big)\\
+\ \frac{1}{2\Delta x\Delta z}\big(&\mathbf{G}\odot(T_{n_x}\otimes I_{n_y}\otimes I_{n_z}) - \mathbf{G}_{z+}\odot(L_{n_x}\otimes I_{n_y}\otimes P_{n_z}) - \mathbf{G}_{z-}\odot(U_{n_x}\otimes I_{n_y}\otimes M_{n_z})\big)\\
B_{31} = \frac{1}{2\Delta x\Delta z}\frac{2}{1-2\nu}\big(&\mathbf{G}\odot(T_{n_x}\otimes I_{n_y}\otimes I_{n_z}) - \mathbf{G}_{z+}\odot(L_{n_x}\otimes I_{n_y}\otimes P_{n_z}) - \mathbf{G}_{z-}\odot(U_{n_x}\otimes I_{n_y}\otimes M_{n_z})\big)\\
+\ \frac{1}{2\Delta x\Delta z}\big(&\mathbf{G}\odot(T_{n_x}\otimes I_{n_y}\otimes I_{n_z}) - \mathbf{G}_{y+}\odot(L_{n_x}\otimes P_{n_y}\otimes I_{n_z}) - \mathbf{G}_{y-}\odot(U_{n_x}\otimes M_{n_y}\otimes I_{n_z})\big)
\end{aligned}
$$

$$
\begin{aligned}
B_{23} = \frac{1}{2\Delta y\Delta z}\frac{2}{1-2\nu}\big(&\mathbf{G}\odot(I_{n_x}\otimes I_{n_y}\otimes T_{n_z}) - \mathbf{G}_{y+}\odot(I_{n_x}\otimes P_{n_y}\otimes L_{n_z}) - \mathbf{G}_{y-}\odot(I_{n_x}\otimes M_{n_y}\otimes U_{n_z})\big)\\
+\ \frac{1}{2\Delta y\Delta z}\big(&\mathbf{G}\odot(I_{n_x}\otimes T_{n_y}\otimes I_{n_z}) - \mathbf{G}_{z+}\odot(I_{n_x}\otimes L_{n_y}\otimes P_{n_z}) - \mathbf{G}_{z-}\odot(I_{n_x}\otimes U_{n_y}\otimes M_{n_z})\big)\\
B_{23} = \frac{1}{2\Delta y\Delta z}\frac{2}{1-2\nu}\big(&\mathbf{G}\odot(I_{n_x}\otimes T_{n_y}\otimes I_{n_z}) - \mathbf{G}_{z+}\odot(I_{n_x}\otimes L_{n_y}\otimes P_{n_z}) - \mathbf{G}_{z-}\odot(I_{n_x}\otimes U_{n_y}\otimes M_{n_z})\big)\\
+\ \frac{1}{2\Delta y\Delta z}\big(&\mathbf{G}\odot(I_{n_x}\otimes I_{n_y}\otimes T_{n_z}) - \mathbf{G}_{y+}\odot(I_{n_x}\otimes P_{n_y}\otimes L_{n_z}) - \mathbf{G}_{y-}\odot(I_{n_x}\otimes M_{n_y}\otimes U_{n_z})\big)
\end{aligned}
$$

$$
\begin{aligned}
g_1(\mathbf{N}^m) &= \lambda\mathscr{D}_x\mathbf{N}^m + \partial\mathbf{B}_1\\
g_2(\mathbf{N}^m) &= \lambda\mathscr{D}_y\mathbf{N}^m + \partial\mathbf{B}_2\\
g_3(\mathbf{N}^m) &= \lambda\mathscr{D}_z\mathbf{N}^m + \partial\mathbf{B}_3
\end{aligned}
$$

$$P_{n_i} = \text{tridiag}\big(0,0,1\big) \in \mathbb{R}^{n_i-1\times n_i-1}$$
$$M_{n_i} = \text{tridiag}\big(1,0,0\big) \in \mathbb{R}^{n_i-1\times n_i-1}$$

$$\partial\mathbf{B}_1 = \Big(\mathbf{u} - \lambda\mathscr{D}_x\mathbf{N}^m + B_{11}\mathbf{u} + B_{12}\mathbf{v} + B_{13}\mathbf{w}\Big)\cdot\mathbb{1}_{\{\partial(i,j,z)=1\}}$$
$$\partial\mathbf{B}_2 = \Big(\mathbf{v} - \lambda\mathscr{D}_y\mathbf{N}^m + B_{11}\mathbf{u} + B_{12}\mathbf{v} + B_{13}\mathbf{w}\Big)\cdot\mathbb{1}_{\{\partial(i,j,z)=1\}}$$
$$\partial\mathbf{B}_2 = \Big(\mathbf{w} - \lambda\mathscr{D}_z\mathbf{N}^m + B_{11}\mathbf{u} + B_{12}\mathbf{v} + B_{13}\mathbf{w}\Big)\cdot\mathbb{1}_{\{\partial(i,j,z)=1\}}$$

In this notation the subscripts n_i actually mean that the matrix has size $n_i - 1$ due to the staggered grid, but the -1 is left out for readability. The $B-$matrices describe the discretisations of the derivatives, and they can be decomposed into sums of Kronecker products as well, which is also done for the reason of designing preconditioners. The matrices P and M are used to apply finite differences in multiple directions, and they represent a forward shift and a backward shift. Note that T, U, L, P, M are linear combinations of each other, but by defining them separately the notations become a bit more readable. The vector-valued g's describe the N-term and correct for the boundary conditions with the terms $\partial\mathbf{B}$. For this we define the operator $\mathscr{D}_i$, which differentiates the vector $\mathbf{N}$ in direction i on the vertex-centred grid. This process involves central differences and taking the average of four surrounding values. We can indeed see that $\partial\mathbf{B_1}$ enforces that $\mathbf{u}$ is 0 at the boundaries, as it cancels out all other elements in those respective rows, and the same for $\partial\mathbf{B}_2$, v and $\partial\mathbf{B}_3$, w. With the Kronecker product formulation the symmetry and negative definiteness of B immediately follows; more details of this can be found in [24]. Note that in a numerical implementation the 'mask' should also be applied on the rows, such that the symmetry of B is preserved. If this is not done, there will be non-zero elements in the columns of B which represent a zero variable, while their transposed counterparts are zero. Furthermore, we can improve the condition number of B by multiplying $\vec{u}$ on the boundary rows with the mean of the matrix B_{11}. This prevents the system from having eigenvalue -1 for each boundary cell. Finally, we combine the system of equations 3.14 back into one equation which we note down as

$$B\vec{u} = g(\mathbf{N}^m) \tag{3.15}$$

$$B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}, \quad g(\mathbf{N}^m) = \begin{bmatrix} g_1(\mathbf{N}^m) \\ g_2(\mathbf{N}^m) \\ g_3(\mathbf{N}^m) \end{bmatrix}$$

The calculated displacements are used to obtain the shear strain, for which the displacements need to be differentiated. As the Von Mises stress needs to be known on the cell-centred grid, and therefore the shear strain as well, it is logical to use central differences again. This leaves us with a $n_x - 2 \times n_y - 2 \times n_z - 2$ grid of shear strains, which we extrapolate on all sides to find its value for all cells. The drop of accuracy on the cell beyond the boundary does not matter too much, as the diffusion is zero there anyways.

In [24] it is proven that this matrix B is symmetric in 2D; the symmetry in 3D follows from the formulation above, but no formal proof is given for this work. They also show that this matrix is spectrally equivalent to the Laplacian operator, with bounds provided by the extrema of $\mathbf{G}$. One can already see that if $\mathbf{G} = \frac{1}{1-2\nu} = 1$ the diagonal blocks of B are identical to the Laplacian operator. In the next Chapter, we delve into the methods we can use to solve the linear-elastic equation to solve it as quickly as possible.

4 Linear-Elastic Equation

In this Chapter we tackle the linear-elastic sub-problem, which needs to be solved in each time-step of the model. For this we first introduce all the methods we want to test in detail in Section 4.1. After that we analyse the speed of convergence in Section 4.2, on which we will base a conclusion of what method should be used in further analyses.

4.1. Methods

The equation we want to solve is Equation 3.15, i.e. $B\vec{u}^m = g(\mathbf{N}^m)$, with the sparse symmetric negative definite matrix B of size $3\underline{n} \times 3\underline{n}$, where $\underline{n} = (n_x - 1) \cdot (n_y - 1) \cdot (n_z - 1)$.

4.1.1. Direct Method

The first and most trivial way of solving is the *Direct* method. This method was used in the previous works, and uses Gaussian elimination on the LU-decomposition of the matrix B.

$$B = LU$$

Here L is a lower diagonal matrix and U an upper diagonal matrix. As B is SPD, the matrices always exist, and we even have $U = L^T$ for some L. We can find these matrices with more Gaussian elimination, which can be done efficiently with e.g. SCIPY. In this symmetric case we call it the Cholesky decomposition instead. After decomposing the matrix, we can perform Gaussian elimination twice to find the solution $\vec{u}$ to the problem. For non-sparse B, the complexity of this method is $\mathcal{O}(\underline{n}^3)$, so its simplicity does come with general high computational costs. For sparse B, the cost is reduced but likely still rather high.

4.1.2. Conjugate Gradients

Instead of this direct method, we can use the *Conjugate Gradient* (CG) method, which is a Krylov method that makes use of conjugate directions. The method, denoted as in [25], can be written in an algorithmic way as follows

Algorithm 1 Conjugate Gradient Method

```
 1: $\vec{u}_0 := 0, \quad \mathbf{r}_0 := g(\mathbf{N}^m) - B\vec{u}_0$
 2: $\mathbf{p}_0 := \mathbf{r}_0$
 3: $k := 0$
 4: **While** stopping criterion not met **do**
 5:     $\alpha_k = \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top (B\mathbf{p}_k)}$
 6:     $\vec{u}_{k+1} = \vec{u}_k + \alpha_k \mathbf{p}_k$
 7:     $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k B\mathbf{p}_k$
 8:     $\beta_k = \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}$
 9:     $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
10:     $k := k+1$
11: **end While**
12: **Return** $\vec{u}_k$
```

The vectors $\mathbf{p}_k$ and $\mathbf{r}_k$ represent the search direction and the residual, respectively. In each iteration the vector $\vec{u}_k$ lies in the Krylov subspace $\mathcal{K}_m(B, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, B\mathbf{r}_0, \ldots, B^{m-1}\mathbf{r}_0\}$, which means that the method converges in a maximum of n iterations. As a stopping criterion it is usual to impose that $||r_k|| \leq \varepsilon ||\mathbf{r}_0||$ for a certain norm and a certain threshold ε. We will take the most standard L_2-norm and a threshold of $\varepsilon = 10^{-5}$ in this work. One can also see that the starting residual for $\vec{u}_0 = 0$ is $\mathbf{r}_0 = g(\mathbf{N}^m)$, but the notation in line 1 of the algorithm is more general. The reason that $B\mathbf{p}_k$ is written between brackets in line 5 is that it should be computed separately, as it is also used in line 7.

The CG method can be sped up by *preconditioning* the system with a *preconditioner M* which is also SPD. This is called the *Preconditioned Conjugate Gradient* (PCG) method. The convergence of the CG method depends on the condition number of B; if the condition number of $P^{-1}BP^{-\top}$ for $M = PP^\top$ is lower, the convergence for solving the modified system is faster. The resulting algorithm can be written in a way where only one system needs to be solved per iteration

Algorithm 2 Preconditioned Conjugate Gradient Method

```
    $\vec{u}_0 := 0, \quad \mathbf{r}_0 := g(\mathbf{N}^m) - B\vec{u}_0$
 2: $\mathbf{z}_0 = M^{-1}\mathbf{r}_0$
    $\mathbf{p}_0 := \mathbf{r}_0$
 4: $k := 0$
    **While** stopping criterion not met **do**
 6:     $\alpha_k = \frac{\mathbf{r}_k^\top \mathbf{z}_k}{\mathbf{p}_k^\top (B\mathbf{p}_k)}$
        $\vec{u}_{k+1} = \vec{u}_k + \alpha_k \mathbf{p}_k$
 8:     $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k B\mathbf{p}_k$
        $\mathbf{z}_{k+1} = M^{-1}\mathbf{r}_{k+1}$
10:     $\beta_k = \frac{\mathbf{r}_{k+1}^\top \mathbf{z}_{k+1}}{\mathbf{r}_k^\top \mathbf{z}_k}$
        $\mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta_k \mathbf{p}_k$
12:     $k := k+1$
    **end While**
14: **Return** $\vec{u}_k$
```

Solving the system in line 9 does not need to be difficult, if the preconditioner is chosen with care. The more M resembles B, the faster the convergence of PCG is per iteration, but the more time it might take to solve the system in each iteration. By taking M to be the diagonal of B, the inverse of M is straightforward, and there is also immediately an improvement to the condition number. This method will be denoted as CG(d).

4.1.3. Laplacian Preconditioning

In Section 3.4 we saw that if we take G to be one everywhere, the diagonal blocks of B are equal to the Laplacian operator Δ; this is the idea presented in [24]. They present an efficient way of solving this system with $M = \operatorname{diag}(\Delta, \Delta)$ in 2D, and we extend it in this work to 3D. First, we write the system to be solved as

$$\begin{bmatrix} \Delta & 0 & 0 \\ 0 & \Delta & 0 \\ 0 & 0 & \Delta \end{bmatrix} \begin{bmatrix} \mathbf{w}^{(1)} \\ \mathbf{w}^{(2)} \\ \mathbf{w}^{(3)} \end{bmatrix} = \begin{bmatrix} \mathbf{z}^{(1)} \\ \mathbf{z}^{(2)} \\ \mathbf{z}^{(3)} \end{bmatrix}$$

$$\Delta = \frac{1}{h^2}\left(T_{n_x} \otimes I_{n_y} \otimes I_{n_z} + I_{n_x} \otimes T_{n_y} \otimes I_{n_z} + I_{n_x} \otimes I_{n_y} \otimes T_{n_z}\right)$$

We simplify here by assuming that we have $\Delta x = \Delta y = \Delta z = h$, which makes the notation more readable. This method also works for non-equal grid step sizes, however, which we will most likely have in practice. We multiply this system from the left with the following matrix

$$\mathscr{Z}_2 = \begin{bmatrix} Z_{n_x} \otimes Z_{n_y} \otimes I_{n_z} & 0 & 0 \\ 0 & Z_{n_x} \otimes Z_{n_y} \otimes I_{n_z} & 0 \\ 0 & 0 & Z_{n_x} \otimes Z_{n_y} \otimes I_{n_z} \end{bmatrix} \tag{4.1}$$

$$Z_{n_k} = \sqrt{\frac{2}{n_i - 1}} \left[\sin\left(\frac{ij\pi}{n_k - 1} \right) \right]_{i,j=0}^{n_k-2}$$

We have that Z_{n_i} is actually the discrete sine-transform (DST) matrix, with $Z_{n_i}^2 = I_{n_i}$ and $Z_{n_i} T_{n_i} Z_{n_i} = \Lambda_{n_i}$. This matrix Λ is diagonal as such

$$\Lambda_{n_i} = \operatorname{diag}\left(-\frac{4}{h^2} \sin^2\left(\frac{j\pi}{2(n_i-1)} \right) \right)_{j=0}^{n_i-2}$$

This property follows from the sine being an eigenfunction of the Laplacian operator. The result of this multiplication from the left is that we are left with a new system, which looks complicated but actually simplifies solving it

$$\begin{bmatrix} \mathscr{T} & 0 & 0 \\ 0 & \mathscr{T} & 0 \\ 0 & 0 & \mathscr{T} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{w}}^{(1)} \\ \tilde{\mathbf{w}}^{(2)} \\ \tilde{\mathbf{w}}^{(3)} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{z}}^{(1)} \\ \tilde{\mathbf{z}}^{(2)} \\ \tilde{\mathbf{z}}^{(3)} \end{bmatrix} \tag{4.2}$$

$$\mathscr{T} = \Lambda_{n_x} \otimes I_{n_y} \otimes I_{n_z} + I_{n_x} \otimes \Lambda_{n_y} \otimes I_{n_z} + \frac{1}{h^2} I_{n_x} \otimes I_{n_y} \otimes T_{n_z}$$

$$\tilde{\mathbf{w}}^{(i)} = \left(Z_{n_x} \otimes Z_{n_y} \otimes I_{n_z}\right)\mathbf{w}^{(i)}, \quad \tilde{\mathbf{z}}^{(i)} = \left(Z_{n_x} \otimes Z_{n_y} \otimes I_{n_z}\right)\mathbf{z}^{(i)}, \quad i = 1,2,3 \tag{4.3}$$

The system in 4.2 can be split into $3(n_x - 1)(n_y - 1)$ independent tridiagonal systems of order $n_z - 1$. These tridiagonal systems can all be solved in $\mathscr{O}(n_z)$ time, after which we need to return to the original variables using

$$\mathbf{w}^{(i)} = \left(Z_{n_x} \otimes Z_{n_y} \otimes I_{n_z}\right)\tilde{\mathbf{w}}^{(i)}, \quad \mathbf{z}^{(i)} = \left(Z_{n_x} \otimes Z_{n_y} \otimes I_{n_z}\right)\tilde{\mathbf{z}}^{(i)}, \quad i = 1,2,3 \tag{4.4}$$

The multiplications in equations 4.3 and 4.4 can be done efficiently using the *Fast Fourier Transform* (FFT) due to the nature of the matrices Z_{n_i}. We need to apply the FFT twice, once in the $x-$ and once in the $y-$ direction, both of which take $\mathcal{O}\big((n_i-1)^2\log(n_i-1)\big)$ time. This is only because the double Z_{n_i} transformations are separable, as $Z_{n_x}\otimes Z_{n_y}\otimes I_{n_z} = (Z_{n_x}\otimes I_{n_y}\otimes I_{n_z})(I_{n_x}\otimes Z_{n_y}\otimes I_{n_z})$.

Similarly, we can change the premultiplication factor in 4.1 to have the FFT also act upon the $z-$direction. We do this by changing the diagonal elements to $Z_{n_x}\otimes Z_{n_y}\otimes Z_{n_z}$, and denote this matrix as $\mathcal{Z}_3$. Instead of having to solve tridiagonal systems, we have a fully diagonal matrix multiplied with $\tilde{\mathbf{w}}$ with $\Lambda_{n_x}\otimes I_{n_y}\otimes I_{n_z} + I_{n_x}\otimes\Lambda_{n_y}\otimes I_{n_z} + I_{n_x}\otimes I_{n_y}\otimes\Lambda_{n_z}$ on its diagonal, so the inversion is trivial. These methods will be denoted as CG(Lap2) and CG(Lap3) respectively. While they solve the same system, the numerical cost of these implementations are different, and it remains to see which is faster.

Another way of abusing the diagonal similarity to the Laplacian operator is by constructing a LDL^T preconditioner, which entails the decomposition of B into a lower triangular matrix, a diagonal matrix and an upper triangular matrix. We define the matrices as follows:

$$B \approx M_L M_D^{-1} M_U = \begin{bmatrix} M_{11} & 0 & 0 \\ M_{21} & M_{22} & 0 \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \begin{bmatrix} M_{11} & 0 & 0 \\ 0 & M_{22} & 0 \\ 0 & 0 & M_{33} \end{bmatrix}^{-1} \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ 0 & M_{22} & M_{23} \\ 0 & 0 & M_{33} \end{bmatrix}$$

Here we will use $M_{11} = M_{22} = M_{33} = \Delta$, and the off-diagonal blocks represent the blocks of B with constant coefficients and no boundary conditions. Solving a system of the form $M_L M_D^{-1} M_U \mathbf{w} = \mathbf{z}$ is done in steps, starting by multiplying the equation from the right with $\mathcal{Z}_3$. We also note that $\mathcal{Z}_3\mathcal{Z}_3 = I_{3n}$. This leads to the following simple equations to solve sequentially

$$\mathcal{Z}_3 M_L M_D^{-1} M_U \mathbf{w} = \mathcal{Z}_3 M_L \mathcal{Z}_3\mathcal{Z}_3 M_D^{-1}\mathcal{Z}_3\mathcal{Z}_3 M_U \mathcal{Z}_3 \mathbf{w} = \mathcal{Z}_3\mathbf{z}$$

$$\begin{aligned} \implies \mathcal{Z}_3 M_L \mathcal{Z}_3 \dot{\mathbf{w}} &= \mathcal{Z}_3\mathbf{z} \\ \mathcal{Z}_3 M_D^{-1}\mathcal{Z}_3 \hat{\mathbf{w}} &= \dot{\mathbf{w}} \\ \mathcal{Z}_3 M_U \mathcal{Z}_3 \tilde{\mathbf{w}} &= \hat{\mathbf{w}} \\ \mathcal{Z}_3 \mathbf{w} &= \tilde{\mathbf{w}} \end{aligned}$$

The first and third of these sequential steps should not be solved by multiplying the matrices explicitly, as that is a computationally expensive task. Instead, one should take advantage of the FFT transform, the triangular structure and the fact that $Z_{n_i} T_{n_i} Z_{n_i} = I_{n_i}$. With this these can be solved with 5 FFT's in 3 directions each. The second step in this process is even easier, by noticing that $(\mathcal{Z}_3 M_D^{-1}\mathcal{Z}_3)^{-1}$ is a diagonal matrix matrix. Solving the full LDL^T system can thus be done in a total of 14 3D FFT's. This CG(LDLT) method needs more than double the amount of work per iteration than CG(Lap2) and CG(Lap3), but it should need fewer iterations.

A slight numerical improvement to the LDLT method can be made using the Eisenstat trick [26]. For this we need the off-diagonal blocks of M_L and M_U to be exactly equal to those of B. The premise of the method is that the computational cost of solving a LDLT system scales with twice the number of non-zero elements of B, but by writing it in a different way and using other search directions p and residuals r the cost scales only once with that number. We will denote this method as CG(Eis), but note that the true off-diagonal blocks can also be used in the LDLT method. It is also important to note that working with a different residual term makes the stopping criterion behave differently.

4.1.4. Recycling

We should acknowledge that this linear-elastic sub-problem is solved many times in a row, while only the right-hand side changes and the matrix B is constant. This is where the idea of recycling comes in, as explained nicely in [27]. When the first system $B\vec{u} = g(\mathbf{N}^0)$ is solved, we can use its Ritz vectors as the starting vectors to every following (P)CG solve. These Ritz vectors can be obtained using a Krylov basis

$$B\hat{R}_m = \hat{R}_{m+1}\bar{H}_m$$

Here $\hat{R}_m$ is a Krylov subspace and $\bar{H}_m$ a Hessenberg matrix, which is banded. Using the Ritz vector corresponding to the smallest eigenvalue b_0 as the starting value of CG, convergence should be established earlier than with a zero starting vector. One can use a smart way (viz Algorithm 3 from [27]) to obtain the Ritz vectors with these matrices, or by using the Power method. They also show that recycling using the last found solution vector u^{k-1} of the previous time-step as the starting vector is a simple yet effective way of speeding up convergence.

4.1.5. Multigrid

Lastly there are multigrid methods [25] to solve our problem, which have a very good theoretical convergence rate. The idea is to iteratively solve the equation with a recursive scheme on different discretisations of the domain. First we apply smoothing to reduce high frequency errors using one to three Gauss-Seidel iterations on the grid. After this we calculate the residual error, which is then restricted to a coarser grid. The residual on the coarse grid is then used in a recursive call to this same code, unless the pre-specified lowest level is reached; in this case we use a direct solver to solve for the residual. Once you go "up", i.e. you exit the recursive call, we smoothen the solution summed with the interpolated output of the recursion to find a better approximation. The amount of times this so-called "V-cycle" should be done to reach a relative error tolerance does not depend on the grid size, but the work per cycle scales with $\mathcal{O}(n)$. An algorithmic formulation of this method is shown below.

Algorithm 3 Geometric Multigrid method

```
1:  $\vec{u}_0 := 0, \quad \Delta x = \Delta y = \Delta z = h$
2:  def **V_cycle**$(u, f, h)$:
3:      $u$ =smooth$(u,\ f,\ h)$
4:      $r_h = f_h - B_h u$
5:      $r_{2h}$ =Restrict$(r_h)$
6:      **If not** smallest grid size:
7:          $du_{2h}$ =V_cycle$(0, r_{2h}, 2h)$
8:      **Else**:
9:          $du_{2h} = B_{2h}^{-1} r_{2h}$
10:     $du_h$ =Interpolate$(du_{2h})$
11:     $u$ =smooth$(u + du, f, h)$
12: **Return** $u$
13: **While** stopping criterion not met **do**
14:     $u$ =V_cycle$(u, g, h)$
15: **end While**
16: **Return** $\vec{u}$
```

The lines of this algorithmic formulation correspond to the text above. Restriction is done with the full weighting restriction operator, as explained in [25]. This uses a stencil with weights for neighbouring points. Given that the current grid has n_i points in a certain direction, the restricted grid will have $n_i/2$ points given that $n_i - 1$ is odd. In 1d the restriction matrix will look as follows:

$$R_h^{2h} = [r_{ij}] \in \mathbb{R}^{n_i/2\times(n_i-1)}, \quad r_{ij} = \begin{cases} 1/2 & j = 2i \\ 1/4 & j = 2i \pm 1 \\ 0 & \text{otherwise} \end{cases}$$

To extend this restriction to 3D we can simply take the Kronecker product of the terms, $\mathscr{R}_h^{2h} = R_h^{2h} \otimes R_h^{2h} \otimes R_h^{2h}$. We want to apply this transformation to each component of $\vec{u}$ separately, so the restriction matrix we will actually be working with is the block diagonal matrix $\mathfrak{R}_h^{2h} = \operatorname{diag}(\mathscr{R}_h^{2h}, \mathscr{R}_h^{2h}, \mathscr{R}_h^{2h})$. The interpolation matrix $\mathfrak{I}_{2h}^h$ is the transpose of this matrix multiplied with 8, as this makes sure the rows add up to one and the terms do not 'shrink' upon interpolation (i.e. the total number of cells remains constant). This does not necessarily hold on the boundary, but as we have zero Dirichlet conditions there we do not care. The system matrix B at every grid level also needs to be transformed properly, which is done as $B_{2h} = \mathfrak{R}_h^{2h} B_h \mathfrak{I}_{2h}^h$.

What we just described here is the Geometric Multigrid method (*GMG*), but there also is the Algebraic Multigrid method (*AMG*). This method uses more abstract elements based on the non-zero elements of the matrix B. There is a nice implementation of this method in python with the PYAMG package [28].

4.2. Convergence Analysis

To determine the best method to use, we check the speed of the methods in various settings. In all experiments the relative tolerance needed to break the iterative processes is 10^{-5} with respect to the norm of the right-hand side.

4.2.1. Simulated data assumptions

The setting we are experimenting on is a $200 \times 200 \times 200$ mm grid, with literature values $\nu = 0.45$ and $\lambda = 2.5e-3$. The cell carrying capacity is determined by assuming spherical tumour cells with a packing density of 0.7405 and a cell radius of $10\mu m$. This gives a tumour cell volume of $4189\mu m^3$, and if we divide the cell volume by this number we find the carrying capacity θ [11, 12, 20]. The value of the spatially dependent shear modulus $G(\bar{x})$ is taken to 2kPa everywhere, except for inside a bubbly domain defined by the mask $\{\sin 7\pi x) + \sin(7\pi y) + \sin(8\pi(z + 3/16)) > 1\}$ where its value is 4kPa, and 20kPa where the tumour is initially defined. The first two values represent fibroglandular and adipose tissue respectively, and these values are from [1, 5].

The tumour inside this domain is centred and round, with the number of tumour cells per voxel depending on a scaled radius $r = 3\sqrt{(x - 1/2)^2 + (y - 1/2)^2 + (z - 1/2)^2}$. The actual value is equal to $N(r) = \theta \exp\left(-3r^6\right) \cdot (1 + \delta)$ with $\delta \sim \mathcal{N}(0, 0.05)$ as long as $|r| < 1$. The values of the coordinates x, y, z in these functions are scaled to lie in between 0 and 1. Lastly, the boundary conditions are imposed only on the boundary of the cubic domain; this means that we are technically considering a cubic breast. Visual illustrations of these functions are shown in the figure below.

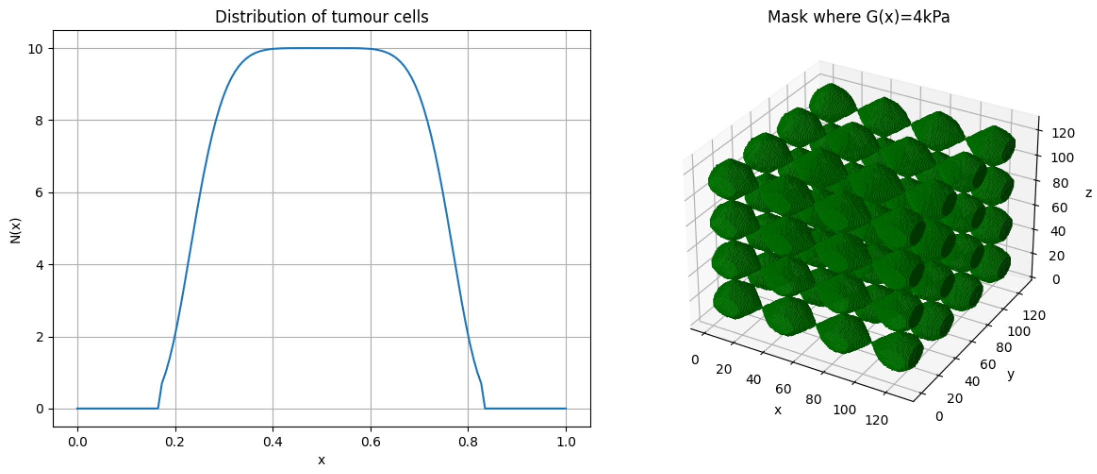

Figure 4.1: (a) 1D representation of the tumour cell distribution within the cubic domain with no noise and $\theta = 10$, (b) Voxel plot showing where the shear modulus is 4kPa; everywhere else it is 2kPa.

All the results in this section were acquired on a desktop computer with an AMD Ryzen 1200 Quad-Core Processor (3.7 GHz) and 16 GB RAM. Timings in subsequent Sections were also done on this computer, unless stated otherwise.

4.2.2. Varying grid size

First we want to check what happens to the amount of time and iterations needed, when we vary the amount of discretisation points. For this we take $n_x = n_y = n_z = n_i$, and we note that for full resolution we need around $n_i = 120$. The following two figures show the result of this test, averaged over 10 runs with small perturbations in the tumour.

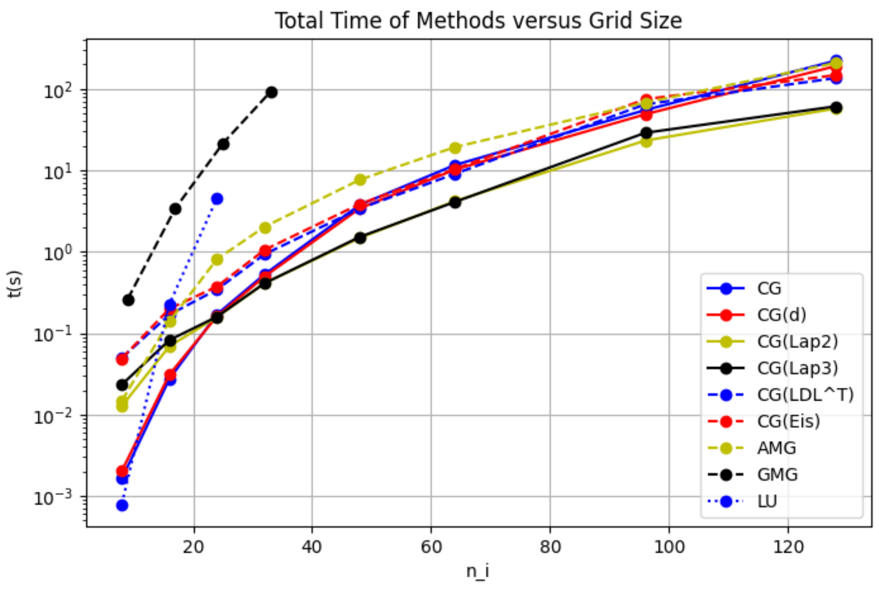

Figure 4.2: Time needed to solve the linear elastic equation versus the grid size for all methods.

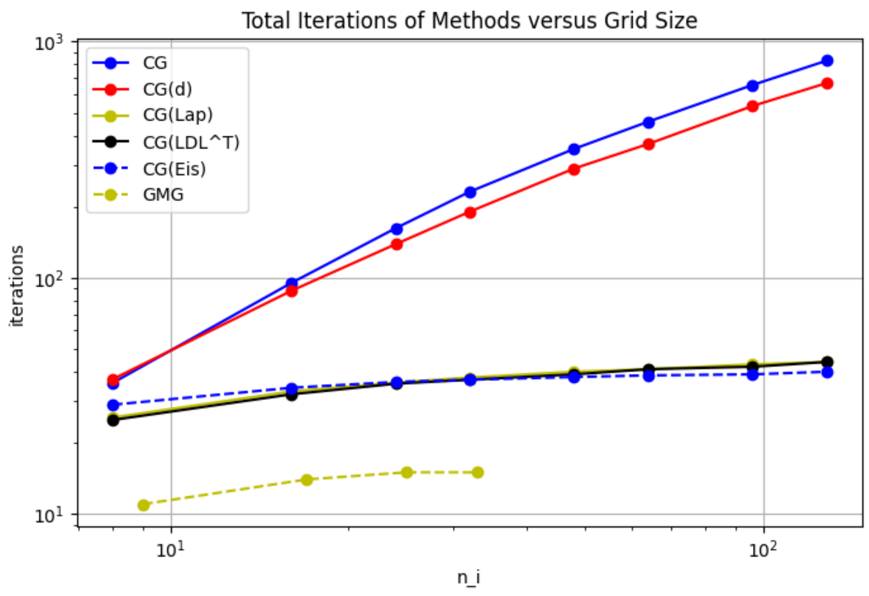

Figure 4.3: Iterations needed to solve the linear elastic equation versus the grid size for all methods.

The GMG and LU method were not evaluated for all grid sizes, as they already required large amounts of time for small grids. In Figure 4.3 the two Laplacian preconditioning methods are displayed in one line as they are essentially the same method with a different implementation, and hence they need the same amount of iterations. Furthermore, the GMG method was evaluated at different grid sizes, as the implementation used benefits from grids of size $n_i = 2^d + 2$ for $d \in \mathbb{N}$ while FFT methods benefit from $n_i = 2^d + 1$.

The first observation we make is that the Laplacian preconditioning method is the clear winner for large grid sizes. The method using two DST's and the Thomas algorithm seems to be slightly faster, especially when the grid size is not a power of 2 (which is beneficial for the FFT calculation method). The LDL^T method which contains more information of the system is slower, and actually takes the same amount of iterations to solve (the lines overlap, and hence the one of the Laplacian methods is not visible). Picking off-diagonal blocks of B with constant coefficients apparently does not make the preconditioner resemble the inverse of B more than the Laplacian preconditioner. The Eisenstadt method, with the actual off-diagonal blocks of B is slightly faster than the LDL^T method and takes less iterations. Diagonal or no preconditioning is not competitive in comparison to these other methods, as well as AMG. For some of the methods pre-work needs to be done, such as determining L and U for the LU method. We can subtract the time needed for this in our analysis, as it only needs to be done once. It was found that this does not influence the conclusions of this Section.

The GMG method has a relatively high run-time compared to the others. This mostly seems to originate from the Gauss-Seidel iterations, of which we do 2 in pre-smoothing and 1 post-smoothing during each V-cycle. One can instead use Jacobi iterations for smoothing, but as our matrix is not diagonally dominant there is no guarantee that this will converge. Faults in the implementation don't seem to be there, especially since the amount of iterations appears constant with respect to the grid size. We check the amount of iterations needed for larger grid sizes by running it once for the larger grid sizes, the result of this is shown in the next table.

n_i	9	17	25	33	49	65
no. iters	11	14	15	15	16	16

Table 4.1: Number of iterations needed for the GMG method for $n_i \times n_i \times n_i$ grids.

Indeed we find that the amount of iterations stays more or less constant, even for larger grid sizes. At $n_i = 65$ the GMG method took over 5 hours; clearly not feasible for our problem. Further improvements to the method could be made by implementing it in lower-level languages such as C or Fortran.

One last thing to check is the order of the increase in iterations as a function of the grid size, which can be established with the log-log plot in Figure 4.3. The slopes are determined by only using the last three data-points, as the slope has converged to a constant value by then. In the next table the slopes are given.

Table 4.2: Slopes of the time increase as a function of the grid size for all the methods.

Method	CG	CG(d)	CG(Lap)	CG(LDL$^\top$)	CG(Eis)	GMG
Slope	0.868	0.866	0.103	0.103	0.0920	0

We see that the slopes are rather low, so the amount of iterations needed does not increase that much with the grid size. The likely reason for this is that the large the grid size is, the sparser the right-hand side becomes. This is because the spatial derivative of N is only non-zero on the surface of a ball in our scenario, the fraction of which compared to the volume of a cube becomes smaller the larger the grid-size is.

4.2.3. Iterations and time

More insights on these iterative methods can be gained by checking the convergence per iteration and time, for various sizes of the grid. In the next figures we show the residual error $||Bu_k - g(\mathbf{N}||$ and the relative error $||u - u_k||$. The 'true' solution u is found by first running CG(Lap) on the problem with a relative error of 10^{-10} as the stopping criterion. First we check the output for $n_i = 16$.

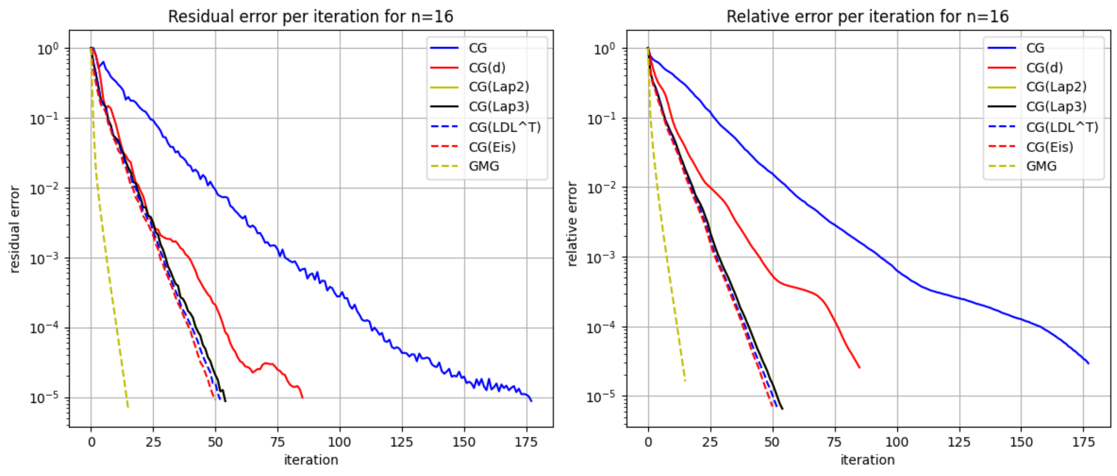

Figure 4.4: Residual and relative error per iteration for a grid with $n_i = 16$.

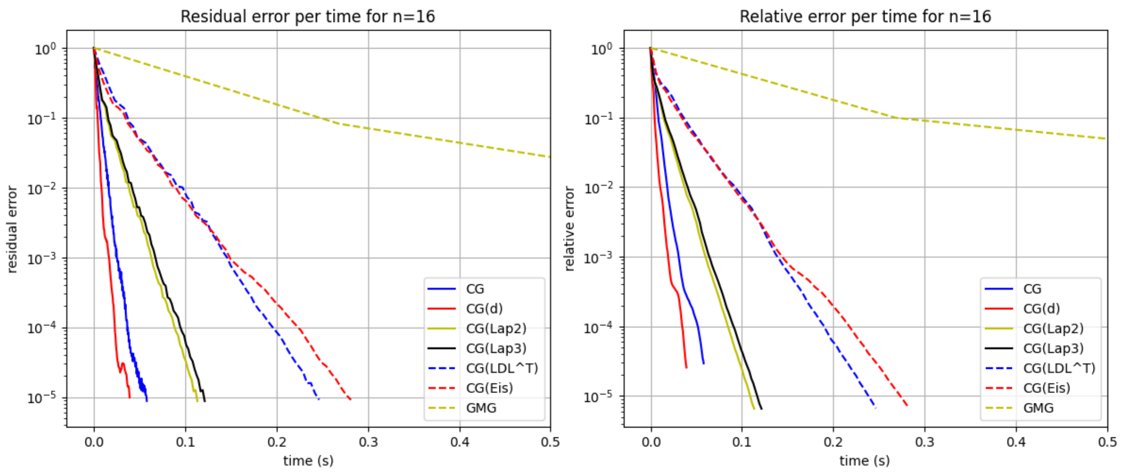

Figure 4.5: Residual and relative error per time for a grid with $n_i = 16$.

As the Eisenstadt method technically solves a transformed problem, we can't impose the exact same stopping criterion on it. Instead, it is solved up to a residual of order 10^{-6} in its own norm. The error per time figure was cropped such that the most interesting part is shown; the GMG method took around 3.5 seconds here. The errors per iteration are shown for both CG(Lap2) and CG(Lap3), but they overlap as they are different implementations of the same method.

The behaviour is more or less as we predicted, with the residual error decreasing exponentially per iteration for all methods. For these small systems the computational cost is low, so simple method as CG and CG(d) win here time-wise, even though they need a relatively high amount of iterations. The theoretically high convergence per iteration of GMG is clear here, but it does not reflect in the time needed. Next we check the same figures for $n_i = 32$.

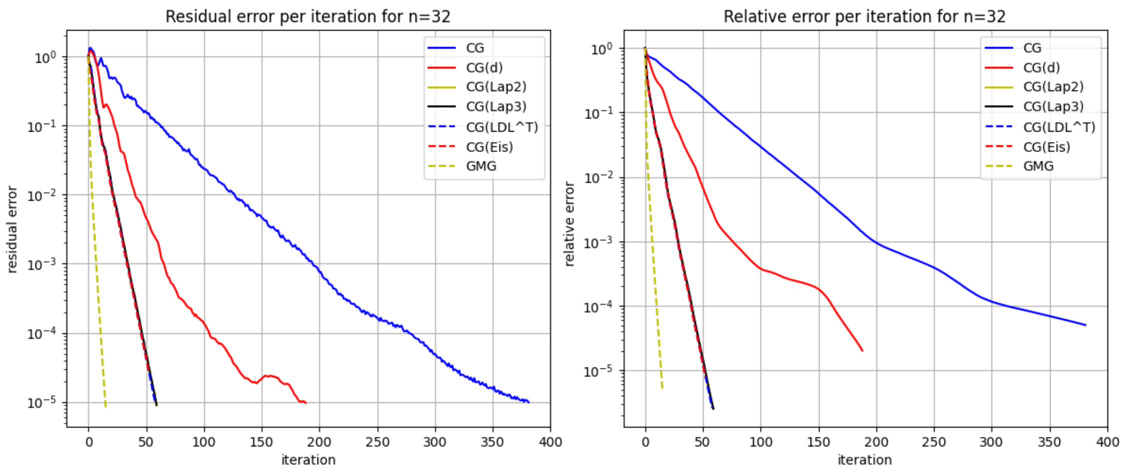

Figure 4.6: Residual and relative error per iteration for a grid with $n_i = 32$.

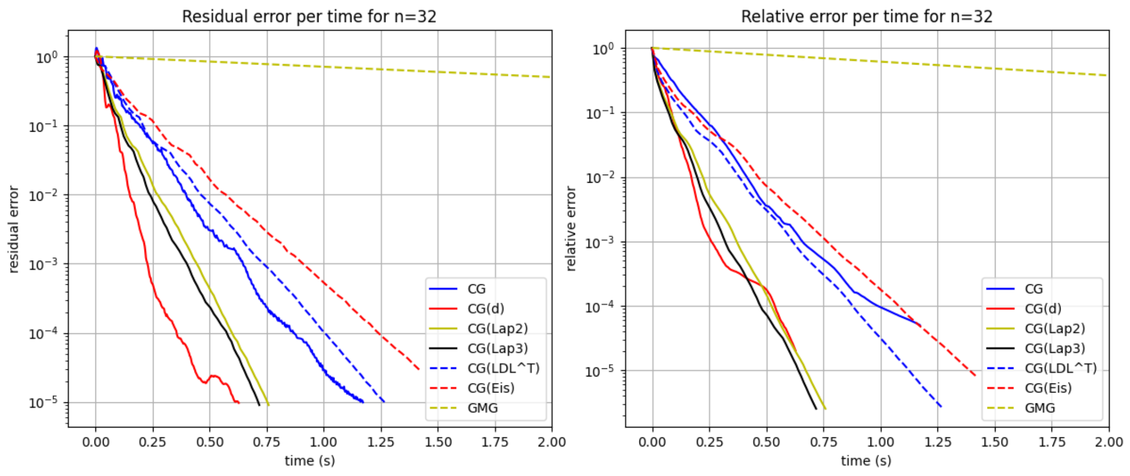

Figure 4.7: Residual and relative error per time for a grid with $n_i = 32$.

The errors per time figures were once again cropped, this time the GMG method took around 100 seconds. One interesting thing we are again able to see here is that the amount of iterations for CG(Lap) and CG(LDLT) / CG(Eis) are the same, with the error decreasing in a very similar manner. With more work, one might be able to find a preconditioner with non-zero off-diagonal blocks of lower computational cost than CG(Lap).

In summary, we found in this Chapter that for large systems it is best to use the CG(Lap2) method. Preconditioners with more information do not add much more value and are more computationally expensive. Further improvements can likely only be achieved by implementing the methods in lower-level programming languages. We continue by doing a detailed analysis on methods for time integration in the next Chapter, where we use CG(Lap2) in each time-step.

5 Temporal Integration

This Chapter starts with an introduction to the methods used for time-stepping in Section 5.1. We then continue with an analysis of the accuracy and the time needed of each of these methods in Section 5.2. After this an extension is made to incorporate high-performance computing in Section 5.2.3.

5.1. Methods

The equation we are evaluating is of the form $\frac{\partial \mathbf{N}^m}{\partial t} = A(\mathbf{N}^m)\mathbf{N}^m + f(\mathbf{N}^m)$ with $-A$ SPD, as in equation 3.13.

5.1.1. First Order Schemes

As usual, we define the Forward Euler (FE) and Backward Euler (BE) method with forward and backward differences for the temporal derivative. Forward difference for a given time-step size of Δt is given by

$$\frac{\partial \mathbf{N}^m}{\partial t} \approx \frac{\mathbf{N}^{m+1} - \mathbf{N}^m}{\Delta t}$$

If substituted into equation 3.13 the value of $\mathbf{N}^{m+1}_{i,j,k}$ can be evaluated with a simple matrix-vector product by moving the known elements to the right-hand side

$$\mathbf{N}^{m+1} = \mathbf{N}^m + \Delta t\big(A(\mathbf{N}^m)\mathbf{N}^m + f(\mathbf{N}^m)\big)$$

We denote the system matrix as $A(\mathbf{N}^m)$ here to highlight the dependency on $\mathbf{N}^m$, which is important for other methods. As Forward Euler is an explicit method, there is a bound on the time step for the method to be stable. This bound was derived in [5], and is given by

$$\Delta t \leq \frac{2}{4D_{\max}\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right) + |k|_{\max} + \alpha C_{\max}} \tag{5.1}$$

The inclusion of the third dimension makes the time step restriction stricter; if $D_{\max}$ is the factor contributing most to the restriction, the maximum time step in 3D is two-thirds of the one in 2D. As the vector $\mathbf{D}$ changes each iteration, it is difficult to respect this bound with full certainty using a fixed value of Δt. In the preceding two works of this thesis, the time step criterion was checked after each parameter-update in the inversion method; if it was too large, the evaluation was done again with half the time step as before. On the other hand, BE is unconditionally stable, with backward difference given by

$$\frac{\partial \mathbf{N}^{m+1}}{\partial t} \approx \frac{\mathbf{N}^{m+1} - \mathbf{N}^m}{\Delta t}$$

It is not too straight-forward to evaluate this system, however. Fixed point iteration would still be unstable for the values of Δt where FE is unstable, so we need to solve it implicitly. The dependency of the matrix A on the vector $\mathbf{N}^{m+1}$ and the quadratic part in the reaction term provide difficulties. We overcome this by making the following assumption about the right-hand side, which has the "easy part implicit, hard part explicit" mindset applied.

$$A(\mathbf{N}^{m+1})\mathbf{N}^{m+1} + f(\mathbf{N}^{m+1}) \approx A(\mathbf{N}^m)\mathbf{N}^{m+1} + \mathbf{k}\cdot\mathbf{N}^m\left(1 - \frac{\mathbf{N}^{m+1}}{\theta}\right) - \alpha\mathbf{C}^{m+1}\cdot\mathbf{N}^{m+1}$$

Note the choice of $\mathbf{N}^{m+1}$ in the reaction term, as this enforces that the resulting system is still negative definite. We take the above to be the right-hand side and the backward difference equation to be the left-hand side, and by rearranging some terms we find the following system for the next time-step.

$$\left(I_n - \Delta t\left(A(\mathbf{N}^m) - \mathbf{k}\cdot\frac{\mathbf{N}^m}{\theta} - \alpha\mathbf{C}^{m+1}\right)\right)\mathbf{N}^{m+1} = \mathbf{N}^m + \Delta t\mathbf{k}\cdot\mathbf{N}^m$$

Solving this can be done once again by CG, as the matrix is definite and symmetric. A diagonal preconditioner is easy to implement, so CG(d) will be done in practice. Moreover, one should see that the matrix on the left-hand side is actually already quite close to the identity matrix, given that Δt is not taken too big. Another possibility for solving this system implicitly is by constructing an enlarged system for both $\mathbf{N}^{m+1}$ and $\mathbf{u}^{m+1}$, and solving them simultaneously. They do this efficiently in [29].

5.1.2. Second Order Schemes

The FE and BE method can be combined into the second-order accurate *Crank-Nicholson* (CN) method, which can be seen as the average value of the two methods. At the cost of needing more time than the individual two methods, the order of accuracy increases to two. The system to solve is now

$$\left(I_n - \frac{\Delta t}{2}\left(A(\mathbf{N}^m) - \mathbf{k}\cdot\frac{\mathbf{N}^m}{\theta} - \alpha\mathbf{C}^{m+1}\right)\right)\mathbf{N}^{m+1} = \mathbf{N}^m + \frac{\Delta t}{2}\mathbf{k}\cdot\mathbf{N}^m + \frac{\Delta t}{2}\left(A(\mathbf{N}^m)\mathbf{N}^m + f(\mathbf{N}^m)\right)$$

We also need to solve this implicitly, which can be done in the same way as for BE. If second order accuracy is required but implicit solutions are difficult to obtain, we could use the *Modified Euler* (ME) method, which is also known as Heun's method. This is a two-step explicit method, and the solution is found as follows.

$$\tilde{\mathbf{N}}^{m+1} = \mathbf{N}^m + \Delta t\left(A(\mathbf{N}^m)\mathbf{N}^m + f(\mathbf{N}^m)\right)$$

$$\mathbf{N}^{m+1} = \mathbf{N}^m + \frac{\Delta t}{2}\left(A(\mathbf{N}^m)\mathbf{N}^m + f(\mathbf{N}^m) + A(\tilde{\mathbf{N}}^{m+1})\tilde{\mathbf{N}}^{m+1} + f(\tilde{\mathbf{N}}^{m+1})\right)$$

This method has the same stability criterion as FE.

5.1.3. RK4 and Generalized Alpha

It remains to see if higher accuracy is necessary for this setting. If so, there is benefit in using the *Runge-Kutta 4* (RK4) method, which is fourth order accurate. The amount of tumour cells at the next time-step is iteratively calculated as follows

$$\begin{aligned}
\mathbf{N}^{m+1} &= \mathbf{N}^m + \tfrac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)\\
k_1 &= A(\mathbf{N}^m)\mathbf{N}^m + f(\mathbf{N}^m)\\
k_2 &= A(\mathbf{N}^m + \tfrac{\Delta t}{2}k_1)(\mathbf{N}^m + \tfrac{\Delta t}{2}k_1) + f(\mathbf{N}^m + \tfrac{\Delta t}{2}k_1)\\
k_3 &= A(\mathbf{N}^m + \tfrac{\Delta t}{2}k_2)(\mathbf{N}^m + \tfrac{\Delta t}{2}k_2) + f(\mathbf{N}^m + \tfrac{\Delta t}{2}k_2)\\
k_4 &= A(\mathbf{N}^m + \Delta t k_3)(\mathbf{N}^m + \Delta t k_3) + f(\mathbf{N}^m + \Delta t k_3)
\end{aligned}$$

For this we need A not only at time t_m and t_{m+1}, but also at the midpoint $t_{m+1/2}$. We can luckily accurately estimate this matrix as shown in the equations. The accuracy is greatly increased here, so the time-steps could plausibly be taken rather large. We might not need to calculate the diffusion at each intermediate step, as the difference will be minimal while the increased computational cost is high. For system matrices with real eigenvalues, the stability region of RK4 is $\sqrt{2}$ times larger than the one of FE.

Another second-order method is the *Generalized* $\alpha-$ method (GA) [30], as used in another mathematically high-level related work on prostate cancer [31]. This method requires values $\mathbf{N}^m$ and $\dot{\mathbf{N}}^m$ to both be known and used. Here the dot indicates the first time derivative for convenience of notation. First a predictor is made for the next time step, with which predictors of two midpoints are assembled

$$\begin{aligned}
\mathbf{N}^{m+1}_{(0)} &= \mathbf{N}^m\\
\dot{\mathbf{N}}^{m+1}_{(0)} &= \tfrac{\gamma-1}{\gamma}\dot{\mathbf{N}}^m\\
\mathbf{N}^{m+\alpha_f}_{(i)} &= \mathbf{N}^m + \alpha_f\big(\mathbf{N}^{m+1}_{(i-1)} - \mathbf{N}^m\big)\\
\dot{\mathbf{N}}^{m+\alpha_m}_{(i)} &= \dot{\mathbf{N}}^m + \alpha_m\big(\dot{\mathbf{N}}^{m+1}_{(i-1)} - \dot{\mathbf{N}}^m\big)
\end{aligned}$$

The subscript i indicates the iteration of the prediction, and a dot above a vector means the first temporal derivative of that quantity. We need to define parameters $\gamma, \alpha_f, \alpha_m$ here, which influence the damping of frequencies. If all three parameters take the value 1, the method coincides with the backward Euler method. For second order stability we require $\gamma = \frac{1}{2} + \alpha_m - \alpha_f$ and $\alpha_m \geq \alpha_f \geq \frac{1}{2}$, which is shown in [32]. They also express the parameters in terms of ρ_∞ as introduced in [30] as follows

$$\alpha_m = \frac{1}{2}\left(\frac{3-\rho_\infty}{1+\rho_\infty}\right), \quad \alpha_f = \frac{1}{1+\rho_\infty}, \quad \rho_\infty \in [0,1]$$

Fro $\rho_\infty = 1$ the highest frequencies are preserved and the method coincides with the midpoint rule, while for $\rho_\infty = 0$ the highest frequency is annihilated in one step. Non-linear problems such as ours have benefit from lower values of this parameter, according to the original authors, but it should be tuned accordingly to each problem. To update the given predictors, we need the nodal values of the non-linear residual, which in our case is defined as

$$\mathbf{R}_{(i)}\big(\dot{\mathbf{N}}^{m+\alpha_m}_{(i)}, \mathbf{N}^{m+\alpha_f}_{(i)}\big) = \dot{\mathbf{N}}^{m+\alpha_m}_{(i)} - A(\mathbf{N}^{m+\alpha_f}_{(i)})\mathbf{N}^{m+\alpha_f}_{(i)} - f\big(\mathbf{N}^{m+\alpha_f}_{(i)}\big)$$

We want this residual to be equal to zero, for which we iterate. To do this, the residual is linearised with respect to the solution variable as

$$\mathbf{R}_{(i)}\big(\dot{\mathbf{N}}^{m+\alpha_m}_{(i)}, \mathbf{N}^{m+\alpha_f}_{(i)}\big) + \frac{d\mathbf{R}_{(i)}\big(\dot{\mathbf{N}}^{m+\alpha_m}_{(i)}, \mathbf{N}^{m+\alpha_f}_{(i)}\big)}{d\dot{\mathbf{N}}^{m+1}_{(i)}}\Delta\dot{\mathbf{N}}^{m+1}_{(i)} = 0$$

The tangent matrix of $\mathbf{R}_{(i)}$ here will be denoted as $\mathbf{S}_{(i)} = \dfrac{d\mathbf{R}_{(i)}\big(\dot{\mathbf{N}}_{(i)}^{m+\alpha_m}, \mathbf{N}_{(i)}^{m+\alpha_f}\big)}{d\dot{\mathbf{N}}_{(i)}^{m+1}}$, and it can be calculated explicitly as

$$\begin{aligned}
\mathbf{S}_{(i)} &= \frac{\partial \mathbf{R}_{(i)}\big(\dot{\mathbf{N}}_{(i)}^{m+\alpha_m}, \mathbf{N}_{(i)}^{m+\alpha_f}\big)}{\partial \dot{\mathbf{N}}_{(i)}^{m+\alpha_m}} \frac{\partial \dot{\mathbf{N}}_{(i)}^{m+\alpha_m}}{\partial \dot{\mathbf{N}}_{(i)}^{m+1}} + \frac{\partial \mathbf{R}_{(i)}\big(\dot{\mathbf{N}}_{(i)}^{m+\alpha_m}, \mathbf{N}_{(i)}^{m+\alpha_f}\big)}{\partial \mathbf{N}_{(i)}^{m+\alpha_f}} \frac{\partial \mathbf{N}_{(i)}^{m+\alpha_f}}{\partial \dot{\mathbf{N}}_{(i)}^{m+1}} \\
&= \alpha_m \frac{\partial \mathbf{R}_{(i)}\big(\dot{\mathbf{N}}_{(i)}^{m+\alpha_m}, \mathbf{N}_{(i)}^{m+\alpha_f}\big)}{\partial \dot{\mathbf{N}}_{(i)}^{m+\alpha_m}} + \alpha_f \gamma \Delta t \frac{\partial \mathbf{R}_{(i)}\big(\dot{\mathbf{N}}_{(i)}^{m+\alpha_m}, \mathbf{N}_{(i)}^{m+\alpha_f}\big)}{\partial \mathbf{N}_{(i)}^{m+\alpha_f}} \\
&= \alpha_m I_n + \alpha_f \gamma \Delta t \Big(-A(\mathbf{N}_{(i)}^{m+\alpha_f}) - \frac{\partial A(\mathbf{N}_{(i)}^{m+\alpha_f})}{\partial \mathbf{N}_{(i)}^{m+\alpha_f}} \mathbf{N}_{(i)}^{m+\alpha_f} - \operatorname{diag}\Big(\mathbf{k} - \frac{2}{\theta}\mathbf{k}\cdot\mathbf{N}_{(i)}^{m+\alpha_f} - \alpha C\Big)\Big)
\end{aligned}$$

The derivatives of $\dot{\mathbf{N}}_{(i)}^{m+\alpha_m}$ and $\mathbf{N}_{(i)}^{m+\alpha_f}$ follow from the definition of their iterative updates introduced earlier, and the update of $\mathbf{N}_{(i)}^{m+1}$ which follows shortly. The derivative of $\mathbf{R}_{(i)}$ with respect to $\mathbf{N}_{(i)}^{m+\alpha_f}$ contains the term $-\frac{\partial A(\mathbf{N}^{m+\alpha_f})}{\mathbf{N}^{m+\alpha_f}}\mathbf{N}^{m+\alpha_f}$, of which the first term can be derived analytically as follows

$$\frac{\partial A}{\partial \mathbf{N}} = \frac{\partial A}{\partial \mathbf{D}} \cdot \frac{\partial \mathbf{D}}{\partial \sigma_{vm}} \cdot \frac{\partial \sigma_{vm}}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial \varepsilon} \cdot \frac{\partial \varepsilon}{\partial \mathbf{u}} \cdot \frac{\partial \mathbf{u}}{\partial \mathbf{N}} \tag{5.2}$$

In the derivative $\frac{\partial D}{\partial \sigma_{vm}}$ we multiply with γ, and in the derivative $\frac{\partial u}{\partial N}$ we multiply with λ, both of which are in the order 10^{-3}. This suggests that this term might be negligibly small. Finding the search direction $\Delta\mathbf{N}_{(i)}^{m+1}$ leads to solving this linear problem in each corrector step

$$\mathbf{S}_{(i)} \Delta \dot{\mathbf{N}}_{(i)}^{m+1} = -\mathbf{R}_{(i)}$$

We omit the parantheses and parameters here for ease of notation. This symmetric system can be solved with the conjugate gradient method, which we speed up with a diagonal preconditioner. The approximate value at the next time step can then be updated as such

$$\begin{aligned}
\dot{\mathbf{N}}_{(i+1)}^{m+1} &= \dot{\mathbf{N}}_{(i+1)}^{m+1} + \Delta \dot{\mathbf{N}}_{(i)}^{m+1} \\
\mathbf{N}_{(i+1)}^{m+1} &= \mathbf{N}_{(i)}^{m+1} + \gamma \Delta t \Delta \dot{\mathbf{N}}_{(i)}^{m+1}
\end{aligned}$$

After this we update the values of $\mathbf{N}$ at the alpha-levels again and iterate further, up until $i = i_{\max}$. Just like with the fixed point method, we likely only need a few iterations to reach a good enough solution.

5.2. Accuracy Analysis

We now check how accurate each method is and weigh it up against the time needed. All simulations are executed for 64 days with varying time-steps. The linear elastic sub-problem is solved with the CG(Lap2) method, as that gave the best results in the previous Chapter. Additional parameters used here are $\gamma = 2.0e-3$, $\beta = 0.25 d^{-1}$, $D_0 = 2.0e-3 mm^3 d^{-1}$ and $\alpha = 9.0e-2(\mu M \cdot d)^{-1}$. The function used for the drug concentration is based on the NBVM method, such that $C_{tissue}^{drug}(\bar{x}, t) = C_0 \exp\big(-\beta(t \mod t_{int})\big)$. Here the time interval between admission is $t_{int} = 15$ days, and the values for C_0 in μM are $0.28, 0.16, 0.70$ for fibroglandular, adipose and tumour tissue respectively [1].

The proliferation rate is chosen based on the initial tumour profile and mimics the results of [5]; the growth of the tumour is largest at its boundary, and we define it by $k(r) = 2\cdot 10^{-3}(1.5 e^{-1200 r^6} - 1.25 e^{-600 r^4})$ with the radius $r = \sqrt{(x-{}^1\!/\!_2)^2 + (y-{}^1\!/\!_2)^2 + (z-{}^1\!/\!_2)^2}$ on scaled coordinates x, y, z between 0 and 1. The function in 1D is shown in the next figure.

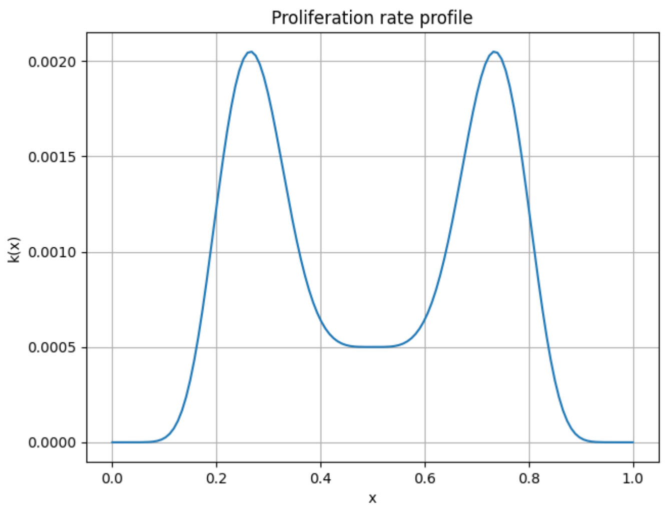

Figure 5.1: 1D representation of the proliferation rate.

In the remainder of this section recycling is used in the linear elastic sub-problem for all methods, with the starting vector being the solution of the previous time-step. This resulted in a drop of iterations needed by up to 50%, with on average a drop of 37.5%. The lowest frequency eigenvector of B as the starting vector gave similar results, but was computationally very expensive to obtain.

5.2.1. Comparison

We evaluate all the schemes as defined above, on an interval of $t_1 = 10$ days with a time-step of $dt = 1$ day. The FE method in our setting is stable then, which means that the ME and RK4 methods should be stable too, as their regions of stability for real functions are at least as big as FE. The time-step dt and the final time t_1 are not too important for this experiment, as we can assume that the time needed per iteration for each method is more or less constant. Nonetheless, we still average the result over 10 runs again. The time needed for all methods is shown in the figure below for varying grid sizes.

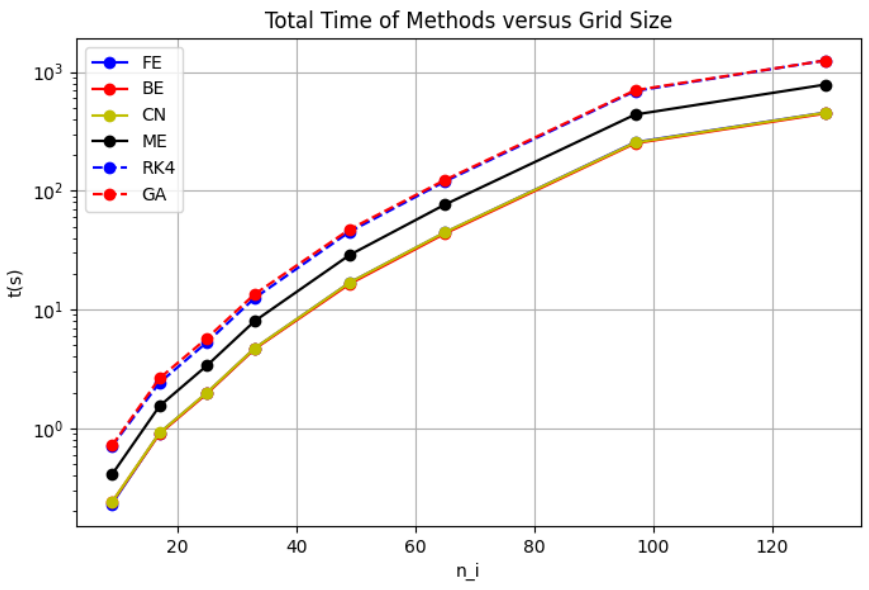

Figure 5.2: Time needed per method for 10 iterations for different grids of size $n_i \times n_i \times n_i$, averaged over 10 runs.

Note that the FE/BE/CN and RK4/GA lines mostly overlap. A few interesting observations can be made here. First of all, the difference in time needed per method grows but there seems to be a more-or-less constant factor between them. In some of the systems, the CN method is slightly faster than the BE method. The first reasonable explanation for this would be that the condition number of the system that CN is trying to solve is better. However, it turns out that for both CN and BE the CG(d) method solves the time step in a single iteration to the required accuracy of 10^{-5}. The need for only a single iteration is also the reason that they are as fast as FE; only one additional matrix-vector multiplication is needed, and the majority of the time is spend in the linear-elastic sub-problem. It seems that the time difference between CN and BE is not significant, but it is nice to see that the second order method CN does not need any additional computation time with respect to the first order BE. As expected, RK4 is one of the slowest methods with its high number of steps, but it is still reasonably fast. Looking at the time needed for the second order GA and ME method, they are not a competitive method in comparison to CN.

To see the workings of all methods, we analyse them further for $n_i = 65$ up to $t_1 = 64$. This choice of n_i is large enough for a representative analysis, small enough that results can be acquired in reasonable time, and it makes the FFT in the linear-elastic sub-problem efficient on the $n_i - 1$ points. The next figure shows the total amount of tumour cells over time as found by all methods in this setting.

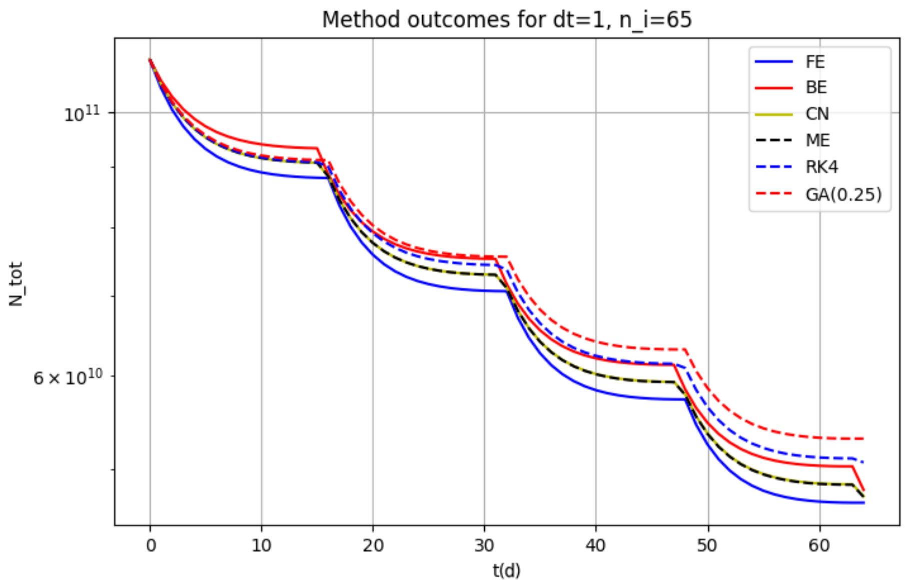

Figure 5.3: Time needed per method for 10 iterations for different grids of size $n_i \times n_i \times n_i$, averaged over 10 runs.

As the chemotherapy agent is assumed to be inserted every 15 days, there is a sharp drop in the total amount of tumour cells at these time-points. The exact ‘elbow’-points differ a bit in location per method, as forward-looking methods encounter the increased chemo-term earlier (viz. FE versus BE). The RK4 method has the highest accuracy, so we can assume that its result is the closest to the truth. However, a time-step of $dt = 1$day is still relatively high, so even this result can still be off a bit. At t_1 there is quite some difference between the methods, with a factor of 1.15 between the biggest and the smallest number. This suggests that accuracy of the method indeed plays a big role, which we investigate more in the next section.

The choice of the damping parameter in the generalized α method up until now was $\rho_\infty = 0.25$, as suggested by [30]. We check what the influence of this parameter is on the simulation. In the next figure the output is shown in the same setting as before.

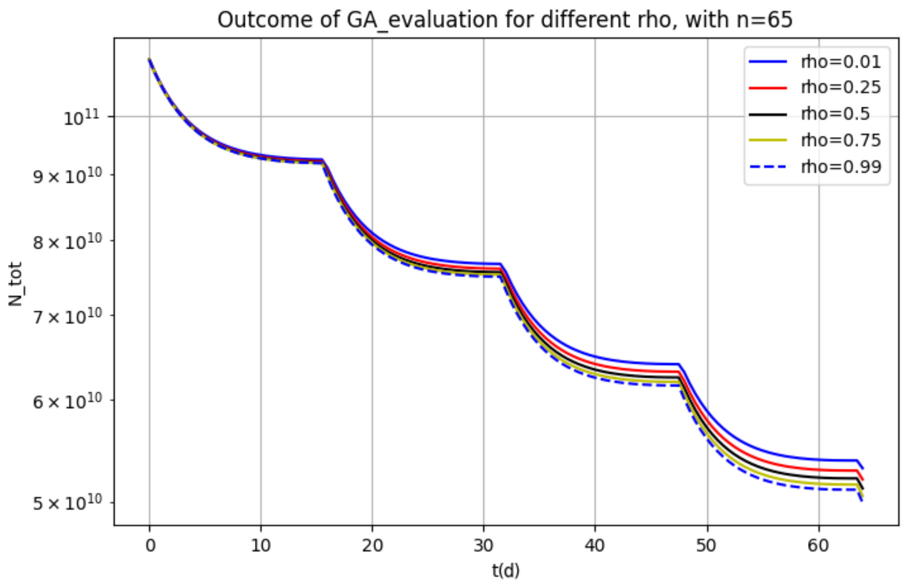

Figure 5.4: Outcome of the Generalised α method for different values of ρ_∞, with $t_1 = 64$ on a $65\times65\times65$ grid and $dt = 1$.

We see that the higher the parameter ρ_∞ gets, the lower the output at t_1 is. Seeing how the solution in Figure 5.2 of GA is too high, it suggests that for our problem there is benefit in using higher values of ρ_∞. Our prediction was that we would benefit from lower values in our problem due to the non-linearity, as stated in [30]. However, we also mentioned that for lower values of ρ_∞ the highest frequencies would be annihilated faster, which resulted in our periodic exponentially decaying chemotherapy term being smoothed out. This is most likely the cause of the overshoot of the solution. Despite the solution of GA being very close to reality for higher values of ρ_∞, its required time is too large for it being competitive in comparison to other methods. Especially when compared to CN, which is also second order accurate. A last thing to note about GA is that we can indeed neglect the small but hard-to-compute term; this results in an error of less than 10^{-6} in the found solution.

It is interesting to see that all other methods consistently underestimate the amount of tumour cells at t_1. Generally, explicit method overestimate the change in values while implicit method underestimate it. The implicit methods likely don't show this behaviour due to their semi-explicitness in the evaluation of the linear-elastic sub-problem. Higher accuracy could be achieved by evaluating this iteratively as well in the implicit solve, or by using multi-step methods like Adam-Bashforth.

5.2.2. Order of errors

In the previous section we saw that the accuracy of the method differs hugely. To investigate this, we look at the outcome of the first order FE, second order CN and fourth order RK4 method. This is done for time-steps of size $dt = 0.25, 0.5, 1, 2, 5$ up until $t_1 = 64$. Note that in our setting of parameters stability is still guaranteed with theses step sizes. The results are shown in the next three figures, shown over the next two pages.

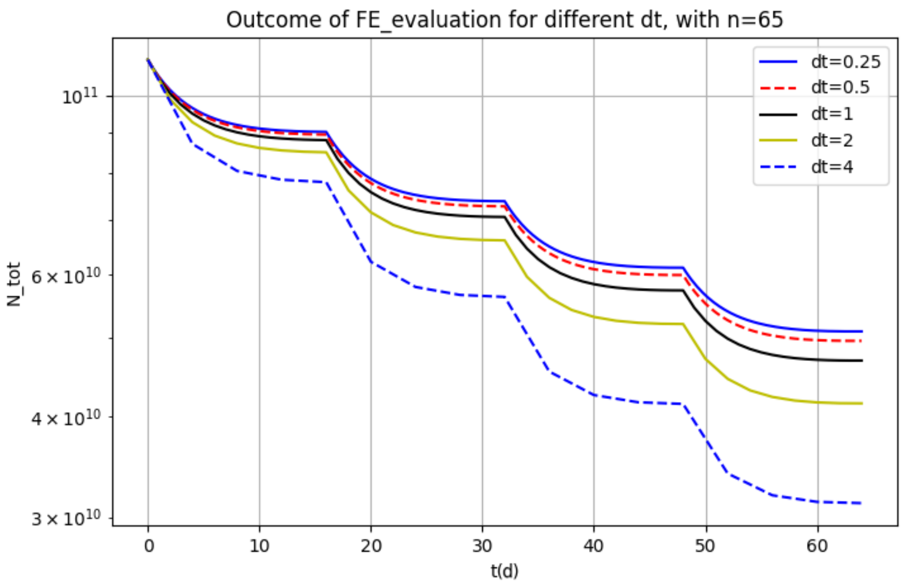

Figure 5.5: Outcome of the Forward Euler method for different time-steps dt, with $t_1 = 64$ on a $65 \times 65 \times 65$ grid.

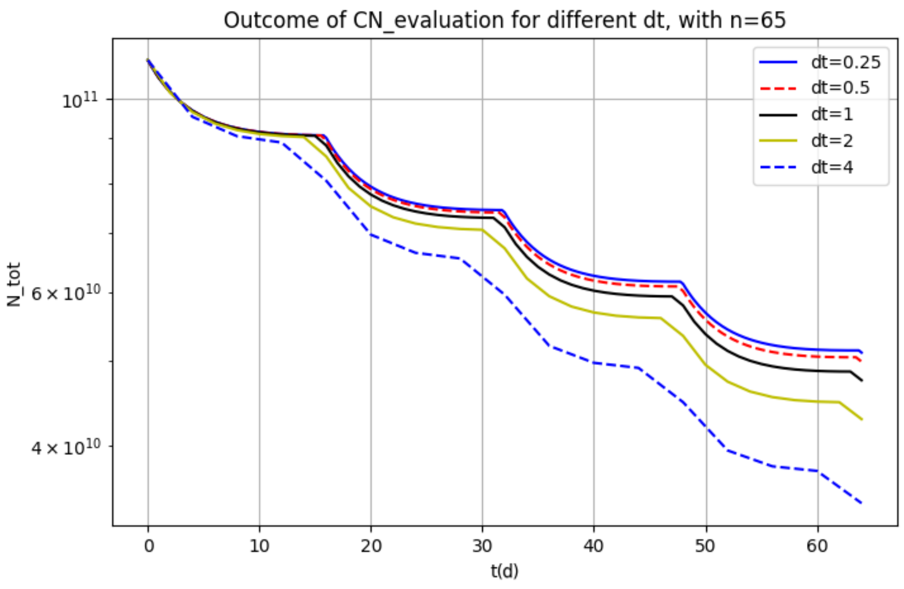

Figure 5.6: Outcome of the Crank Nicolson method for different time-steps dt, with $t_1 = 64$ on a $65 \times 65 \times 65$ grid.

The first observation we make is that for all three methods the total number of tumour cells goes up when dt is decreased. With big time-steps, the effect of the chemo-therapy term is more dominant, as high values of its exponentially decaying term are considered for larger periods of time. We also see for all methods that the error at t_1 halves whenever the time-step is halved, assuming that the 'true' solution lies slightly above $dt = 0.25$. For the first-order method FE this is correct, though for the second- and fourth-order method we expected more rapid convergence. A likely cause for this is that in the derivation of the order of error for the higher order methods we assume that our function is twice differentiable. Due to the non-linearity of the chemotherapy term we do not achieve this, and hence for larger values of Δt the accuracy is worse. It is clear nonetheless that the higher the order of accuracy, the earlier the method converges to the true solution.

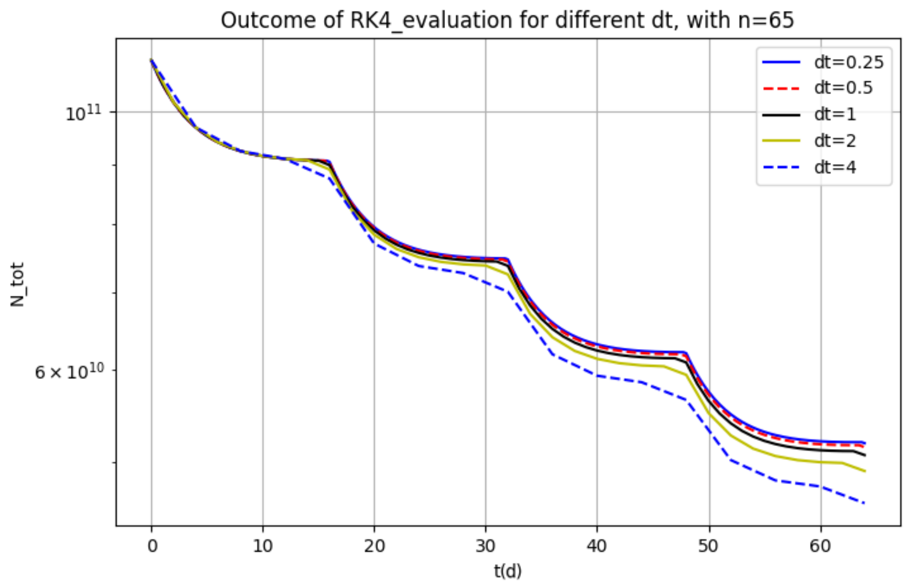

Figure 5.7: Outcome of the Runge-Kutta 4 method for different time-steps dt, with $t_1 = 64$ on a $65 \times 65 \times 65$ grid.

From the three images we can conclude that we either need a high order of accuracy or a small time-step in order to make a good prediction. If the inverse problem later on is solved for a lower accuracy method, the parameters we obtain will not be representative for the patient. Without a doubt this would result in bad results for the error at time t_2, assuming that the model itself is an accurate description of reality. The RK4 method with $dt = 1$ and the CN method with $dt = 0.5$ should both be good enough. Looking at figure 5.2 and assuming that a simulation with $dt = 0.5$ takes twice as long to execute, we can add $\log_{10}(2) \approx 0.3$ to the logarithmic $y-$axis to deduce that the CN method is faster than the RK4 method with half the time-step. There are methods to speed RK4 up though, which we see in the next section.

5.2.3. Parallel Timestepping

There is still a huge plausible reduction in computation time by having the time-steps be executed in parallel. This is where the Parareal algorithm [33] comes in. For this algorithm we need a fast coarse solver and a fine high accuracy solver of the initial value problem. Furthermore, we have a parallel computing CPU infrastructure with P processors.

Let us denote $\mathscr{C}(\mathbf{N}^m, dt_c, t_{next})$ as the solution of the coarse solver with initial value $\mathbf{N}^m$ at time t_{next} with coarse step size dt_c, and we define $\mathscr{F}(\mathbf{N}^m, dt_f, t_{next})$ likewise. The coarse time step should depend on the amount of processors we are using, such that $dt_c := t_{\max}/P$. The fine step dt_f size should be a divisor of dt_c, and it should be in the resolution that we want the final solution to be. Our first initial guess of the solution is then fully defined by the coarse solver as $\mathbf{N}_0^{m+1} = \mathscr{C}(\mathbf{N}_0^m, dt_c, t_m + dt_c)$, where $\mathbf{N}_0^0$ is the initial value. Then we run the fine solver in parallel on all the time 'slices', and calculate all the values $\mathscr{F}(\mathbf{N}_i^m, dt_c, t_m + dt_c)$ for the iteration number i. The next iterates are then found with a head-tail-esque method, such that the discontinuous slices are continuous again, and the new coarse iterates are $\mathbf{N}_{i+1}^{m+1} = \mathscr{C}(\mathbf{N}_{i+1}^m, dt_c, t_m + dt_c) + \mathscr{F}(\mathbf{N}_i^m, dt_c, t_m + dt_c) - \mathscr{C}(\mathbf{N}_i^m, dt_c, t_m + dt_c)$. This algorithm continues iterating until the iterates converge, which we check by evaluating $||\mathbf{N}_{i+1}^{m+1} - \mathbf{N}_i^{m+1}|| < \epsilon ||\mathbf{N}_0||$ with tolerance $\epsilon = 10^{-5}$.

In our setting, the coarse solver to be used should be fast and able to make big steps. The unconditionally stable CN method should be appropriate for this; another option would be BE, but CN is more accurate and takes the same amount of time to calculate. According to the previous Section, the CN method is only approximately three times as fast as the RK4 method, which is not enough. Hence we skip the linear elastic sub-problem in the coarse solver, and set the Von Mises stress to be zero everywhere. The fine solver should be the fourth order accurate RK4 method; from the previous two Sections we see that these choices are the most valid. In the next figure, the workings of the parareal algorithm is demonstrated for a small example with $n_i = 33$, $t_1 = 64$, $dt_f = 1$, $P = 8$ and $dt_c = 8$.

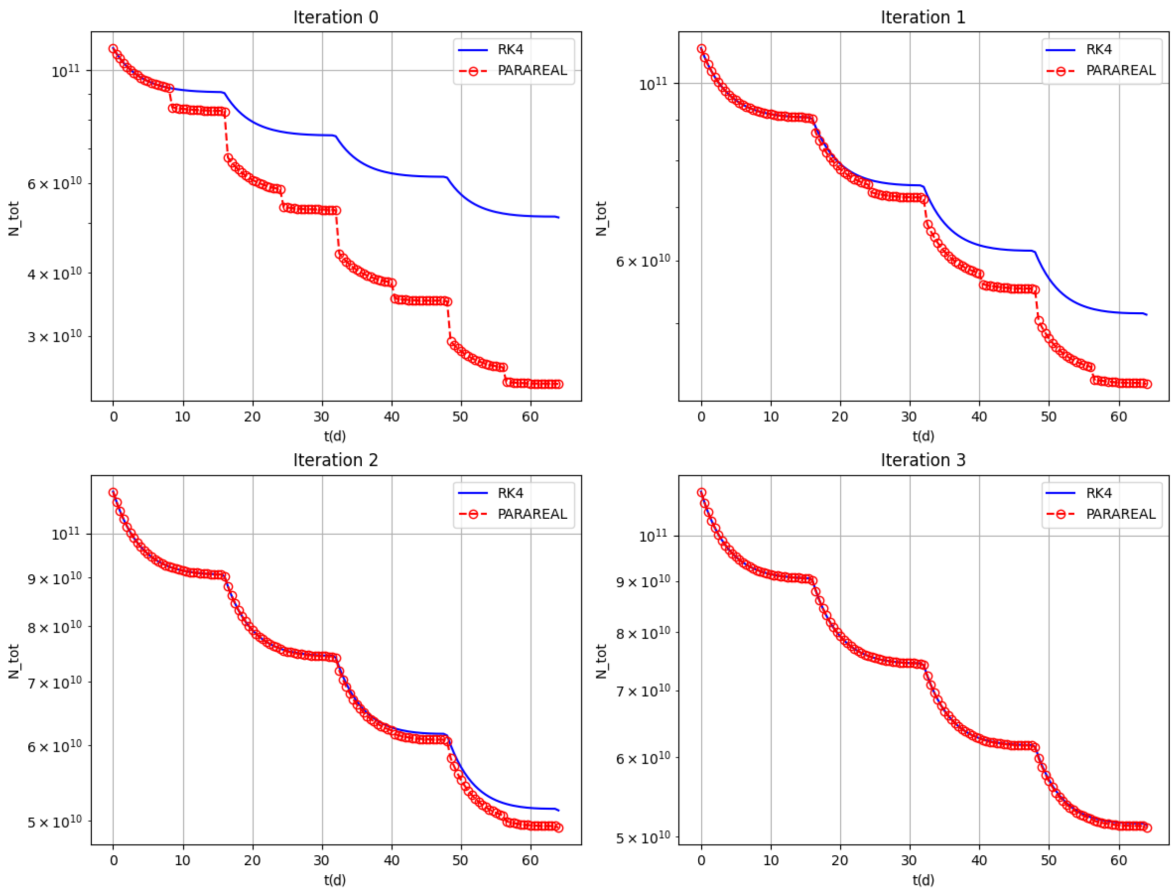

Figure 5.8: Output of the Parareal algorithm, with CN as the coarse solver and RK4 as the fine solver, with $n_i = 33$, $P = 8$, $dt_f = 1$.

This image nicely demonstrates how the individual slices converge towards each other, and how they do so in 4 iterations. Note that the actual parareal method will take 5 iterations, as it will only terminate when the relative difference between subsequent iterations are small enough. If we were to run the normal RK4 method in the same setting, the amount of sequential RK4 steps would be the same computational cost as in 8 Parareal iterations, neglecting the time needed for the CN steps. This results in a theoretical maximum speedup of $S_8 = 2$ for this small example. With more processors this speedup should be even larger; this example was just taken for its visual interpretability.

Now we would like to test the speedup in practice on large-scale examples. For this we made use of the Delft Blue Supercomputer [34], with the MPI4PY-package [35]. The CPUs we use are Intel XEON E5-6248R 24C 3.0GHz, with 48 cores and 192GB RAM. The test setting we will be using now has $n_i = 65$ and $dt_f = 0.5$, and $t_1 = 64$ is unchanged. The speedup is tested for $P = 4, 8, 16, 32, 64$, and the results of this test are shown in the table below.

Table 5.1: Speedup S_P achieved for a differing amount of processors P, as well as the amount of iterations and actual time needed t in seconds. Corrected times and speedups are marked with an asterisk $*$.

P	1	4	8	16	32	64
iters	-	4	5	4	3	3
$t(s)$	835.34	805.17	657.17	417.88	220.69	179.89
S_P	1	1.04	1.27	2.00	3.76	4.64
$t^*(s)$	835.34	732.68	409.09	202.18	114.90	128.22
S_P^*	1	1.14	2.04	4.13	7.27	6.51

The first column represents the output of the sequential algorithm. The results are great, and we see a maximum speedup of 4.64. They are not as high as expected, however. After a lot of tweaking it was found that this does not have to do with the algorithm or the model, but with the working of PYTHON and MPI4PY on the Delft Blue hardware. Some processors were calculating at lower speeds than the others (up to $2.5\times$), even when executing basic scripts. To show the attainable results, two more rows are shown in the table. These show the corrected time measure, where every processor takes as much time to do the coarse solve as the fastest, and the corresponding corrected speedup. Here we see that the speedup is more in line with the theoretical one, with a maximum speedup of 7.27.

The difference between the theoretical speedup and the one in practice is mainly explained by the time needed for the coarse solver. The corrected communication time is in the order of $0.1 - 1s$, so this does not play a significant role. For the 'fast' processors, the communication time is rather high as they need to wait for the slower ones.

An important consideration to make is that the more processors we use, the more calculations we need to do with the coarse solver. As this method should inherently be fast this is not that big of an issue for a low to medium amount of processors. When 8 processors are used, we spend 2.10% of the full calculation time on the coarse solver, but for 64 processors this shoots up to 40.2%. The cost of the coarse solver can be reduced further, for example by solving it on a coarser grid and interpolating the result. In the corrected timings we see that the total computation time actually increases when going from 32 to 64 processors.

We conclude this chapter with the decision to use Parareal for time integration with the amount of processors equal to a quarter of the amount of time-steps. If that many aren't available, we use as many as possible. The fine solver is the Runge-Kutta 4 method, and the coarse solver the Crank-Nicolson method.

6 Parameter Estimation

In this Chapter the inverse problem is solved on the test data-set. The assumptions of the simulated data are the same as in the previous two Chapters, unless stated otherwise. The methods which are to be tested are first introduced in Section 6.1, after which we compare their workings and best-found results in Section 6.2.

6.1. Methods

The equation to minimise in its general form is $\mathfrak{F}(\mathbf{P}^{(k)}) = ||\mathfrak{f}(\mathbf{P}^{(k)})||_2^2 = ||\mathbf{N}_{data}(t_1) - \mathbf{N}_{model}(t_1;\mathbf{P}^{(k)})||_2^2$ for a parameter vector $\mathbf{P}^{(k)}$. Sensitivity of this equation is mathematically described by the Jacobian of the model. Regularisation terms are added later as well, to enforce uniqueness of the solution.

6.1.1. Non-linear optimisation

The most common way we found in literature of solving the inverse problem is the *Levenberg Marquardt* (LM) algorithm [36]. This algorithm is initiated with an initial guess $\mathbf{P}^{(0)}$, which we are free to choose and should base on literature values. To find the correct search direction s_k for our next guess $\mathbf{P}^{(k+1)} := \mathbf{P}^{(k)} + s_k$ we linearise the model prediction using the Jacobian.

$$\mathbf{N}_{model}\big(t,\mathbf{P}^{(k)} + s_k\big) \approx \mathbf{N}_{model}\big(t,\mathbf{P}^{(k)}\big) + J s_k, \quad J := \frac{\partial \mathbf{N}_{model}\big(t,\mathbf{P}^{(k)}\big)}{\partial \mathbf{P}^{(k)}}$$

With this linearisation, we can approximate the error of a plausible next iterate with parameters $\mathbf{P}^{(k)} + s_k$ as

$$\begin{aligned}||\mathbf{N}_{data}(t_1) - \mathbf{N}_{model}(t_1;\mathbf{P}^{(k)} + s)||_2^2 \approx{}& ||\mathbf{N}_{data}(t_1) - \mathbf{N}_{model}(t_1;\mathbf{P}^{(k)})||_2^2 \\ &- 2\big(\mathbf{N}_{data}(t_1) - \mathbf{N}_{model}(t_1;\mathbf{P}^{(k)})\big)^\top J s_k + s_k^\top J^\top J s_k\end{aligned}$$

Setting the derivative of this entity with respect to s_k equal to zero gives the following normal equations

$$\big(J^\top J\big) s_k = J^\top \mathfrak{f}(\mathbf{P}^{(k)})$$

What we derived here is actually what is known as the *Gauss-Newton* (GN) method. The modification LM gives to this is by adding a damping factor λ_d to the equation as follows

$$\big(J^\top J + \lambda_d I_n\big) s_k = J^\top \mathfrak{f}(\mathbf{P}^{(k)}) \tag{6.1}$$

For a large value of this λ_d the method resembles the *Gradient Descent* (GD) method; indeed we see that s_k will go in the direction of the biggest descent. The algorithm starts with a large value of

λ_d, which gradually decreases each iteration. If a new iterate has a higher error than the previous, λ_d is increased instead. The factors by which we decrease and increase are 12 and 3 respectively, which are taken from [37]. The termination can be done whenever we want, such as when there is no improvement for a set amount of iterations, when the error becomes sufficiently small or after a set amount of time. For this method we need to compute the Jacobian, which will be talked about in the next sub-section.

The second method we are testing is the *Trust Region Reflective* (TRF) algorithm [38], which has a big benefit with respect to LM that bounds can be imposed on the parameters. The idea of Trust regions algorithms is that at iteration k the next iterate is determined as a solution to a quadratic sub-problem, which approximates the model well within a certain region, i.e.

$$\mathfrak{F}(\mathbf{P}^{(k)} + s) - \mathfrak{F}(\mathbf{P}^{(k)}) \approx \psi_k(s) = g_k^\top s + \frac{1}{2} s^\top (H_k + C_k) s, \quad \forall s : ||D_k s|| \leq \Delta s_k$$

$$g_k = \nabla \mathfrak{f}(\mathbf{P}^{(k)}), \quad H_k = \nabla^2 \mathfrak{f}(\mathbf{P}^{(k)}), \quad C_k = D_k \mathrm{diag}(g_k) J_k^v D_k$$

From the derivation of the LM method we find that we can approximate the gradient and Hessian as $g_k \approx -2J^\top \mathfrak{f}(\mathbf{P}^{(k)})$ and $H_k \approx 2J^\top J$. The matrix D_k is a scaling matrix dependent on the bounds $[l_i, u_i]$ of the parameters. It is defined as follows using auxilliary vector $\vec{v}$.

$$\vec{v} = [v_i(\mathbf{P})], \quad v_i(\mathbf{P}) = \begin{cases} u_i - \mathbf{P}_i & g_i < 0, u_i < \infty \\ \mathbf{P}_i - l_i & g_i \geq 0, l_i > -\infty \\ -1 & g_i < 0, u_i = \infty \\ 1 & g_i \geq 0, l_i = -\infty \end{cases}, \quad D_k = \mathrm{diag}\big(|\vec{v}(\mathbf{P}^{(k)})|^{-1/2}\big)$$

With this matrix D_k we define the last matrix C_k, which enforces the bounds of the parameters. It involves the Jacobian matrix J_k^v of $\vec{v}$, which is diagonal and has a -1 in the first case of the values for v_i, a 1 in the second case, and zeroes elsewhere. The minimum of the quadratic approximation $\psi_k(s)$ is denoted as s_k, and it represents the best parameter step for the next iterate. In smaller systems one can find this minimum with matrix factorisation, but this is not viable for us. Instead we use the Conjugate Gradient Steihaug [39] method, which works as shown in Algorithm 4.

With this method we can find the best next iterate within the bounds of our parameters. Just as in regular CG, the parameter ε here represents a relative tolerance stopping criterion. We can now define the ratio between the approximated and actual reduction in objective value $\rho_k = \big(\mathfrak{f}(\mathbf{P}^{(k)} + s_k) - \mathfrak{f}(\mathbf{P}^{(k)})\big)/\psi_k(s_k)$. We also define some parameters $0 < \mu < \eta < 1$ and $0 < \gamma_0 < \gamma_1 < 1 < \gamma_2$ which decide what we do in each iteration. If $\rho_k > \mu$, i.e. the prediction is accurate, we set $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)} + s$, else we leave it at $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)}$. Then we update the trust region: if $\rho_k < 0$ we pick $\Delta s_{k+1} = \gamma_0 \Delta s_k$, if $\rho_k < \mu$ we pick $\Delta s_{k+1} = \max(\gamma_0 \Delta s_k, \gamma_1 ||D_k s_k||)$, if $\rho_k \in (\mu, \eta)$ we pick $\Delta s_{k+1} = \max(\gamma_1 \Delta s_k, ||D_k s_k||)$, and if $\rho_k \geq \eta$ we pick $\Delta s_{k+1} = \max(\Delta s_k, \gamma_2 ||D_k s_k||)$. This choice of trust region updates is common in most literature on trust regions, but there are other options. The norm can also be chosen freely, and we use the L_2-norm as in the rest of this thesis.

Algorithm 4 Conjugate Gradient Steihaug

```
1: $\mathbf{s}_k^0 := 0, \quad \mathbf{r}_0 := -g_k, \quad \mathbf{z}_0 = D_k^{-2}\mathbf{r}_0, \quad \mathbf{d}_0 = \mathbf{z}_0$
2: $j := 0$
3: **While** stopping criterion not met **do**
4:     $\gamma_j = \mathbf{d}_j^\top H_k \mathbf{d}_j$
5:     **If** $\gamma_j \leq 0$
6:         $\tau = \min\{\tau\,;\, ||D_k(\mathbf{s}_k^j + \tau\mathbf{d}_j)|| = \Delta s_k\}$
7:         $\mathbf{s}_k^j = \mathbf{s}_k^j + \tau\mathbf{d}_j$
8:         **Return**
9:     $\alpha_j = \mathbf{r}_j^\top \mathbf{z}_j / \gamma_j$
10:    $\mathbf{s}_k^{j+1} = \mathbf{s}_k^j + \alpha_j \mathbf{d}_j$
11:    **If** $||D_k \mathbf{s}_k^{j+1}|| > \Delta s_k$
12:        $\tau = \{\tau\,;\, ||D_k(\mathbf{s}_k^j + \tau\mathbf{d}_j)|| = \Delta s_k\}$
13:        $\mathbf{s}_k^j = \mathbf{s}_k^j + \tau\mathbf{d}_j$
14:        **Return**
15:    $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j H_k \mathbf{d}_j$
16:    **If** $||D_k \mathbf{r}_{j+1}|| < \varepsilon ||D_k g_k||$
17:        $\mathbf{s}_k^j = \mathbf{s}_k^{j+1}$
18:        **Return**
19:    $\mathbf{z}_{j+1} = D^{-2}\mathbf{r}_{j+1}$
20:    $\beta_j = \mathbf{r}_{j+1}^\top \mathbf{z}_{j+1} / \mathbf{r}_j^\top \mathbf{z}_j$
21:    $\mathbf{d}_{j+1} = \mathbf{z}_{j+1} + \beta_j \mathbf{d}_j$
22:    $j = j + 1$
23: **end While**
24: **Return** $\mathbf{s}_k^j$
```

A method combining the ideas of LM and trust regions is Powell's dog leg method [40–42]. For this method we define the same trust region as in TRF with H_k, g_k and Δs_k again. This time we compute the Gauss-Newton direction s_{gn} and the steepest descent direction s_{sd} of this quadratic model, i.e.

$$H_k s_{gn,k} := g_k, \quad s_{sd,k} := \frac{g_k^\top g_k}{g_k^\top H_k g_k} g_k$$

The Gauss-Newton step is obtained with a simple CG(d) solve. These steps are similar to the directions one would take in LM. With these two steps we update $\mathbf{P}^{(k)}$ as follows: if $||D_k s_{gn,k}|| \leq \Delta s_k$ we set $s = s_{gn,k}$, if both $||D_k s_{gn,k}|| > \Delta s_k$ and $||D_k s_{sd,k}|| > \Delta s_k$ we set $s = \Delta s_k / ||s_{sd,k}|| \cdot s_{sd,k}$, and else we set $s = s_{sd,k} + \tau(s_{gn,k} - \tau_1 s_{sd,k})$ where τ defines the dogleg point, and is chosen such that $||D_k s|| = \Delta s_k$. Our next attempted parameter vector is then $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)} + s$, the output of which is evaluated. The scaling matrix D_k is the same as in TRF, and the trust region is also updated in the same way with the model function $\psi_k(s)$. The third option for the search direction is called the 'dog leg' step, as it is a broken line which approximates the optimal path for our parameter to take. Note that while this method uses the same trust region, it does not respect the upper and lower bounds strictly. There is still incentive to stay within the region, and if needed an adjustment can be made such that the bounds are strict; this is called the dog-box method [43].

Lastly we try out the adjoint state method, of which great explanations are given in [29, 44]. For this method we need a primary equation which contains all to-be-optimised parameters; fortunately they are all in the reaction-diffusion equation already, so we can take that. If parameters of the linear-elastic equation were to be optimised, a equation for a simultaneous solve would need to be stated. We then define the adjoint equation for the adjoint state variable $\phi(\bar{x},t)$ as follows

$$\frac{\partial\phi(\bar{x},t)}{\partial t} = A^*(\phi)\phi(\bar{x},t) + f^*(\phi)$$

$$A^* = A, \quad f^*(\phi) = \mathbf{k}\cdot\phi(\bar{x},t)\Big(1 - \frac{2N(\bar{x},t)}{\theta}\Big) - \alpha C(t)\phi(\bar{x},t)$$

The adjoint matrix A^* is the same as the regular matrix A as it is Hermitian, due to its symmetry and realness. We can put in the values of ϕ in the linear-elastic equation as we would for the regular system. The non-linear adjoint operator f^* involves the Fréchet derivative of the reaction term, which also depends on the values of $N(\bar{x},t)$. The discretised initial condition for the adjoint equation at time $t_1 =: m\cdot\Delta t$ is defined as

$$\boldsymbol{\phi}^m = \frac{\partial\mathfrak{F}(\mathbf{P})}{\partial\mathbf{N}^m} = -2\mathfrak{f}(\mathbf{P}) = -2\big(\mathbf{N}_{data}(t_1) - \mathbf{N}_{model}(t_1;\mathbf{P}^{(k)})\big)$$

We need to find the adjoint state variable at time t_0, which can be done by evaluating the adjoint equation backwards. Note that all intermediary values of $\mathbf{N}^m$ have to be stored for this, which requires ample working memory. Using this adjoint vector, the gradient of the objective can be found as follows

$$\frac{d\mathfrak{F}(\mathbf{P})}{d\mathbf{P}} = \frac{\partial\mathfrak{F}(\mathbf{P})}{\partial\mathbf{N}} + \Delta t\sum_{i=0}^{m}(\boldsymbol{\phi}^i)^\top\Big(\frac{\partial A(\mathbf{N}^i)}{\partial\mathbf{P}}\mathbf{N}^i + \frac{\partial f(\mathbf{N}^i)}{\partial\mathbf{P}}\mathbf{N}^i\Big)$$

As in the other three methods, the derivatives of the time operator with respect to the parameter vector are already partially computed in the time-stepping, so this does not require much extra time. The AS method only gives the gradient, so the step-size is yet to be determined. Line search can find a local minimum along the line of the gradient, but for each attempted parameter vector we need to evaluate the whole model from time t_0 to t_1. This takes too much time for our purposes, and hence we attempt to extract more information of our systems Hessian using the L-BFGS method [45, 46]. With this we iteratively get a better approximation of the inverse Hessian $(H^{(k)})^{-1}$ along with each attempted parameter vector. It also only needs to store a set amount m of vectors instead of the whole matrix. The initial guess and the updates are given by the following algorithm.

$$s_k = \mathbf{P}^{(k+1)} - \mathbf{P}^{(k)}, \quad y_k = \nabla\mathfrak{F}(\mathbf{P}^{(k+1)}) - \nabla\mathfrak{F}(\mathbf{P}^{(k)}), \quad \rho_k = \frac{1}{y_k^\top s_k}$$

$$H_0^{(0)} = 20\Big|\Big|\nabla\mathfrak{F}(\mathbf{P}^{(0)})\Big|\Big|_2 ||\mathbf{P}^{(0)}||_2^{-1} I_{n^3}$$

Algorithm 5 L-BFGS Method

```
1: $q = \nabla\mathfrak{F}(\mathbf{P}^{(k)})$
2: For $i = k-1,\ldots,k-m$:
3:     $\alpha_i = \rho_i s_i^\top q$
4:     $q = q - \alpha_i y_i$
5: end For
6: $\gamma_k = \frac{s_{k-1}^\top y_{k-1}}{y_{k-1}^\top y_{k-1}}$
7: $H_k^0 = \gamma_k^{-1} I_{n^3}$
8: $z = (H_k^0)^{-1} q$
9: For $i = k-m,\ldots,k-1$
10:     $\beta_i = \rho_i y_i^\top z$
11:     $z = z + s_i(\alpha_i - \beta_i)$
12: end For
13: $s_k = -z$
14: Return $s_k$
```

The initial guess of the Hessian is one such that the step size is 5% of the length of the parameter vector. This step is not too important for the remainder of the algorithm, as long as the matrix H_0^0 is SPD. The resulting step s_k of the algorithm is an approximation of the Gauss Newton step $s_k = -(H_k)^{-1}\nabla\mathfrak{F}(\mathbf{P}^{(k)})$. The amount of vectors m we need to store for accurate results is quite low, the value $m = 10$ is often used and works well. In line 8 a system needs to be solved, but the first Hessian approximation is diagonal so this only involves element-wise division. Note that for H_k to remain SPD we need each value of $s_i^\top y_i$ to be non-negative. This is not guaranteed due to the probable non-convexity of our problem and lack of line search, so if this value is negative the rank one update corresponding to this index is skipped.

Another option for determining the step size in Adjoint State methods is by changing to a second order Adjoint State method. For this one adds two more adjoint variables, and each optimisation step the model needs to be evaluated forwards and backwards one additional time with these new variables. Due to the additional memory requirements and time limitations this was not implemented, but for problems where Jacobians are hard to estimate this is likely one of the best options.

6.1.2. Approximating Jacobians

The Jacobian of our model is needed for the LM, TRF and DL method. A naive calculation of this would be to vary one of the parameters slightly, run the model from t_0 to t_1 and check the difference in the results. However, with more parameters to tune than the amount of pixels this is not doable; for a system of size $100 \times 100 \times 100$ we would need 10^6 model runs to find the Jacobian, plus this would need to be done for several iterations.

Instead we approach the calculation of the Jacobian in a more analytic way. The calculation of the amount of tumour cells $\mathbf{N}^{m+1}$ involves both the parameter vector $\mathbf{P}$ and the amount of tumour cells at the previous time-step $\mathbf{N}^m$ (for one-step methods). However, this previous time-step is also dependent on the parameters, and the step before. Let $\mathbf{N}^{m+1} = \mathscr{S}(\mathbf{N}^m, \mathbf{P})$ represent the scheme we use to find the value at the next time-step, then its derivative with respect to $\mathbf{P}$ is as follows.

$$\frac{d\mathbf{N}^{m+1}}{d\mathbf{P}} = \frac{\partial\mathscr{S}(\mathbf{N}^m,\mathbf{P})}{\partial\mathbf{P}} + \frac{\partial\mathscr{S}(\mathbf{N}^m,\mathbf{P})}{\partial\mathbf{N}^m}\cdot\frac{d\mathbf{N}^m}{d\mathbf{P}}$$

The input data $\mathbf{N}^0$ is independent of the parameters, i.e. $\frac{d\mathbf{N}^0}{d\mathbf{P}} = 0$. There is thus a possibility of calculating the Jacobian of the model along with the time-stepping, given that the derivatives here are simple to compute. To find the Jacobian for the RK4 method we first derive the one for FE, whose scheme will be denoted as $\mathscr{FE}(\mathbf{N}^m, \mathbf{P})$. The following three derivatives are found for the three optimisation parameters $D_0, \mathbf{k}, \alpha$.

$$\mathbf{N}^{m+1} = \mathscr{FE}(\mathbf{N}^m, \mathbf{P}) = \mathbf{N}^m + \Delta t\Big(A(\mathbf{N}^m)\mathbf{N}^m + \mathbf{k}\mathbf{N}^m\Big(1 - \frac{\mathbf{N}^m}{\theta}\Big) - \mathbf{C}^m\mathbf{N}^m\Big)$$

$$\frac{d\mathbf{N}^{m+1}}{dD_0} = \Delta t\Big(\frac{A(\mathbf{N}^m)}{D_0}\mathbf{N}^m + \Big(I + A(\mathbf{N}^m) + \mathbf{k}\big(1 - \frac{2\mathbf{N}^m}{\theta}\big) - \alpha\mathbf{C}^m\Big)\frac{d\mathbf{N}^m}{dD_0} + \frac{\partial A(\mathbf{N}^m)}{\partial \mathbf{N}^m}\mathbf{N}^m\frac{d\mathbf{N}^m}{dD_0}\Big)$$

$$\frac{d\mathbf{N}^{m+1}}{d\mathbf{k}} = \Delta t\Big(\mathbf{N}^m\Big(1 - \frac{\mathbf{N}^m}{\theta}\Big) + \Big(I + A(\mathbf{N}^m) + \mathbf{k}\big(1 - \frac{2\mathbf{N}^m}{\theta}\big) - \alpha\mathbf{C}^m\Big)\frac{d\mathbf{N}^m}{d\mathbf{k}} + \frac{\partial A(\mathbf{N}^m)}{\partial \mathbf{N}^m}\mathbf{N}^m\frac{d\mathbf{N}^m}{d\mathbf{k}}\Big)$$

$$\frac{d\mathbf{N}^{m+1}}{d\alpha} = \Delta t\Big(-\mathbf{C}^m\mathbf{N}^m + \Big(I + A(\mathbf{N}^m) + \mathbf{k}\big(1 - \frac{2\mathbf{N}^m}{\theta}\big) - \alpha\mathbf{C}^m\Big)\frac{d\mathbf{N}^m}{d\alpha} + \frac{\partial A(\mathbf{N}^m)}{\partial \mathbf{N}^m}\mathbf{N}^m\frac{d\mathbf{N}^m}{d\alpha}\Big)$$

Note that the sums of matrix and vectors here actually represent the matrix summed with a matrix containing the vector on its main diagonal, for conciseness. The first term in each derivative is equal to the contribution of the diffusion, reaction and chemo terms divided by their respective parameter, as they are all linearly dependent on them. This means that no additional calculations are needed for these terms. The second term in each derivative has the same pre-factor, so this only needs to be calculated once. The third and last term is difficult, but we saw in the GA method that the derivative of the matrix A with respect to $\mathbf{N}^m$ is negligibly small (see Equation 5.2).

Another apparent difficulty is the fact that the derivative with respect to $\mathbf{k}$ is a $n_i^3 \times n_i^3$ matrix, which needs to be multiplied with another $n_i^3 \times n_i^3$ matrix in each iteration. Luckily the matrix is sparse, with only 1 non-zero diagonal after the first iteration. The matrix it is multiplied with has 7 non-zero diagonals, so the amount of diagonals increases each iteration. The influence of the parameter $\mathbf{k}_{i,j,k}$ in a certain cell 'spreads' one step in the $x-$, $y-$ and $z-$direction each iteration. With approximately 100 cells in each direction and fewer time-steps than that, the result is that the matrix is still mostly filled with zeroes. For the implementation of this, the Jacobian values of the Jacobian with an absolute value smaller than 10^{-10} were neglected in all intermediate steps; this greatly decreases the memory-leak due to numerical precision. Moreover, the influence of these cells are negligible given that the amount of tumour cells per voxel is generally smaller than 10^{10}.

We tested this way of calculating the Jacobian on a small $12 \times 12 \times 12$ system with $t_1 = 64$, such that we can still find the 'real' Jacobian in acceptable time. It was found that the integrated Jacobian method has a mean relative error of only $3 \cdot 10^{-5}$. This is a more than acceptable result, and we should be able to find optimal parameters in this manner. Note that the order of the error is the same as the order of the neglected term. The time needed per iteration of the FE method increased by a factor of 1.047, and for a larger system with $n_i = 65$ this factor in to 1.203 (both averaged over 10 runs).

The calculation of the Jacobian with a model using RK4 for temporal integration is similar, and makes use of the chain rule. We can express it as follows.

$$\mathbf{N}^{m+1} = \mathbf{N}^m + \frac{\Delta t}{6}\big(\mathscr{FE}(\mathbf{N}^m) + 2\mathscr{FE}(\mathbf{N}^m + {}^{\Delta t}\!/_2 k_1) + 2\mathscr{FE}(\mathbf{N}^m + {}^{\Delta t}\!/_2 k_2) + \mathscr{FE}(\mathbf{N}^m + \Delta t k_3)\big)$$

$$\begin{aligned}\frac{d\mathbf{N}^{m+1}}{d\mathbf{P}} = \Bigg(&\frac{d\mathscr{F}\mathscr{E}(\mathbf{N}^{m}}{d\mathbf{P}} + 2\frac{d\mathscr{F}\mathscr{E}(\mathbf{N}^{m} + {}^{\Delta t}/_{2}k_1)}{d\mathbf{P}} \cdot \frac{1}{2}\frac{d\mathscr{F}\mathscr{E}(\mathbf{N}^{m})}{d\mathbf{P}} \\ &+ 2\frac{d\mathscr{F}\mathscr{E}(\mathbf{N}^{m} + {}^{\Delta t}/_{2}k_2)}{d\mathbf{P}} \cdot \frac{1}{2}\frac{d\mathscr{F}\mathscr{E}(\mathbf{N}^{m} + {}^{\Delta t}/_{2}k_1)}{d\mathbf{P}} \cdot \frac{1}{2}\frac{d\mathscr{F}\mathscr{E}(\mathbf{N}^{m})}{d\mathbf{P}} \\ &+ \frac{d\mathscr{F}\mathscr{E}(\mathbf{N}^{m} + \Delta t k_3)}{d\mathbf{P}} \cdot \frac{d\mathscr{F}\mathscr{E}(\mathbf{N}^{m} + {}^{\Delta t}/_{2}k_2)}{d\mathbf{P}} \cdot \frac{1}{2}\frac{d\mathscr{F}\mathscr{E}(\mathbf{N}^{m} + {}^{\Delta t}/_{2}k_1)}{d\mathbf{P}} \cdot \frac{1}{2}\frac{d\mathscr{F}\mathscr{E}(\mathbf{N}^{m})}{d\mathbf{P}}\Bigg)\end{aligned}$$

This involves a lot more steps than the just the FE method. One issue here could be the time needed to compute this all, and another that the sparsity of the Jacobian disappears faster. Hence we test if the Jacobian as calculated by the FE method while using the RK4 for temporal integration is accurate enough. For the same small system as before we found a relative mean error of $5 \cdot 10^{-2}$ in comparison to the real one. This might be good enough to find optimal parameters, which we will see from tests later on. When using the RK4 Jacobian method the increase in time per iteration is a factor of 1.068, while the FE Jacobian method only has a factor of 1.015. After enlarging the system to a size of $n_i = 64$ the factors change to 1.886 and 1.036 respectively. These factors were again averaged over 10 runs.

6.1.3. Regularisation

To be able to uniquely solve this optimisation problem, we have to adjust the objective with a regularisation term. The first and most common way of doing this is with Tikhonov regularisation, where our objective of equation 3.10 changes to

$$\min_{P} ||\mathbf{N}_{data}(t_1) - \mathbf{N}_{model}(t_1;\mathbf{P})||_2^2 + ||\Gamma\mathbf{P}_{sub}||_2^2$$

The objective is now quadratic in a subset $\mathbf{P}_{sub}$ of the parameters $\mathbf{P}$, so in a pool of optimal solutions it will pick the one with the smallest L^2–norm. There are at least $n+2$ parameters we want to optimise, i.e. $\mathbf{k}, D_0, \alpha$, so at least two parameters need to be included in P_0. A subsequent problem is then that the weight hyper-parameter Γ needs to be tuned; in [1] different values of the hyperparameter greatly influenced results. We can attempt to tackle this problem with cross-validation tests on large (simulated) data-sets, or by manual tweaking.

This regularisation technique can be extended to general norms instead of the L^2-norm used above by setting the objective as follows

$$\min_{P} ||\mathbf{N}_{data}(t_1) - \mathbf{N}_{model}(t_1;\mathbf{P})||_2^2 + ||\Gamma R\mathbf{P}_{sub}||_2^2$$

With a clever choice of R we can minimise the total variation of $\mathbf{P}_0 := \mathbf{k}$, which favours looking for solutions with $\mathbf{k}$ varying as little as possible. Similarly choosing R to represent the Laplacian we will look at smooth solutions only, as sudden jumps or discontinuities are not favourable to this objective. These choices of matrices are actually respectively equal to

$$R_{TV} = \begin{bmatrix} U_{n_x} \otimes I_{n_y} \otimes I_{n_z} \\ I_{n_x} \otimes U_{n_y} \otimes I_{n_z} \\ I_{n_x} \otimes I_{n_y} \otimes U_{n_z} \end{bmatrix}$$

$$R_{\Delta} = T_{n_x} \otimes I_{n_y} \otimes I_{n_z} + I_{n_x} \otimes T_{n_y} \otimes I_{n_z} + I_{n_x} \otimes I_{n_y} \otimes T_{n_z}$$

The matrices U_{n_i} and T_{n_i} are defined as in Chapter 3. We could also define the regularisation with matrix norms $||\cdot||_{R^\top R}$, but this hides the 'action' of the regularisation operator more. Regularisation always introduces bias to our model, so we need to be careful to not jump to conclusions on test data; Laplacian regularisation on a data-set with a smoothly generated $\mathbf{k}$ will naturally yield good results. When using TV and Laplacian regularisation on $\mathbf{k}$, it is wise to also apply Tikhonov regularisation on D_0 and α to keep them from diverging accidentally.

In order to implement regularisation on the non-linear optimisation methods we describe the regularisation term in the objective function with $\mathscr{R}(\mathbf{P})$. In the LM method we differentiate the objective function of the next iterate with respect to the step s, so we also need to differentiate the regularisation term. We find the following identity.

$$\mathscr{R}(\mathbf{P}+s) = ||\Gamma R(\mathbf{P}_{sub}+s_{sub})||_2^2 = (\Gamma R\mathbf{P}_{sub})^\top \Gamma R\mathbf{P} + 2(\Gamma R\mathbf{P}_{sub})^\top \Gamma R s_{sub} + (\Gamma R s_{sub})^\top \Gamma R s_{sub}$$

$$\implies \frac{\partial \mathscr{R}(\mathbf{P}+s)}{\partial s} = 2(\Gamma R)^\top \Gamma R\mathbf{P}_{sub} + 2(\Gamma R)^\top \Gamma R s_{sub} =: 2\mathfrak{R}_0 + 2\mathfrak{R}_1 s_{sub}$$

Thus we see that we can simply modify the LM equation 6.1 to the next one.

$$\left(J^\top J + \lambda_d I_n + \mathfrak{R}_1\right) s_k = J^\top \mathfrak{f}(\mathbf{P}^{(k)}) - \mathfrak{R}_0$$

For the TRF and DL method we have to modify the quadratic approximation $\psi_k(s)$ to include the regularisation terms. Since the matrix H_k and the vector g_k correspond to the left- and right-hand side of the LM equation respectively, the change we need to make to them is similar. As we don't differentiate in this setting we need to multiply the terms with 2, and the result is as follows.

$$g_k = -2J^\top \mathfrak{f}(\mathbf{P}^{(k)}) - 2\mathfrak{R}_0, \quad H_k = 2J^\top J + 2\mathfrak{R}_1$$

For the adjoint state method we insert the regularisation terms in the L-BFGS part. In algorithm 5, we add $\mathfrak{R}_1$ to the initial Hessian estimate in line 7, which means that we need to solve a system in line 8. This is done with CG(d) and should not take many iterations to solve. We also update the definition of y_k to include $\mathfrak{R}_0$ in the gradients. Note that the regularisation term in the inverse Hessian vanishes the more terms there are included in the L-BFGS procedure.

6.2. Parameter Analysis

We want to avoid having to try each combination of parameter estimation method, regularisation method and hyperparameters, as this will take too much time. Hence we first pick the appropriate regulariser which works best on a small problem for one of the parameter estimation methods. After this large-scale tests are performed with all the parameter estimation methods, and their results are evaluated.

In the TRF and DL method the parameters used were $\mu = 0.25, \eta = 0.75, \gamma_0 = 0.0625, \gamma_1 = 0.5, \gamma_2 = 1$ just like the original authors in [38]. Additionally, in TRF the parameters all have a lower bound of 0 for physical relevance, D_0 and α have an upper bound of 1 and k of 0.02 to avoid divergence. The proliferation rate has a low upper bound for physical relevance as well; a tumour in real life will certainly not grow faster than 2% a day, and likely much slower even. In DL we construct the matrices with an upper bound of 0.005 on k, but note that this is not an actual upper bound. We can have values of k higher than this upper bound, so by putting it lower than the actual value we hope to achieve physically relevant values. The simulated data set we use is the same as in Section 5.2.

6.2.1. Choice of Regulariser

The three regularisers and their hyperparameters will be compared on a small problem with $n_i = 33$, $dt = 1$, $t_1 = 64$, $t_2 = 128$ and $P = 32$. The 'real' parameters are the same as in the previous sections, and the initial parameter combination is picked as follows: k is taken to be uniformly spread between 0 and 0.02 on all voxels, D_0 is taken from an exponential distribution with $\lambda = 0.006$, and α is taken from an exponential distribution with $\lambda = 0.23$. The dog-leg method is picked to try these regularisers out, as this method seemed to be rather quick in testing. The upper semi-bounds on the parameters also enforce stability of RK4. The maximum allowed timestep for FE according to Equation 5.1 is 1.95 when the parameters are at their highest, and RK4 is stable up to $\sqrt{2}$ times that maximum. The parameter estimation method is stopped if the normalized score (i.e. square root of the score divided by the norm of $N(t_1)$) is less than $\epsilon = 10^{-5}$, if there isn't improvement in 5 iterations or if the process takes more than 30 minutes.

In the following table we show the results of this test. For each combination of regulariser and hyperparameter we calculated the error in the Frobenius norm of the predicted number of tumour cells at t_2 and the proliferation rate k as $||N_{data}(t_2) - N_{model}(t_2)||_F / ||N_{data}(t_2)||_F$, $||k_{data} - k_{model}||_F / ||k_{data}||_F$. We also calculated the mean relative difference of their values element-wise. This was only done where the 'real' values are larger than 10^3 for N and 10^{-5} for k to avoid exploding values in areas of no importance. Furthermore, we calculate the relative difference in D_0 and α as $(D_{0,model} - D_{0,data})/D_{0,data}$, $(\alpha_{model} - \alpha_{data})/\alpha_{data}$.

Table 6.1: Regulariser test results for the dog-leg method.

			$N(t_1)$			$N(t_2)$			k		D_0	α
Reg	Par	Score	Err	Iters	Err	Diff	Pear	Err	Diff	Pear	Err	Err
Tik	10	$<\epsilon$	$<\epsilon$	9	1.2e-2	5.0e-2	1.0	4.3	3.1	0.27	0.30	-1.0e-2
	10^2	$<\epsilon$	$<\epsilon$	9	1.2e-2	5.2e-2	1.0	3.9	1.9	0.28	0.32	-9.9e-3
	10^3	$<\epsilon$	$<\epsilon$	10	1.3e-2	5.3e-2	1.0	3.4	1.2	2.1	0.31	-9.0e-3
	10^4	$<\epsilon$	$<\epsilon$	10	1.2e-2	5.0e-2	1.0	3.3	1.1	0.32	0.32	-9.6e-3
	10^5	2.8e-5	$<\epsilon$	35	1.2e-2	1.1e-2	1.0	0.55	0.22	0.95	4.4e-2	-1.0e-2
	10^6	2.7e-4	3.4e-5	40	9.4e-3	2.4e-2	1.0	0.45	0.19	0.97	0.26	-1.3e-2
TV	10	$<\epsilon$	$<\epsilon$	10	1.4e-2	4.3e-2	1.0	7.3	7.1	0.13	0.23	-7.7e-3
	10^2	$<\epsilon$	$<\epsilon$	9	1.5e-2	5.7e-2	1.0	5.4	11	0.25	0.37	-6.6e-3
	10^3	$<\epsilon$	$<\epsilon$	9	1.5e-2	7.4e-2	1.0	7.7	15	0.18	0.59	-6.0e-3
	10^4	$<\epsilon$	$<\epsilon$	10	1.4e-2	4.6e-2	1.0	4.0	10	0.37	0.28	-7.9e-3
	10^5	1.5e-5	$<\epsilon$	35	1.4e-2	7.7e-3	1.0	0.73	1.7	0.95	-1.3e-2	-7.3e-3
	10^6	1.5e-4	$<\epsilon$	35	1.3e-2	1.3e-2	1.0	0.78	1.8	0.95	4.3e-2	-9.2e-3
Lap	10	$<\epsilon$	$<\epsilon$	11	1.5e-2	5.7e-2	1.0	12	14	-4.4e-3	0.36	-6.3e-3
	10^2	$<\epsilon$	$<\epsilon$	10	1.1e-2	7.4e-2	1.0	11	18	9.7e-4	0.62	-1.2e-2
	10^3	$<\epsilon$	$<\epsilon$	9	1.4e-2	4.4e-2	1.0	5.6	7.9	5.2e-2	0.24	-8.2e-3
	10^4	$<\epsilon$	$<\epsilon$	9	1.2e-2	3.8e-2	1.0	5.1	6.6	5.2e-2	0.22	-1.0e-2
	10^5	1.4e-5	$<\epsilon$	33	1.6e-2	7.3e-3	1.0	0.89	1.3	0.92	-1.0e-2	-5.7e-3
	10^6	1.4e-4	$<\epsilon$	30	1.2e-2	7.7e-3	1.0	0.91	1.3	0.91	7.5e-3	-9.6e-3

A first observation is that the size of the error of $N(t_2)$ does not vary that much between the methods, while there are slight deviations in the relative element-wise difference. The explanation for this is that the simulations with low relative difference have their error located mainly on the small elements, which are not taken into account for this metric. The Pearson correlation coefficient is as high as can be for $N(t_2)$, which implies that the predicted outcome is equivalent to a scaled off-set version of the real outcome. In related research the Concordance correlation coefficient was used instead, which takes into account that the means of the compared quantities can be different. This was not done in this work, as for a qualitative prediction the exact values are not the biggest concern. Although there are vastly different outcomes for the found parameter values, the inversion method is always able to fit the model to t_1, which in turn also results in a good estimation at t_2.

In all three regularisation methods, we see that a high valued hyperparameter results in more accurate estimations of k. For low values, the proliferation rate is factors too high. A good explanation for this is that the proliferation is overshadowed by the much more impactful drug efficacy α, so higher values do not generally result in more tumour cells. The proliferation is most important at the boundary of the tumour, where the diffusion rate D_0 is also very important. We see that the more accurate (i.e. lower) k gets, the better D_0 gets as well, as they do not need to counteract each other anymore. The Laplacian regulariser is clearly the worst at finding good values for k, which probably has to do with the fact that our choice of k does not have a second derivative close to zero.

When the hyperparameter was set to 10^5 or 10^6, the methods were not able to find a solution with a low enough score in time. This is due to a combination of two factors. First, the regularisation might be too harsh, and exploring in directions which would be good for the error at $N(t_2)$ is made impossible by the large regularisation term. Second, the score cut-off criterion also takes the regularisation term into account. The second column under $N(t_2)$ shows that just the error is low enough in all-but-one case, but that the enormous regularisation term forces the parameter tuning to continue.

The all-round best results are found with Tik and TV with high hyperparameters, with Lap lacking behind slightly. As the proliferation rate k in reality is likely piece-wise continuous among tissue types, there is hope that TV gives better results on real data. Hence the dog-leg method will be executed using TV regularisation with hyperparameter 10^5.

Due to the expected size of the variables (known from literature) it might be handy to have the hyperparameters be different for the three parameters. As α is expected to be around 10 times as big as $k_{\max}$, its regularisation parameter should be 10 times smaller. This was tested on our best found regulariser. The results were marginally better, but it is negligible in practice. If non-satisfactory results are found on real data it might be worth playing around with this some more.

We also tried this best-found regulariser on LM, TRF, and AS, and results turned out to be a bit different. The same test was repeated for these three methods, the results of which can be found in Appendix A.2. The conclusions are that we run LM with Tikhonov regularisation and hyperparameter 10^6, TRF with Tikhonov regularisation and hyperparameter 10^5, and AS with TV regularisation and hyperparameter 10^5. Regularisation was found to not have such a big effect on AS due to the L-BFGS implementation, so its choice is mainly based on the assumptions.

6.2.2. Large-scale testing

With a properly tuned regulariser and hyperparameter combination we are now ready to compare the parameter estimation methods. We return to the test scenario with $n_i = 65$ with a maximum simulation time of 4 hours, leaving the other settings unchanged from the last section. In the following figures we see the reduction in error of the amount of tumour cells at t_1, as well as the score according to the objective functions over the simulation time for all four methods.

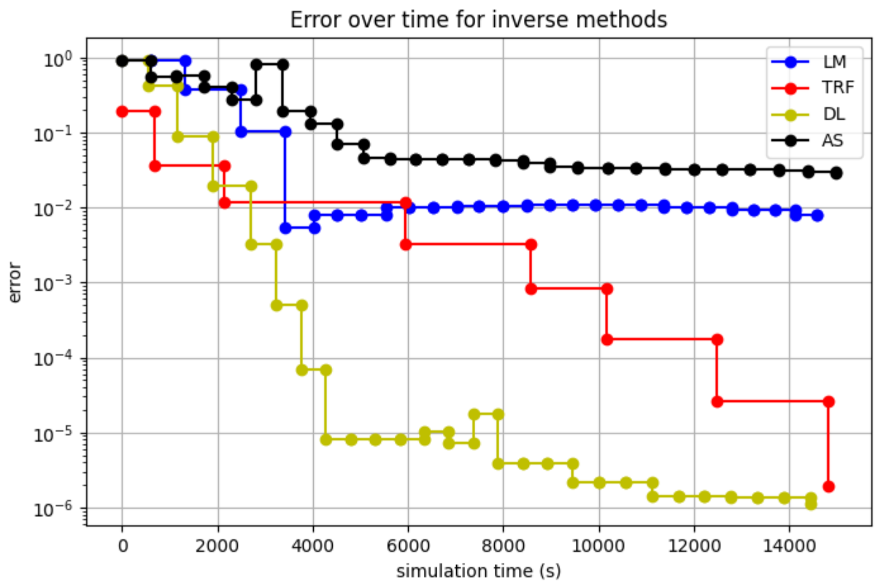

Figure 6.1: Normalised error reduction of the inversion methods over the simulation time.

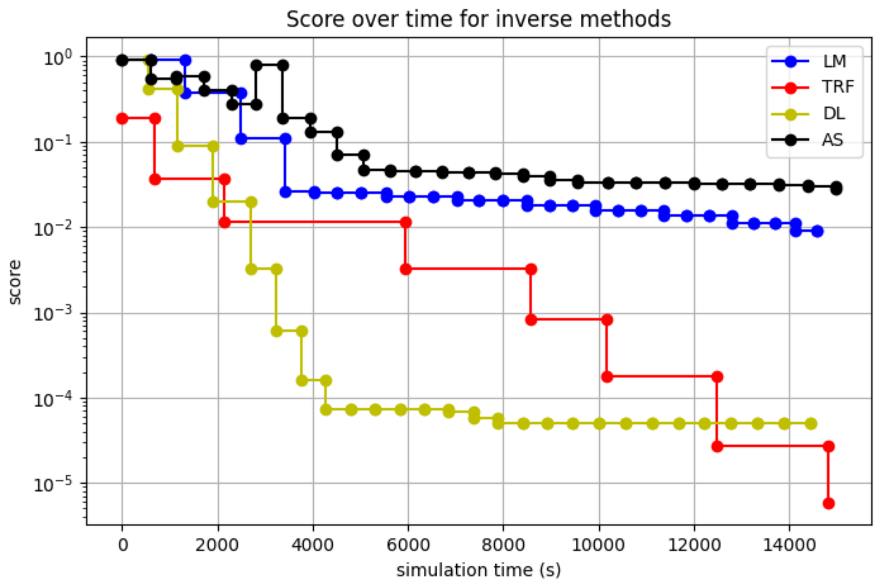

Figure 6.2: Objective function score of the inversion methods over the simulation time.

The first observation we make is that the four methods start off almost equally strong. The two trust region methods succeed in continuously lowering the score and the error, while the other two methods seem to get stuck. After a while LM has a raising error while the score drops a bit; this has to do with the balance between reducing the regularisation term and the actual error. The fact that AS performs the worst is not too strange, as it is the only Quasi-Newton method amongst real Newton methods. Furthermore, the steps it takes are just estimates of finding the Gauss-Newton point, while the others in addition to this also incorporate the steepest descent direction. We see this reflected in the fact that the score of the AS method even increases a single time. The TRF algorithm has the slowest steps, as the CGT method generally requires ample iterations to find a satisfactory step. These steps seem to always decrease the error and score immensely though, so compared to LM and AS it is worth taking this time. The DL approximates the same step, and although the score does not decreases as much each iteration, a satisfactory solution is found way earlier. It is great to see that this method does the majority of its work within the first two hours, which suggests that with even larger problems we might attain the goal of optimising the parameters within a working day.

Next we verify how accurate the parameter estimates are, and how good the predictive power is at t_2. A table is given below with the same metrics as in the last Section for the four methods. Additionally, a figure with the total amount of tumour cells over the time domain is given for qualitative interpretation of the found parameter sets.

Table 6.2: Output of the large-scale inversion methods test.

	$N(t_1)$			$N(t_2)$			k			D_0	α
Method	Score	Err	Iters	Err	Diff	Pear	Err	Diff	Pear	Err	Err
LM	8.9e-3	7.9e-3	27	8.0e-2	4.0e-2	1.0	2.2	0.50	0.95	-4.3e-3	8.7e-2
TRF	2.4e-5	1.1e-5	11	0.19	9.5e-2	1.0	5.1	4.6	0.90	5.7e-2	0.20
DL	5.0e-5	$<\epsilon$	26	3.3e-2	1.6e-2	1.0	1.2	2.5	0.94	-2.1e-2	1.6e-2
AS	2.8e-2	2.8e-2	26	3.3e-2	0.98	1.0	4.6	6.3	2.6e-2	56	-1.3e-2

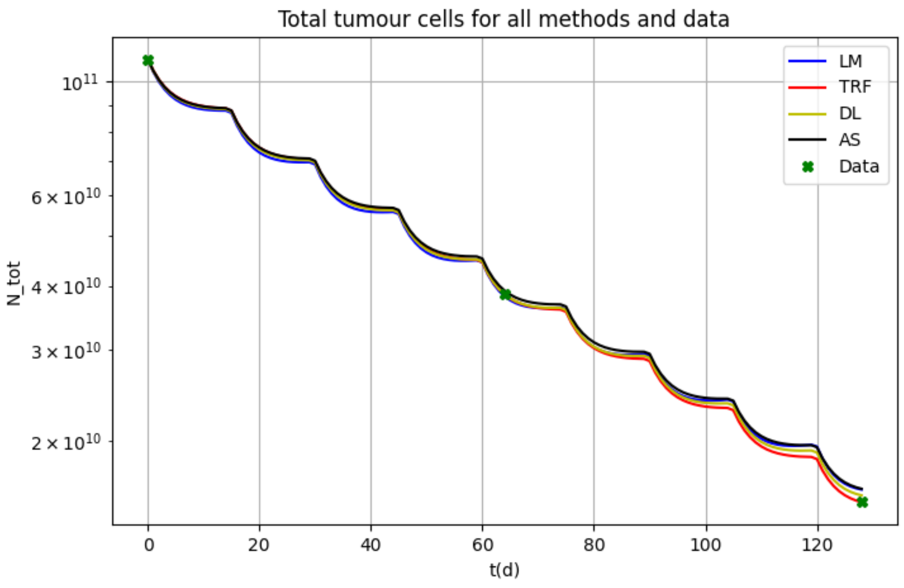

Figure 6.3: Simulations of total tumour cell growth using best-found parameters of all four methods.

In almost all metrics we find that the dog-leg method performs best. The most important of these is the performance at t_2, of which DL is best as well. Only for AS the error is as good, but this is most likely a coincidence. The fact that AS scores well in these metrics is not likely to be a attribute of the method, but just luck. Again we see that the values the three parameters need not be to close to the 'real' values, and vastly different combinations can model very similar behaviour. This is again a consequence of the parameters being able to take over each others roles of diffusing, growing and shrinking.

One possible issue in the future is the almost guaranteed over-estimation of the proliferation rate. We have verified that the model can find a satisfactory combination of parameters for more-or-less random initial parameters. Hence we can now introduce some bias by setting the initial parameters to be lower than the real ones. This might steer the model into choosing relatively smaller parameters values overall, especially when combined with the regularisation. In the next test we do this, and set the initial parameter values to be roughly half of the real ones. The results are once again shown in the next table and figure.

Table 6.3: Output of the second large-scale inversion methods test.

	$N(t_1)$			$N(t_2)$			k			D_0	α
Method	Score	Err	Iters	Err	Diff	Pear	Err	Diff	Pear	Err	Err
LM	1.1e-3	7.8e-4	28	6.0e-2	3.2e-2	1.0	1.2	0.34	-0.21	8.8e-3	-9.6e-2
TRF	$<\epsilon$	$<\epsilon$	7	5.0e-3	5.4e-3	1.0	1.5	2.5	0.60	-4.0e-3	-2.9e-2
DL	4.4e-5	$<\epsilon$	23	7.3e-3	6.6e-3	1.0	0.33	0.98	0.97	-5.8e-3	-3.2e-2
AS	5.5e-3	5.4e-3	32	2.4e-2	8.7e-2	1.0	1.8	2.4	0.11	1.9	-5.1e-2

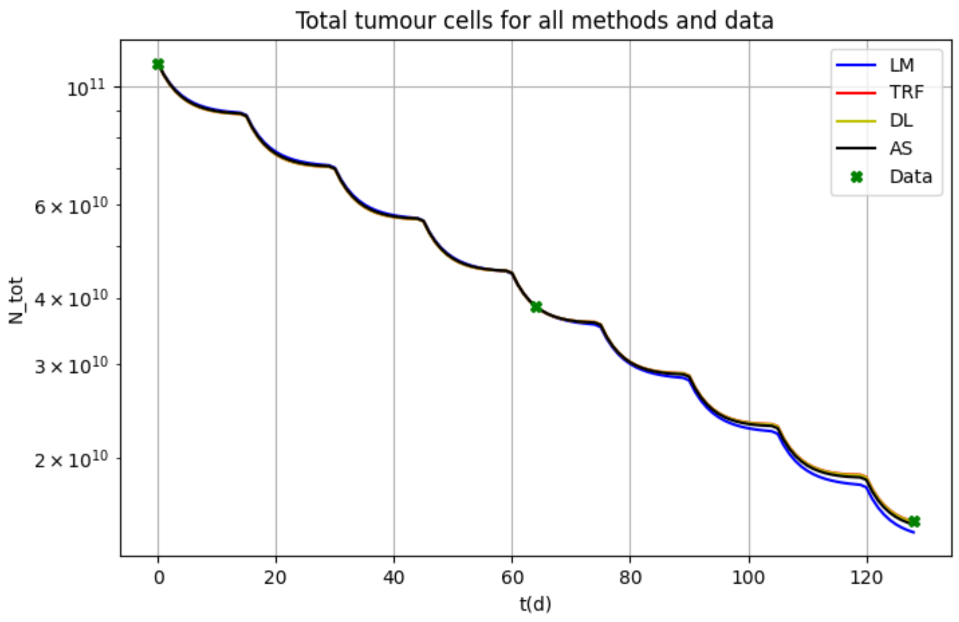

Figure 6.4: Second round of simulations of total tumour cell growth using best-found parameters of all four methods.

It is evident from this data that the change in initial parameters had its intended effect. The prediction at t_2 is now even better and makes a near-perfect estimate of the final amount of tumour cells. Out of the four methods, the trust region methods are the best, with slightly better results for DL. With this we conclude that the best way to go forward is to use Powell's dog-leg method with Total Variation Regularisation with hyperparameter $1\cdot10^5$. The next Chapter verifies if this method does not only work on simulated data, but on real data as well.

7 Patient Results

In this final content Chapter the predictive capabilities of the numerically optimised model are tested on the real data. This is structured by first giving all the relevant figures and metrics in Section 7.1. The implications and consequences of the findings are then thoroughly discussed in Section 7.2.

7.1. Predictions

7.1.1. Scaling Data

For the efficiency of the FFT's in the linear elastic sub-problem, we know that it is important that the size of each spatial dimension is not prime, and preferrably a power of two plus one. Hence we downscale the data slightly, with the downscaling factor among each dimension around 2. Because of this rather small factor we can simply linearly interpolate the data, and scale it. The scaling is done in a way that the total amount of tumour cells over the domain remains the same; each voxel value is multiplied with the same factor for this. This should still respect the original distribution of cells, and give a good representation of what the tumour looks like at a different scale. If the downscaling factor is lower than two, then information from each original voxel is used, and hence information loss will be minimal. For finding the drug concentration in the tissue with the NBVM method, we need to also resize the DCE-MRI curves. This can be done without scaling, as this data needs to be scaled such that the maximum value in this array is one anyways.

The tests on the real data will hence be performed on a system with dimensions $n_x = 81, n_y = 81, n_z = 61$; initial test were done on $97, 97, 65$, but for some patients the memory requirements became too big. The increase in total voxels makes the regularisation parameter weigh differently, so it is important to check if that has a noticeable influence on the results. In addition to this, we only use $P = 12$ processors and time-steps of approximately $dt = 1$. Specifically the time-steps were $dt = t_1/(8 \cdot P)$, such that each rank has to do eight RK4 steps. More processors would be beneficial, but although there are ample the additional memory requirements were too much. In the next table some simple data of the downscaled data is displayed.

Table 7.1: Details timings of the scans of all patients, as well the size of their tumours at those times.

	t_1	t_2	#NZ(t_0)	#NZ(t_1)	#NZ(t_2)
p1	90	169	979	65	30
p2	87	191	1843	104	99
p3	84	174	1168	662	165

Here #NZ(t_i) means the number of non-zero elements in the vector of tumour cells at time t_i. More technical details of the scans can be found in Appendix A.1 It is evident that the three patient scenarios are quite distinct, with differing initial sizes and responses. The tests in this Chapter were performed on the cluster of the BIGR group of the Erasmus Medical Center [47]. The CPUs used run on 2.9GHz with 512GB RAM.

7.1.2. Calibration Results

The first experiment perform is the calibration, as described in the previous chapters, on all three patients with their best-found hyperparameters. This uses Lap2 preconditioning, Parareal time integration, TV(10^5) regularisation and the Dog-leg method. In the next table and figures the results of this experiment are displayed.

Table 7.2: Outcome of fitting the model on real data of three patients.

		$N(t_1)$			$N(t_2)$			k		D_0	α
Patient	Score	Err	Iters	Err	Diff	Pear	Mean	Min	Max		
p1	6.6e-2	3.9e-2	115	1.2	1.1	0.48	-2.4e-3	-9.0e-2	0.64	7.2e-2	0.50
p2	3.7e-2	2.6e-2	26	1.6	1.0	1.8e-2	-4.6e-3	-0.43	0.27	2.3e-2	0.51
p3	6.4e-2	4.5e-2	44	2.4	0.99	8.3e-2	-3.0e-3	-0.19	0.60	6.4e-2	0.20

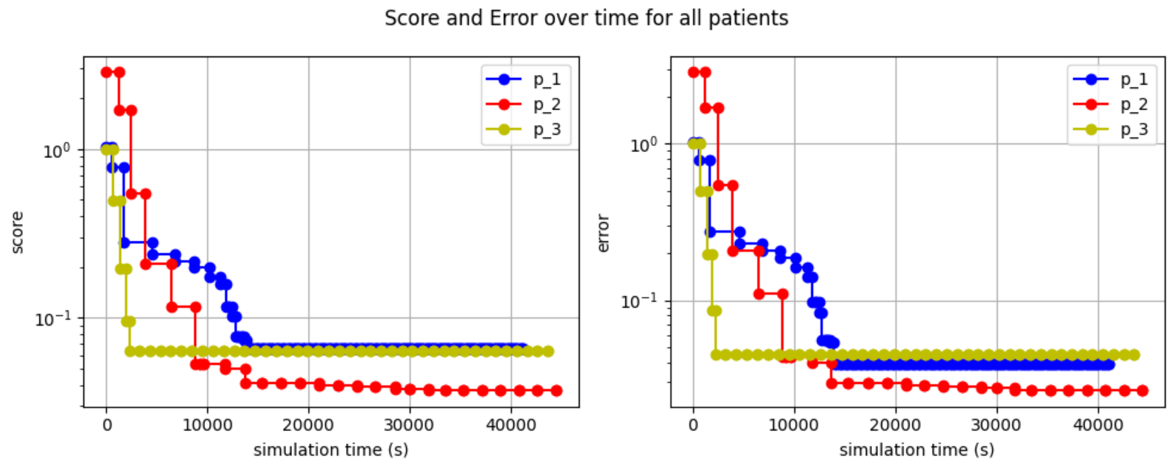

Figure 7.1: Score and error over simulation time for all three patients

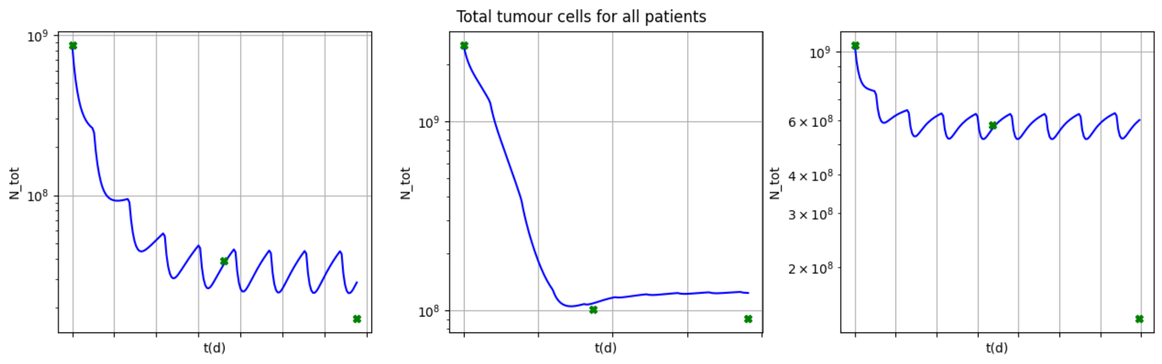

Figure 7.2: Total amount of tumour cells for all three patients between times t_0, t_1 and t_2.

The score and error at t_1 don't go under the threshold anymore, but that was to be expected. With measurement, modelling and a minimal amount of numerical errors, an error in the single-digit percentage range is more than acceptable. The difference in the amount of iterations per patient has to do with the amount of non-zero tumour cell values in the voxels at t_0 and t_1; for p2 this is more than double than the other two patients, causing the Jacobian to be less sparse and hence more computation is needed.

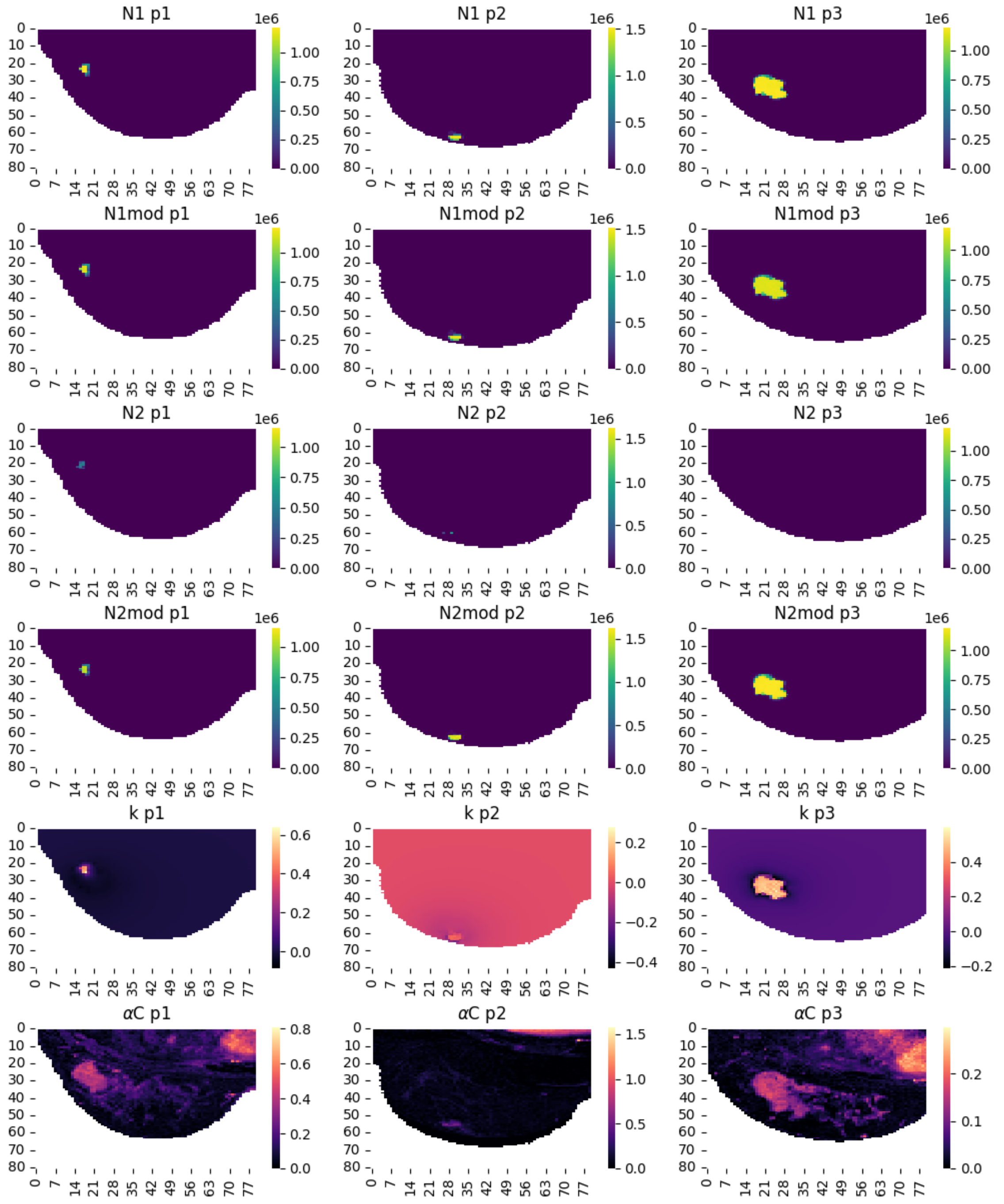

Figure 7.3: Visualisation of the prediction and calibration results on slices of all the patients.

The results at the validation point are not bad, but have a lot left to wish for. We see that the calibration seems to converge to a solution which has an equilibrium about the values at t_1, which causes the predictions at later time-points to be similar to that. This is also reflected in the rather high values for k and α. In the Figure 7.3, the measured and calculated tumour values at t_1 and t_2 are given in the most characteristic slice of the tumour, as well as the found value for k and αC. The suffix '-mod' here indicates that it is a predicted / modelled value.

For patient p1 there doesn't happen too much to the tumour shape and location after t_1, and hence the prediction seems alright. For p2 and p3, however, the tumour still shrinks quite a bit, and moves slightly within the breast as well. This is clearly reflected in the Pearson correlation coefficient at t_2. The movement could be a result of movement of the patient in between scans, but most of this should have been fixed in the registration phase of the post-processing of the MRI images. We also see that the proliferation is large where the tumour is still prevalent at t_1, and zero or slightly negative elsewhere. Physically this makes sense, as the tumour should be most active in its core. The rate at which this proliferation happens, according to our model, is too high though, causing the predicted tumour at t_2 to be too big in all three patients.

7.1.3. Changing Hyperparameters

In the next experiment we change the regularisation hyperparameter to accomodate for the higher values of the parameters than previously expected. Both a lower and a higher value, 10^4 and 10^6, are tried to see if better results are found. With a lower value, we expect the parameters to be able to take higher values, while not getting stuck due to interaction of the error and regularisation term. With higher values we expect physically more realistic values, but convergence might be harder to achieve. The stability criterion will be tighter when the regularisation is less strict, but this should not be an issue for our chosen value of dt. The results of this second experiment are shown below.

Table 7.3: Results on real data for a lower hyperparameter of 10^4.

	$N(t_1)$			$N(t_2)$			k			D_0	α
Patient	Score	Err	Iters	Err	Diff	Pear	Mean	Min	Max		
p1	1.4e-2	1.2e-2	109	1.3	1.2	0.48	-4.4e-3	-0.11	0.57	4.5e-3	0.17
p2	0.14	6.4e-2	72	1.5	1.0	1.1e-3	-5.7e-3	-9.8e-2	0.16	8.3e-3	0.56
p3	7.5e-3	5.2e-3	26	2.4	1.0	6.0e-2	-5.9e-3	-0.47	0.68	9.4e-3	0.19

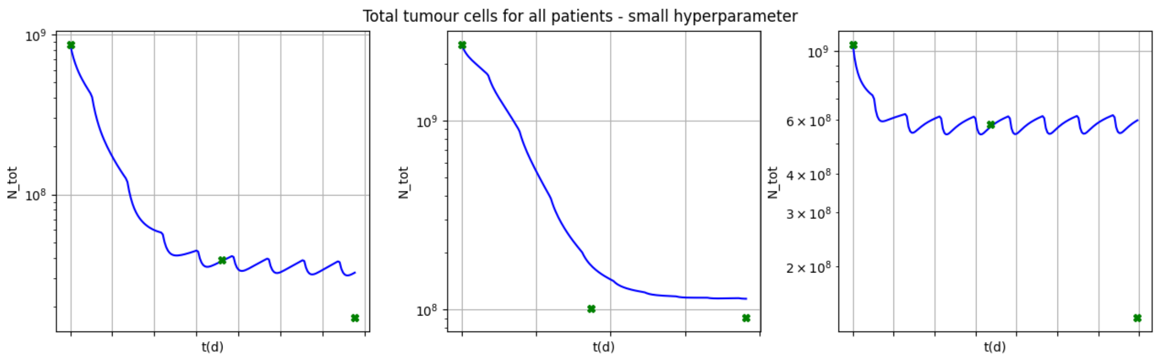

Figure 7.4: Total amount of tumour cells for all three patients between times t_0, t_1 and t_2 for a low hyperparameter (10^4).

Table 7.4: Results on real data for a higher hyperparameter of 10^6.

	$N(t_1)$			$N(t_2)$			k			D_0	α
Patient	Score	Err	Iters	Err	Diff	Pear	Mean	Min	Max		
p1	0.20	9.4e-2	141	1.1	1.2	0.50	-4.3e-3	-0.12	4.6e-2	2.9e-4	2.1e-2
p2	0.14	6.7e-2	78	2.0	1.0	1.4e-3	-3.6e-3	-0.10	0.15	1.1e-2	0.17
p3	7.2e-2	3.0e-2	139	2.1	1.0	3.7e-3	-2.7e-3	-0.12	8.3e-2	1.4e-3	6.1e-4

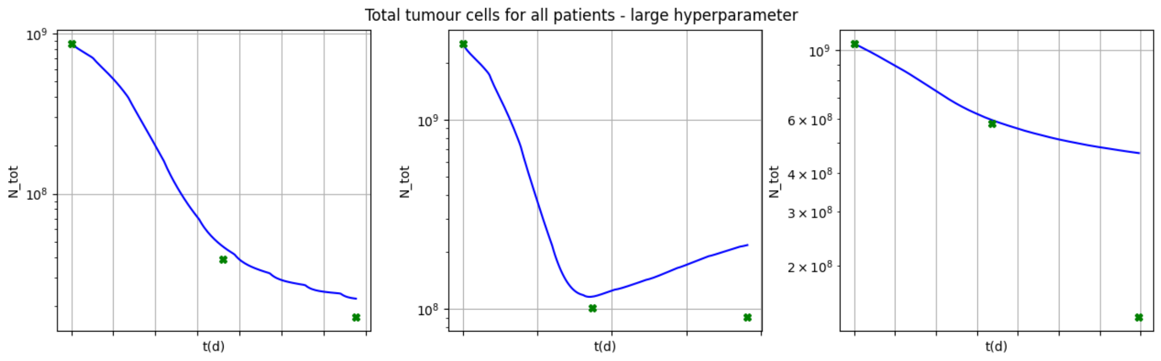

Figure 7.5: Total amount of tumour cells for all three patients between times t_0, t_1 and t_2 for a high hyperparameter (10^6).

Overall for lower hyperparameters the score and error at t_1 decrease slightly. This was expected for the score as the regularisation doesn't contribute that much anymore. Contrary to expectations, lower values for the parameters were found all over the board, especially for the proliferation. This suggests that the regular experiment got stuck in a local minimum due to the higher regularisation, instead of getting stuck due to the true parameters being larger than allowed. The found parameter-set also generates an oscillating solution about an equilibrium again, but with lower amplitude than the regular case.

With a more strict regularisation the score-value of the calibration goes up, and the error at t_1 as well. The values found for the parameters, especially k, are in a physically more realistic range however. The oscillations also disappear, giving more realistic results. For patient p2 the tumour is expected to grow quite a bit after t_1, which is due to the sharply reduced value of α.

These results suggest that different hyperparameters for the three parameters might be beneficial for the outcome. Hence, in the next experiment we test what the outcome is of using a hyperparameter of 10^6 for k, 10^5 for D_0 and 10^4 for α. The results are shown in the following table and image.

Table 7.5: Results on real data for mixed hyperparameters.

	$N(t_1)$			$N(t_2)$			k			D_0	α
Patient	Score	Err	Iters	Err	Diff	Pear	Mean	Min	Max		
p1	0.20	9.4e-2	141	1.1	1.2	0.50	-4.3e-3	-0.11	4.6e-2	1.3e-4	2.0e-2
p2	0.14	6.4e-2	72	2.0	1.0	1.4e-4	-2.9e-3	-9.8e-2	0.16	9.8e-3	0.48
p3	7.2e-2	2.9e-2	148	2.1	1.0	2.4e-5	-2.7e-3	-0.12	8.3e-2	1.4e-3	6.5e-4

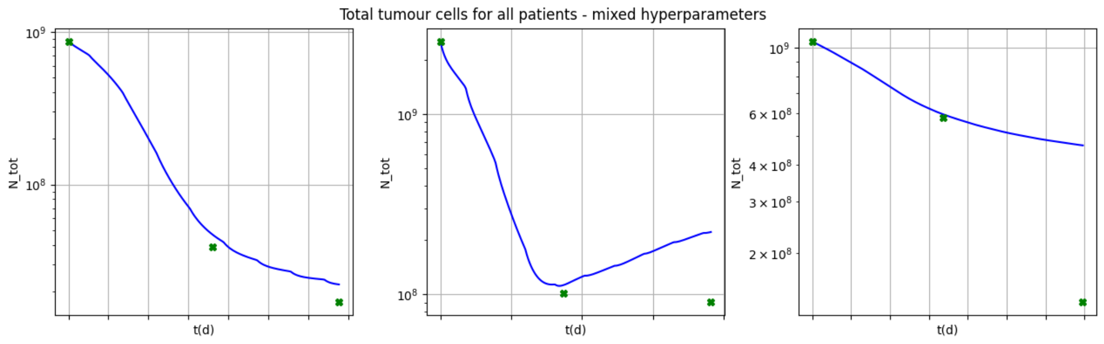

Figure 7.6: Total amount of tumour cells for all three patients between times t_0, t_1 and t_2 for mixed hyperparameters.

These results are indeed a combination of what was seen before. The values of k are in a physically realistic range, and no oscillations are observed. Predictions are still not that accurate, but with only three patients this does not necessarily indicate that the model is not suited for this problem.

7.1.4. Three-image Calibration

Lastly we check if the model can accurately fit the patient data using all three images, which gives an indication whether or not the data can be properly modelled with the dynamics of our coupled PDE. This changes the objective function and trust region slightly, now incorporating a second Jacobian term for the sensitivity of the parameters with respect to $N(t_2)$. Due to the additional size of this second Jacobian, the problem has been down-scaled further to $n_x = n_y = n_z = 61$. This is not due to speed bottlenecks, but due to memory requirements when calculutating Jacobians with parareal. This experiment gives the following output.

Table 7.6: Results on real data when calibration is done on all three scans.

	$N(t_1)$				$N(t_2)$			k			D_0	α
Patient	Score	Err	Iters	Pear	Err	Diff	Pear	Mean	Min	Max		
p1	0.42	0.48	72	0.88	0.51	1.1	0.86	-3.4e-3	-0.24	0.14	2.2e-2	4.8e-2
p2	1.5	1.9	28	0.12	1.1	0.84	0.19	4.4e-4	-7.9e-2	7.7e-2	8.5e-2	0.13
p3	0.56	0.70	48	0.78	0.75	0.52	0.66	2.1e-3	-0.29	0.40	6.2e-2	5.7e-2

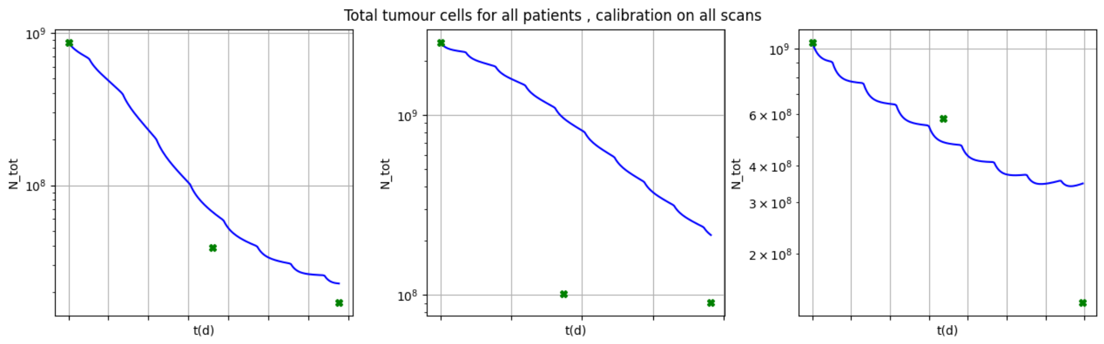

Figure 7.7: Total amount of tumour cells for all three patients between times t_0, t_1 and t_2 when calibrated on all three scans.

Note that the found parameters can differ slightly from the previous subsection due to differently sized voxels. To run the model to t_2 in each iterations is more than twice the work than running it just to t_1, as the Jacobian increases in size each time-step. Hence there weren't as many iterations done in the same amount of time, but local minima were found nonetheless.

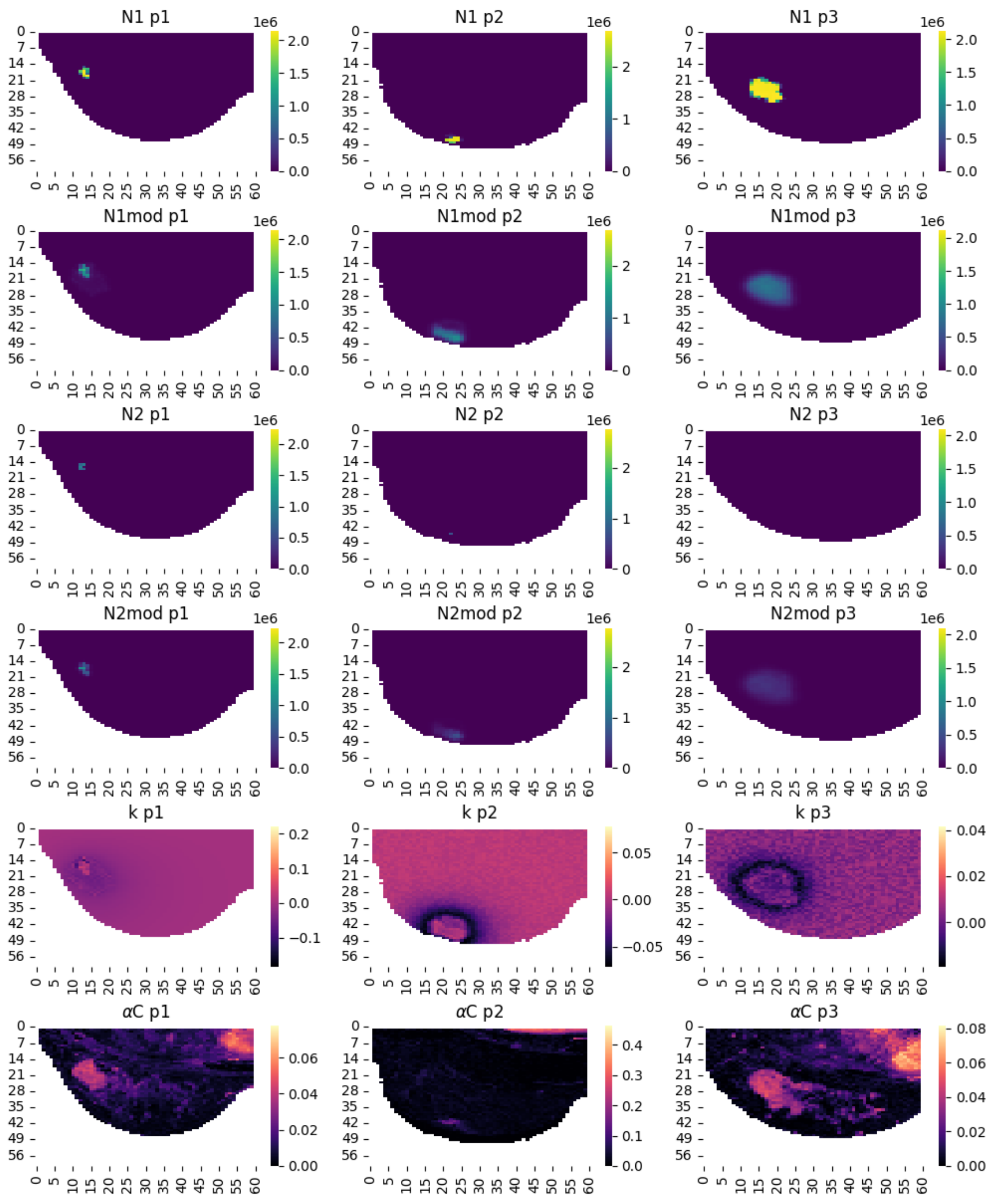

Figure 7.8: Visualisation of the prediction and calibration results on slices of all the patients when calibrated on all three scans.

For all three patients, the errors at both t_1 and t_2 are still in the double digit percentage range. The correlation between the modelled and actual tumour is significantly higher, however, suggesting that there is a problem in offset and / or scaling. It is also clear from Figure 7.8 that the solution no longer stabilizes about the scan at t_1. The poor results can be the cause of two factors: either the model does not work too well on the data, or the straight-forward dual objective optimisation implementation is not satisfactory. Both of these causes can be investigated if future research is done on the verification of the suitability of the model to the data.

7.2. Discussion

For this model to be used clinically, some points need to be investigated more. We delve a bit deeper into the data, the model and a practical use-case.

7.2.1. Data Assumptions

First of all, the results are assumed not to be affected by the downscaling. Due to the step of manual registration of the tumour boundary in the pre-processing, the detail tumour boundary is not drastically worse when other resolutions are used. Loss of fine details of the fibroglandular and adipose tissue types also should not cause worse predictions, as the region of interest is marked as tumour tissue anyways. In reality, the dead tumour cells leave behind 'empty' cells, whose elastic properties are different. Incorporating this in the form of a temporally and N-dependent tissue type (and hence shear modulus) could improve quality of the results.

While the downscaling was not necessary for the time-requirements, it was vital for the memory-requirements of the program. The size of the Jacobian is rather large, even when stored in a sparse manner. Combining this with the Parareal method forces one to store more than one Jacobian per rank, which leads to RAM requirements not feasible for standard computers. Communication with MPI of these Jacobians was implemented in both a centralised and decentralised manner. The first has all Jacobians gathered on the root node, after which the update is calculated and the new Jacobians are scattered. The second manner sends Jacobians from the ranks to the root node and updates them one-by-one. The first way requires a root node with exorbitant RAM in the scale of 100GB, while the second way requires each node to have 10GB to themselves. The speedups achieved with Parareal are likely also achievable with parallel matrix-vector multiplications, so this might be more feasible in practice. It is also suggested to use lower-level languages in future research, to optimally use hardware and to prevent encountering python-related bugs in high-performance computing.

One could also use parallelisation for searching the state space for multiple solutions at once. This could especially contribute when the model has been running for a while, as the model initially decreases quickly in error but seems to get 'stuck' after some time. More radical changes in parameters could be attempted more efficiently with this type of parallelisation, making it easier to escape local minima.

7.2.2. Model Assumptions

A first discrepancy between the model applied to the simulated and real data is the scale of the found parameters. The assumptions of the simulated tumour model were based on clinical knowledge, and how a tumour is expected to grow in real life. If the proliferation rate is 10%, a small tumour would double in size in less than eight days. Still, we calibrate to find values higher than this, and the only reason the tumour doesn't explode in size is the logistical growth shrinking it again. The counteraction of the proliferation and chemo terms makes calibration of this model extremely difficult. Regularisation effectively helps with this, but forcing the parameters to take on values that 'feel realistic' to the user inhibits the search for the 'real' solutions.

Second, we assumed that the tumour would have a Gaussian shape, in the sense that the boundary of the tumour is a smooth transition. The data received was more binary, as the voxels where the tumour was located always had the amount of tumour cells equal to the capacity θ. Due to the downscaling there was some smoothness introduced, however. With binary data, the Von Mises stress will only be non-zero at the exact location of the tumour boundary. The diffusion will not follow the expected behaviour of a linear-elastic system in that case. The binary data is likely also the cause of the large values of the calibrated parameters. To obtain the MRI image at t_1, which is also binary, the proliferation and chemo term need to be large, in order to have either 0 or θ cells in all voxels. It is not the sole reason for the mediocre results, but likely the main cause.

With knowledge that the data is practically binary in real life, one should take advantage of this data structure. One way to do this is to use level-set methods [48], which model the boundary implicitly. These methods are used often in computational fluid dynamics, but to the best of our knowledge they have not been investigated in computational oncology.

7.2.3. Practical Use

It has been shown that it is computationally feasible to calibrate the model. Even with the relatively old CPU cluster of the Erasmus hospital, the model can create a prediction in acceptable time. With better use of the parallelisation and more consistent CPU behaviour, one could definitely make predictions at an even higher resolution within a working day. To make the model more useful for the analysis of the doctors, there should be a recommendation for volumes of future admissions of chemotherapy. This should not be more difficult than one-parameter optimisation in the case of a constant dosis for each remaining round of chemotherapy. If we want to consider different doses per round or a variable amount of rounds, some combinatorial optimisation is required. With some knowledge of the admitted substances, a lot of options could be filtered away beforehand however.

The results of the calibration on all three images suggest that before clinical implementation, there should be more research on the suitability of the model. The error found when fitting on all three images is simply too high at the moment to suggest that our model is accurate. One way of testing suitability is researching if there exist solutions to the inverse problem on (simulated) datasets of multiple scans, with more sophisticated dual-objective optimisation methods. Exploring other methods, as mentioned before, also creates insights in this manner.

Apart from the amount of images used in calibration, the timing also plays a role. While this has not been considered in this work, in literature it has been discussed already, such as in [49]. With a more sophisticated timing of scans, the calibration can guarantee more accurate predictions.

8
Conclusion and Future Research

To improve the speed and accuracy of the DI-MCRD model, four possible improvement areas were identified: the linear-elastic sub-problem of the breast tissue, the temporal evolution of the modeled tumour, the estimation of personalised parameters for patients, and the consideration of deviations of ideal scenarios with real data. Each of these issues was investigated thoroughly, in order to answer the formal research question of this thesis:

How can the 3D DI-MCRD model be implemented more efficiently, in order to tune the model in a few hours and accurately predict tumour response for HER2+ breast cancer patients?

The sub-problem of linear elasticity saw vast improvements in speed when the solution method was changed to the Preconditioned Conjugate Gradient method. Using the Laplacian operator as the preconditioner gave the best results, and the resulting system was solved in a novel manner using FFT's in two dimensions and a tridiagonal solver in the third. It was found that high-order methods are needed to guarantee that the simulations, which span over a 100 days, give an accurate result. The fourth order Runge-Kutta 4 method was chosen to obtain this, which was implemented with the Parareal method. The coarse solver for this was the unconditionally stable second order Crank-Nicolson method, with which a speed-up of >7 can be found, making it faster than regular first order methods. Fourth order convergence of the error was not found due to the lack of continuity of the chemotherapy term.

The best result for finding patient parameters on simulated data was obtained with Powell's Dog-leg method in combination with Total Variation Regularisation. A novel way of estimating the Jacobian analytically was used, which was the sole reason that regular Newton-methods can be used instead of Quasi-Newton methods. Near-optimal solutions were found on a simulated dataset, and it was found that the found parameters are very sensitive with respect to the regularisation hyperparameter. With the combination of all these improvements, the goal of tuning the model and generating an accurate prediction within a working day has been attained successfully.

The model was also tested on a small real data set of three patients. While it is possible to calibrate the model such that the known MRI scan could be recreated with single-digit percentage error, the prediction of the validation MRI scan is not sufficiently accurate for clinical usage. One of the reasons for this is the counteraction of the proliferation and chemo-therapy force terms in the model, which makes estimation of their respective patient-specific parameters difficult. Second, the parameters which were found seem to be a lot higher than one would expect from a medical point of view; this makes the connection with the 'logical' simulated data weaker. Furthermore, the real data represented the data in a binary manner, while smooth transitions of the boundary of the tumour were expected. This makes growth and shrinkage of the tumour with a diffusive term less suitable.

In future work, one should explore the possibilities of different models which might better suit the data. Level-set methods are very effective for binary data, implicitly modeling the boundary of the tumour instead of modelling the cells individually. The tissue types should also extend to contain a fourth category of dead tumour cells, to better reflect the elastic properties of the breast over time. The suitability of the current model can be investigated by verifying if solutions exist when fitting the parameters to all the scans of a patient; this was briefly explored in this work, but no satisfactory results were found just yet.

On the implementation side, there are likely better ways of utilising parallelisation of the code. While the Parareal method works great, its memory requirements when also calculating the Jacobian are nearly impractical. It should be compared to straightforward parallel matrix-vector products before continuing with other research. If one wants to support the conclusions found with the real data better, more patients should be added to the data pool. If these scans are yet to be acquired, one should first look into existing research of the timing of scans to make predictions more accurate.

List of references

[1] Eva Slingerland. Modelling Breast Cancer Treatment: Mechanically Coupled Reaction-Diffusion model to predict tumour response in HER2+ patients. Master's thesis, Delft University of Technology, 2022.

[2] Integraal Kankercentrum Nederland. Kanker in 2032. `https://iknl.nl/kanker-in-2032`, October 2022. Online.

[3] CBS. Doodsoorzaken 2000-2020. `https://www.cbs.nl/nl-nl/longread/statistische-trends/2021/doodsoorzaken-2000-2020?onepage=true`, September 2021. Online.

[4] Melina Arnold, Eileen Morgan, Harriet Rumgay, Allini Mafra, Deependra Singh, Mathieu Laversanne, Jerome Vignat, Julie R Gralow, Fatima Cardoso, Sabine Siesling, et al. Current and future burden of breast cancer: Global statistics for 2020 and 2040. *The Breast*, 66:15–23, 2022.

[5] Nathalie L. Oudhof. Predicting Tumour Response: Using magnetic resonance imaging to predict tumour response of HER2+ breast cancer patients. Master's thesis, Delft University of Technology, 2022.

[6] Peter Boyle, Bernard Levin, et al. *World cancer report 2008*. IARC Press, International Agency for Research on Cancer, 2008.

[7] Christopher P Wild, Bernard W Stewart, and C Wild. *World cancer report 2014*. World Health Organization Geneva, Switzerland, 2014.

[8] David G Hicks and Swati Kulkarni. HER2+ breast cancer: review of biologic relevance and optimal use of diagnostic tools. *American journal of clinical pathology*, 129(2):263–273, 2008.

[9] Larry Norton. Evolving concepts in the systemic drug therapy of breast cancer. In *Seminars in oncology*, volume 24, pages S10–3, 1997.

[10] Larry Norton. Kinetic concepts in the systemic drug therapy of breast cancer. In *Seminars in oncology*, volume 26, pages 11–20, 1999.

[11] Jared A Weis, Michael I Miga, Lori R Arlinghaus, Xia Li, A Bapsi Chakravarthy, Vandana Abramson, Jaime Farley, and Thomas E Yankeelov. A mechanically coupled reaction–diffusion model for predicting the response of breast tumors to neoadjuvant chemotherapy. *Physics in Medicine & Biology*, 58(17):5851, 2013.

[12] Jared A Weis, Michael I Miga, Lori R Arlinghaus, Xia Li, Vandana Abramson, A Bapsi Chakravarthy, Praveen Pendyala, and Thomas E Yankeelov. Predicting the response of breast cancer to neoadjuvant therapy using a mechanically coupled reaction–diffusion model. *Cancer research*, 75(22):4697–4707, 2015.

[13] David A Hormuth, Jared A Weis, Stephanie L Barnes, Michael I Miga, Erin C Rericha, Vito Quaranta, and Thomas E Yankeelov. A mechanically coupled reaction–diffusion model that incorporates intra-tumoural heterogeneity to predict in vivo glioma growth. *Journal of The Royal Society Interface*, 14(128):20161010, 2017.

[14] Larry Norton. Conceptual basis for advances in the systemic drug therapy of breast cancer. In *Seminars in oncology*, volume 24, pages S11–2, 1997.

[15] Laura Heacock, Alana Lewin, Abimbola Ayoola, Melanie Moccaldi, James S Babb, Sungheon G Kim, and Linda Moy. Dynamic contrast-enhanced mri evaluation of pathologic complete response in human epidermal growth factor receptor 2 (her2)-positive breast cancer after her2-targeted therapy. *Academic radiology*, 27(5):e87–e93, 2020.

[16] Angela M Jarrett, David A Hormuth, Stephanie L Barnes, Xinzeng Feng, Wei Huang, and Thomas E Yankeelov. Incorporating drug delivery into an imaging-driven, mechanics-coupled reaction diffusion model for predicting the response of breast cancer to neoadjuvant chemotherapy: theory and preliminary clinical results. *Physics in Medicine & Biology*, 63(10):105015, 2018.

[17] Vishwa S Parekh and Michael A Jacobs. Multiparametric radiomics methods for breast cancer tissue characterization using radiological imaging. *Breast cancer research and treatment*, 180(2):407–421, 2020.

[18] Mette S van Ramshorst, Claudette E Loo, Emilie J Groen, Gonneke H Winter-Warnars, Jelle Wesseling, Frederieke van Duijnhoven, Marie-Jeanne T Vrancken Peeters, and Gabe S Sonke. MRI predicts pathologic complete response in HER2-positive breast cancer after neoadjuvant chemotherapy. *Breast Cancer Research and Treatment*, 164(1):99–106, 2017.

[19] Stefan Klein, Marius Staring, Keelin Murphy, Max A Viergever, and Josien PW Pluim. Elastix: a toolbox for intensity-based medical image registration. *IEEE transactions on medical imaging*, 29(1):196–205, 2009.

[20] Angela M Jarrett, David A Hormuth II, Chengyue Wu, Anum S Kazerouni, David A Ekrut, John Virostko, Anna G Sorace, Julie C DiCarlo, Jeanne Kowalski, Debra Patt, et al. Evaluating patient-specific neoadjuvant regimens for breast cancer via a mathematical model constrained by quantitative magnetic resonance imaging data. *Neoplasia*, 22(12):820–830, 2020.

[21] Angela M Jarrett, David A Hormuth, Vikram Adhikarla, Prativa Sahoo, Daniel Abler, Lusine Tumyan, Daniel Schmolze, Joanne Mortimer, Russell C Rockne, and Thomas E Yankeelov. Towards integration of 64Cu-DOTA-trastuzumab PET-CT and MRI with mathematical modeling to predict response to neoadjuvant therapy in HER2+ breast cancer. *Scientific reports*, 10(1):1–14, 2020.

[22] Cornelis Vuik, Fredericus Johannes Vermolen, Martin Bastiaan Gijzen, and MJ Vuik. *Numerical Methods for Ordinary differential equations*. VSSD, 2007.

[23] JJIM van Kan, August Segal, and Fredericus Johannes Vermolen. *Numerical methods in scientific computing*. VSSD, 2005.

[24] Bernard Bialecki and Andreas Karageorghis. Finite difference schemes for the Cauchy–Navier equations of elasticity with variable coefficients. *Journal of Scientific Computing*, 62(1):78–121, 2015.

[25] C Vuik and DJP Lahaye. Scientific computing (wi4201). *Lecture notes for wi4201*, 2012.

[26] Stanley C Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM Journal on Scientific and Statistical Computing*, 2(1):1–4, 1981.

[27] Reinaldo Astudillo, JM de Gier, and Martin B van Gijzen. Accelerating the induced dimension reduction method using spectral information. *Journal of Computational and Applied Mathematics*, 345:33–47, 2019.

[28] Nathan Bell, Luke N. Olson, and Jacob Schroder. PyAMG: Algebraic multigrid solvers in python. *Journal of Open Source Software*, 7(72):4142, 2022.

[29] Jared A Weis, Michael I Miga, and Thomas E Yankeelov. Three-dimensional image-based mechanical modeling for predicting the response of breast cancer to neoadjuvant therapy. *Computer methods in applied mechanics and engineering*, 314:494–512, 2017.

[30] Jintai Chung and GM1223971 Hulbert. A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method. *Journal of Applied Mechanics*, 60, 1993.

[31] Guillermo Lorenzo, Thomas JR Hughes, Pablo Dominguez-Frojan, Alessandro Reali, and Hector Gomez. Computer simulations suggest that prostate enlargement due to benign prostatic hyperplasia mechanically impedes prostate cancer growth. *Proceedings of the National Academy of Sciences*, 116(4):1152–1161, 2019.

[32] Kenneth E Jansen, Christian H Whiting, and Gregory M Hulbert. A generalized-α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method. *Computer methods in applied mechanics and engineering*, 190(3-4):305–319, 2000.

[33] Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. Résolution d'edp par un schéma en temps «pararéel». *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 332(7):661–668, 2001.

[34] Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 1). `https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1`, 2022.

[35] Lisandro Dalcin and Yao-Lung L Fang. mpi4py: Status update after 12 years of development. *Computing in Science & Engineering*, 23(4):47–54, 2021.

[36] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[37] David A Hormuth, Stephanie L Eldridge, Jared A Weis, Michael I Miga, and Thomas E Yankeelov. Mechanically coupled reaction-diffusion model to predict glioma growth: methodological details. In *Cancer Systems Biology*, pages 225–241. Springer, 2018.

[38] Thomas F Coleman and Yuying Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on optimization*, 6(2):418–445, 1996.

[39] Trond Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

[40] Michael JD Powell. A hybrid method for nonlinear equations. *Numerical methods for nonlinear algebraic equations*, 1970.

[41] Michael JD Powell. A new algorithm for unconstrained optimization. In *Nonlinear programming*, pages 31–65. Elsevier, 1970.

[42] Manolis LA Lourakis and Antonis A Argyros. Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1526–1531. IEEE, 2005.

[43] C Voglis and IE Lagaris. A rectangular trust region dogleg approach for unconstrained and bound constrained nonlinear optimization. In *WSEAS International Conference on Applied Mathematics*, volume 7, pages 9780429081385–138, 2004.

[44] Andrew M Bradley. Pde-constrained optimization and the adjoint method. Technical report, Technical Report. Stanford University. https://cs. stanford. edu/˜ ambrad ..., 2013.

[45] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

[46] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[47] Part of the Radiology Group of the Erasmus Medical Center.

[48] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[49] Julie C DiCarlo, Angela M Jarrett, Anum S Kazerouni, John Virostko, Anna Sorace, Kalina P Slavkova, Stefanie Woodard, Sarah Avery, Debra Patt, Boone Goodgame, et al. Analysis of simplicial complexes to determine when to sample for quantitative dce mri of the breast. *Magnetic Resonance in Medicine*, 89(3):1134–1150, 2023.

A
Appendix

A.1. Details of MRI Scans

The number of voxels in the $x-$ and $y-$ dimension are 180 for patient p_1 and p_3, and 210 for patient p_2. In the next table, the size of the voxels and the number of slices in the $z-$dimension are given for each patient.

Table A.1: Size of the voxels and the number of slices in the $z-$dimension for the three patients.

		DW			DCE		
		t_0	t_1	t_2	t_0	t_1	t_2
Voxel length (mm)	p_1	6.5	6.5	6.5	1.6	1.6	2.2
	p_2	4.8	6.5	6.5	0.9	2.2	2.2
	p_3	6.5	6.5	6.5	1.6	1.6	2.2
Number of slices	p_1	32	32	32	120	120	86
	p_2	34	32	32	160	86	86
	p_3	32	32	32	120	120	86

All scans except for p_2 at t_0 were made with a field strength of $1.5T$. They used 2 pre- and 6 post-contrast scans for DCE with $15mL$ ProHance as the contrast-agent. More acquisition parameters are shown in the next tables.

Table A.2: Standard MRI acquistion parameters.

	DW-MRI	DCE-MRI
Scan sequence	Spin Echo	Vibrant
Repetition time (ms)	5468	4.724
Echo time (ms)	68.6	2.208
Flip angle (degrees)	90	10
Voxel dimension (mm)	1.4×1.4	0.66×0.66
Acquisition matrix	256×256	512×512

Table A.3: MRI acquistion parameters of p_2 at t_0.

	DW-MRI	DCE-MRI
Scan sequence	unknown	unknown
Repetition time (ms)	9337	4.33
Echo time (ms)	108	1.29
Flip angle (degrees)	90	unknown
Voxel dimension (mm)	1.89×1.89	0.80×0.80
Acquisition matrix	190×116	448×448

All patients had the same treatment schedule, the one for p_1 is shown in the next figure as an example.

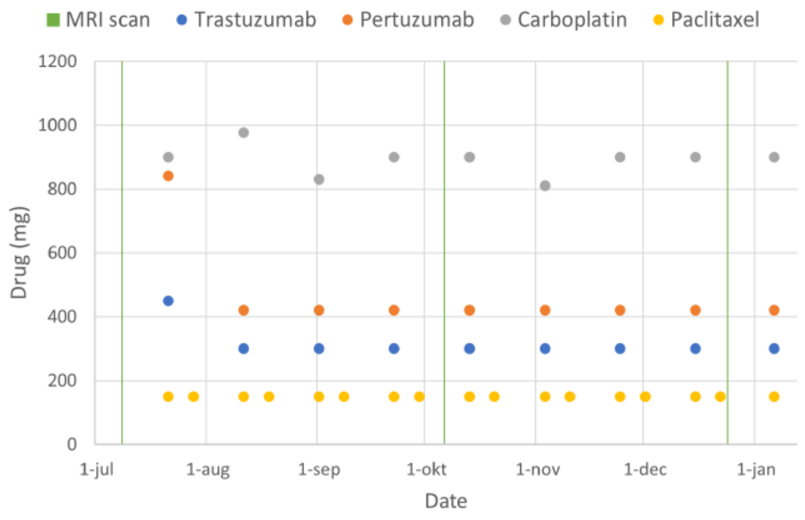

Figure A.1: Treatment schedule of patient p_1.

A.2. Full Regulariser Test Results

In the next three tables we see the results of the regulariser test for Levenberg-Marquardt, Trust Region Reflective and Adjoint State, respectively.

Table A.4: Regulariser test results for the Levenberg-Marquardt method.

			$N(t_1)$			$N(t_2)$			k		D_0	α
Reg	Par	Score	Err	Iters	Err	Diff	Pear	Err	Diff	Pear	Err	Err
Tik	10^3	$<\epsilon$	$<\epsilon$	7	0.17	8.1e-2	1.0	4.1	1.0	0.94	-6.2e-2	0.18
	10^4	$<\epsilon$	$<\epsilon$	7	0.17	8.9e-2	1.0	4.0	0.79	0.93	3.9e-2	0.18
	10^5	8.5e-5	$<\epsilon$	37	0.17	8.9e-2	1.0	4.0	0.78	0.93	4.0e-2	0.18
	10^6	8.3e-4	4.2e-5	35	0.16	8.7e-2	1.0	3.9	0.76	0.93	2.9e-2	0.17
TV	10^3	$<\epsilon$	$<\epsilon$	7	0.17	9.3e-2	1.0	5.3	10	0.83	-0.11	0.18
	10^4	$<\epsilon$	$<\epsilon$	7	0.17	8.5e-2	1.0	4.8	7.9	0.88	-8.1e-2	0.18
	10^5	2.4e-5	$<\epsilon$	39	0.17	8.4e-2	1.0	4.8	7.9	0.88	-8.2e-2	0.18
	10^6	2.4e-4	$<\epsilon$	34	0.17	8.4e-2	1.0	4.8	7.9	0.88	-8.3e-2	0.18
Lap	10^3	$<\epsilon$	$<\epsilon$	7	0.17	9.6e-2	1.0	5.9	13	0.72	-0.12	0.18
	10^4	$<\epsilon$	$<\epsilon$	7	0.17	8.5e-2	1.0	4.6	7.2	0.88	-7.9e-2	0.18
	10^5	1.7e-5	$<\epsilon$	28	0.17	8.4e-2	1.0	4.6	6.7	0.89	-9.1e-2	0.18
	10^6	1.7e-4	$<\epsilon$	26	0.17	8.3e-2	1.0	4.6	6.7	0.88	-8.2e-2	0.18

Note the table above has no results for small hyperparameters, as these values led to non-physical results, which in turn also caused instability in the explicit numerical methods.

Table A.5: Regulariser test results for the Trust Region Reflective method.

			$N(t_1)$			$N(t_2)$			k		D_0	α
Reg	Par	Score	Err	Iters	Err	Diff	Pear	Err	Diff	Pear	Err	Err
Tik	10	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	5.1	6.5	0.48	1.9e-2	8.1e-2
	10^2	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	5.1	6.5	0.48	1.9e-2	8.1e-2
	10^3	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	5.0	6.5	0.48	1.9e-2	8.1e-2
	10^4	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	2.7	2.4	0.91	1.9e-2	8.1e-2
	10^5	5.8e-5	$<\epsilon$	38	8.8e-2	4.4e-2	1.0	2.4	1.7	0.93	1.1e-2	8.1e-2
	10^6	5.9e-4	1.7e-5	34	8.4e-2	4.0e-2	1.0	2.6	2.4	0.91	2.2e-2	7.6e-2
TV	10	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	5.1	6.5	0.48	1.9e-2	8.1e-2
	10^2	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	5.1	6.5	0.48	1.9e-2	8.1e-2
	10^3	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	5.4	8.0	0.61	1.9e-2	8.1e-2
	10^4	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	3.8	5.3	0.83	1.8e-2	8.1e-2
	10^5	3.5e-5	$<\epsilon$	36	8.8e-2	4.4e-2	1.0	4.9	7.4	0.70	-2.4e-3	8.1e-2
	10^6	2.0e-4	$<\epsilon$	38	8.8e-2	4.4e-2	1.0	3.2	5.6	0.90	-4.9e-2	8.0e-2
Lap	10	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	5.1	6.5	0.48	1.9e-2	8.1e-2
	10^2	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	5.1	6.5	0.48	1.9e-2	8.1e-2
	10^3	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	5.6	8.1	0.52	1.9e-2	8.1e-2
	10^4	$<\epsilon$	$<\epsilon$	7	8.8e-2	4.2e-2	1.0	5.3	6.3	0.48	1.9e-2	8.1e-2
	10^5	1.6e-5	$<\epsilon$	29	8.8e-2	4.3e-2	1.0	3.4	5.0	0.82	4.6e-2	8.1e-2
	10^6	1.5e-4	$<\epsilon$	21	8.7e-2	4.2e-2	1.0	2.6	3.9	0.91	-4.5e-2	8.0e-2

Once again slightly better results were found for Tikhonov and TV regularisation with a hyper-parameter inbetween values, this time being $5 \cdot 10^3$.

Table A.6: Regulariser test results for the Adjoint State method.

		$N(t_1)$			$N(t_2)$			k			D_0	α
Reg	Par	Score	Err	Iters	Err	Diff	Pear	Err	Diff	Pear	Err	Err
Tik	10	2.4e-3	2.4e-3	29	2.1e-2	0.17	1.0	4.7	6.4	3.0e-2	1.2	-2.8e-4
	10^2	1.7e-3	1.7e-3	34	2.1e-2	5.9e-2	1.0	4.7	6.4	3.1e-2	-0.58	7.3e-4
	10^3	1.9e-3	1.9e-3	32	2.1e-2	3.1e-2	1.0	4.7	6.4	3.1e-2	9.9e-3	5.1e-4
	10^4	2.4e-3	2.4e-3	29	2.1e-2	0.17	1.0	4.7	6.4	3.0e-2	1.2	-2.8e-4
	10^5	2.4e-3	2.4e-3	29	2.1e-2	0.17	1.0	4.7	6.4	3.0e-2	1.2	-3.0e-4
	10^6	3.0e-3	2.9e-3	30	2.0e-2	0.38	1.0	4.7	6.4	3.0e-2	3.1	5.2e-5
TV	10	4.0e-3	4.0e-3	23	2.0e-2	0.60	1.0	4.7	6.4	2.8e-2	5.1	-5.9e-4
	10^2	2.2e-3	2.2e-3	30	2.1e-2	0.11	1.0	4.7	6.4	3.1e-2	0.72	-1.5e-4
	10^3	2.4e-3	2.4e-3	29	2.1e-2	0.17	1.0	4.7	6.4	3.0e-2	1.2	-2.8e-4
	10^4	2.3e-3	2.3e-3	31	2.1e-2	4.1e-2	1.0	4.7	6.4	3.1e-2	-0.42	1.3e-3
	10^5	2.6e-3	2.6e-3	27	2.0e-2	0.23	1.0	4.7	6.4	3.0e-2	1.8	-1.0e-4
	10^6	2.9e-3	2.7e-3	30	2.1e-2	0.23	1.0	4.7	6.4	3.0e-2	1.7	-4.9e-4
Lap	10	2.5e-3	2.5e-3	28	2.1e-2	0.22	1.0	4.7	6.4	3.0e-2	1.7	-1.7e-4
	10^2	2.3e-3	2.3e-3	31	2.1e-2	4.1e-2	1.0	4.7	6.4	3.1e-2	-0.41	1.3e-3
	10^3	2.5e-3	2.5e-3	28	2.1e-2	0.22	1.0	4.7	6.4	3.0e-2	1.7	-1.7e-4
	10^4	2.2e-3	2.2e-3	30	2.1e-2	0.11	1.0	4.7	6.4	3.1e-2	0.72	-1.5e-4
	10^5	2.6e-3	2.6e-3	27	2.1e-2	0.23	1.0	4.7	6.4	3.0e-2	1.7	-1.0e-4
	10^6	3.7e-3	3.3e-3	28	2.0e-2	0.46	1.0	4.7	6.4	3.0e-2	3.8	-8.9e-4

Convergence was not found in the same time-span, probably due to the fact that this is a Quasi-Newton method as opposed to regular Newton methods.