7-25-2018

Mathematical Model and Parameter Estimation for Tumor Growth

Lujun Yin
ly2976@rit.edu

Mathematical Model and Parameter Estimation for Tumor Growth

By

Lujun Yin

A thesis submitted in partial fulfillment of
the requirements for the degree of Master of Science
in Applied Mathematics
from the School of Mathematical Sciences
Rochester Institute of Technology

25 July 2018

Advisor:	Dr. Baasansuren Jadamba
Co-advisor:	Dr. Akhtar Khan
	Dr. Ephraim Agyingi
Committee member:	Dr. Manki Cho

Abstract

Numerous models in applied mathematics are expressed as a system of partial differential equations involving certain coefficients. In this work, we consider a tumor growth model originally proposed by Ward and King in 1997. Our main goal is to find an efficient and accurate numerical method for identification of parameters in the model (an inverse problem) from measurements of the evolving tumor over time. The so-called direct problem, in this case, is to solve a system of coupled nonlinear partial differential equations for given fixed values of the unknown parameters. We compare several derivative free and gradient based methods for the solution of the inverse problem which is formulated as an optimization problem with a constraint that is a system of partial differential equations (PDEs). Finally, we modify the original model to include a random parameter and solve the new optimization problem using the Monte Carlo method. The thesis is organized as follows. In the first two introductory chapters, we discuss the original model and the non-dimensionalized version of the model equations. The next chapter is devoted to the optimization formulation of the inverse problem. In the following chapters, we compare performances of the optimization methods. In the final chapter, we discuss the performance comparison of the optimization methods for the cases where the random parameter in the model follows either uniform or truncated normal distributions.

Keywords: Tumor growth model, inverse problem, optimization methods, random parameters

Dedication

I would like dedicate this to my family, first for their encouragement through the process, and second, for understanding that I would have to give up a lot of time with them in order to pursue my goals.

-I love you

Acknowledgement

I would like to thank my family, friends, and mentors for helping me through this learning process, your support was vital to my success.

I am especially grateful for my advisor, Dr. Akhtar Khan and Dr.Baasansuren Jadamba. In the two years I have been their student, he has been patient, kind, and understanding. Dr. Khan and Dr. Jadamba are extremely resourceful and spends much of their valuable time procuring guest speakers and other materials to better aid his students. The multiple viewpoints they provides to each problem is invaluable.

I really love the study experience in RIT. As an international student, I feel different cultures and education there. I am very appreciate to every professor who spent their precious time on teaching me and answering my questions patiently. I will never forget the two years of studying here. I thank everyone in RIT.

Most importantly, I thank my parents for their encouragement for me to complete my degree, they can be most persuasive and motivating. Throughout college, I rarely had the opportunity to visit them for more than a few days at a time, and in a way, they have sacrificed more by having me stay in school. I love my family very much.

Special thanks to my committee members: Dr. Baasansuren Jadamba, Dr. Akhtar Khan, Dr. Ephraim Agyingi, Dr. Manki Cho

Contents

List of Figures

Chapter 1

Introduction

1.1 Partial Differential Equation (PDE)

A partial differential equation (PDE) describes a relation between an unknown function and its partial derivatives. PDEs exist frequently in most topics of physics and engineering. Moreover, in recent years we have seen a dramatic increase in the applications of PDEs in many areas such as biology, chemistry, computer sciences and in economics. The general form of a PDE for a function $u(x_1, x_2, ..., x_n)$ is

$$F(x_1, x_2, ..., x_n, u, u_{x_1}, u_{x_2}, ..., u_{x_n}, ...) = 0$$

where $x_1, x_2, ..., x_n$ are the independent variables, u is the unknown function, and u_{x_i} denotes the partial derivative $\frac{\partial u}{\partial x_i}$. The equation is, in general, supplemented by additional conditions such as initial conditions (as we have often seen in the theory of ordinary differential equations (ODEs)) or boundary conditions.

The fundamental theoretical question is whether the problem consisting of the equation and its associated side conditions is well-posed. The French mathematician Jacques Hadamard $(1865-1963)$ coined the notion of well-posedness. According to his definition, a problem is called well-posed if it satisfies all of the following criteria:

Existence The problem has a solution.

Uniqueness There is no more than one solution.

Stability A small change in the equation or in the side conditions gives rise to a small change in the solution.

If one or more of the conditions above does not hold, we say that the problem is ill-posed. One can fairly say that the fundamental problems of mathematical physics are all well-posed. However, in certain engineering applications we might tackle problems that are ill-posed. In practice, such problems are unsolvable. Therefore, when we face an ill-posed problem, the first step should be to modify it appropriately in order to render it well-posed.

1.2 Optimization

Optimization is an important tool in decision science and in the analysis of physical systems. To use it, we must first identify some objective, a quantitative measure of the performance of the system under study. This objective could be profit, time, potential energy or any quantity or combination of quantities that can be represented by a single number. The objective depends on certain characteristics of the system, called variable or unknowns. Our goal is to find values of the variables that optimize the objective.

The process of identifying objective, variables and constraints for a given problem is known as modeling. Construction of an appropriate model is the first step but the most important step is the optimization step. Once the model has been formulated, an optimization algorithm can be used to find its solution. Usually, the algorithm and model are complicated enough that a computer is needed to implement this process. After an optimization algorithm has been applied to the model, we must be able to recognize whether it has succeeded in its task of finding a solution. In many cases, there are elegant mathematical expressions known as optimality conditions for checking that the current set of variables is indeed the solution of the problem. If the optimality conditions are not satisfied, they may give useful information on how the current estimate of the solution can be improved. Finally, the model may be improved by applying techniques such as sensitivity analysis, which reveals the sensitivity of the solution to changes in the model and data.

1.3 Mathematical Formulation

Mathematically speaking, optimization is the minimization or maximization of a function subject to constraints on its variables. We use the following notation:

- x is the vector of variables, also called unknowns or parameters;
- f is the objective function, a function of x that we want to maximum or minimize;
- c is the vector of constraints that the unknowns must satisfy. This is a vector function of the variables of x. The number of components in c is the number of individual restrictions that we place on the variables.

The optimization problem can then be written as

$$\min_{x \in R^n} f(x) \quad \text{subject} \quad \text{to} \quad \begin{cases} c_i(x) = 0 & i \in \varepsilon \\ c_i(x) \geq 0 & i \in \mathrm{I} \end{cases} \tag{1.1}$$

where f and each c_i are scalar-valued functions of the variables x, and I, ε are sets of indices.

Chapter 2

A Tumor growth model

In this chapter, we include an introduction to a tumor growth model originally introduced by Ward and King [19] and later considered by D.A.Knopoff [1] and specify the direct and inverse problems associated to the model.

2.1 The Direct Problem

Scientists believe that mathematical modeling of tumor growth is an effective and important part in promoting knowledge about cancer, which has become one of the most popular studied topics in mathematical biology. In the history of mathematical biology,there are many mathematical models of tumor growth including continuous models and discrete models.
The advantages of continuous models are that they are understandable, tractable to mathematical analysis and intuitive from biological principles. They contain a few parameters and can use laws from physics. On the other hand, discrete models are able to work in other scales and each cell can be treated independently with no extra complication.
Mathematical models of avascular multicellular spheroids are typically continuous models which consist of an ordinary differential equation (ODE) representing the evolution of the outer tumor boundary, and a set of partial differential equations (PDEs) describing the dramatic tumor. That is why in this

general approach of modeling, the key variables are the tumor size, e.g., tumor radius, and the concentration. Since the tumor changes in size over time, the domain on which the models are formulated must be determined as part of the solution process, giving a vast class of moving boundary problems.

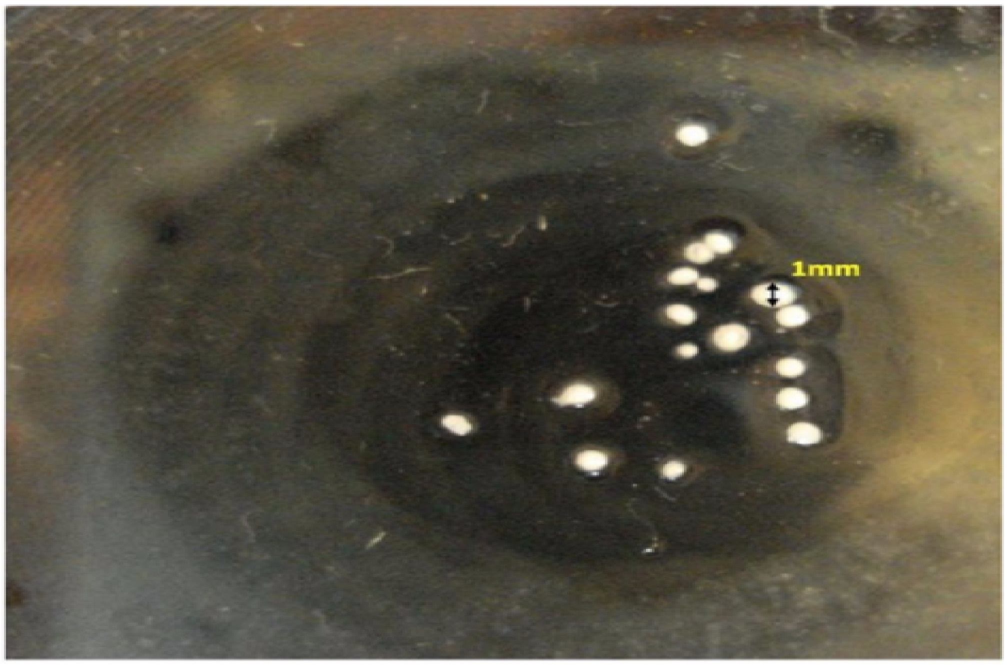


Figure 2.1: Microscopic image of neoplastic colonies that grow with an nutrient supply

In this work, we consider the model proposed by Ward and King [19]. The tumor is considered to be a spheroid consisting of a continuous of living cells, in one of two states: live or dead. The rates of birth and death depend on the nutrient. It is supposed that those processes generate volume changes, leading to cell movement described by a velocity field. In this tumor growth model, tumor growth model has the following characteristics:

- Mass of rapidly proliferating cells are supported by the adequate glucose and oxygen concentration of the surrounding environment.
- Growing spheroid model has the following factors:

1. Nutrients are readily available at the rim.
2. Concentration of nutrients significantly decreases as we move from the rim to the inner-portions of the tumor.
3. Necrotic core reduces the volume.

- Tumor-angiogenesis factors (TAFs) support tumor growth.

Assuming spherical symmetry, the system of equations to be studied is:

$$\frac{\partial \eta}{\partial t}+\frac{1}{r^2}\frac{\partial\left(r^2 v\eta\right)}{\partial r} = \left[k_m(\varsigma,\theta)-k_d(\varsigma,\theta)\right]\eta \tag{2.1}$$

$$\frac{\partial \eta}{\partial t}+\frac{1}{r^2}\frac{\partial\left(r^2 v\varsigma\right)}{\partial r} = \frac{D}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial \eta}{\partial r}\right)-\beta k_m(\varsigma,\theta)\,\eta \tag{2.2}$$

$$\frac{1}{r^2}\frac{\partial\left(r^2 v\right)}{\partial r} = \left[V_1 k_m(\varsigma,\theta)-(V_1-V_D)\,k_d(\varsigma,\theta)\right]\eta \tag{2.3}$$

Where the dependent variables η , ς and v are the live cell density (cells/unit volume), nutrient concentration and velocity, respectively.
The function k_m and k_d are taken to be generalized MichaelisMenten kinetics with exponent 1 so that we can get the following:

$$k_m(\varsigma,\theta)=A\tfrac{\varsigma}{\varsigma_c+\varsigma}$$
$$k_d(\varsigma,\theta)=B\left(1-\sigma\tfrac{\varsigma}{\varsigma_d+\varsigma}\right)$$

Initial and boundary conditions:

$$\eta(r,0)=\eta_1(r)$$
$$\tfrac{\partial\varsigma}{\partial r}(0,t)=0$$
$$v(0,t)=0$$
$$\varsigma(\wp(t),t)=c_0$$
$$\tfrac{d\wp}{dt}=v(\wp(t),t)$$

where c_0 is the external nutrient concentration.Boundary conditions (2.7) and (2.8) reflect the symmetry that was assumed. At the start time $t=0$, tumor is evolved to a certain stage.

2.2 Nondimensionalization and fixed domain method

Following the formulations used in [2],The mathematical model is rescaled and the domain $[0,\wp(t)]$ of the tumor is transformed onto the interval [0,1]. This is a common approach when dealing with boundary problems. Hence, we define the following functions

$$\begin{aligned}
&N(y,t) = V_L\eta\left(y\wp(t/A),t/A\right)\\
&C(y,t) = \tfrac{1}{c_0}\varsigma\left(y\wp(t/A),t/A\right)\\
&V(y,t) = \tfrac{1}{Ar_0}v\left(y\wp(t/A),t/A\right)\\
&S(t) = \tfrac{1}{r_0}\wp(t/A)\\
&a(c,\vartheta) = \tfrac{1}{A}\left[k_m(c,\vartheta) - k_d(c,\vartheta)\right]\\
&b(c,\vartheta) = \tfrac{1}{A}\left[k_m(c,\vartheta) - (1-\delta)k_d(c,\vartheta)\right]\\
&k(c,\vartheta) = \widehat{B}k_m(c,\vartheta)
\end{aligned}$$

Where $r_0 = \left(\frac{3V_L}{4\pi}\right)^{1/3}$ is the radius of a single cell,$\delta = \frac{V_D}{V_L}, \widehat{\beta} = \frac{r_0^2\beta}{V_L c_0 D}$ and $\vartheta = [A,B,c_c,c_d,\sigma]$ with $c_c = \frac{\varsigma_c}{c_0}, c_d = \frac{\varsigma_d}{c_0}$.

The new system that is to be solved on (y,t) domain $[0,1]\times[0,T]$ is as follows:

$$\begin{aligned}
&N_t - N_y\tfrac{S'}{S}y + \tfrac{V}{S}N_y = N\left(a(C,\vartheta) - b(C,\vartheta)N\right), \quad 0<y\le 1, \quad t>0\\
&C_{yy} + \tfrac{2}{y}C_y = K(C,\vartheta)NS^2, \quad 0<y\le 1, \quad t>0\\
&V_y + \tfrac{2}{y}V = b(C,\vartheta)NS, \quad 0<y\le 1, \quad t>0
\end{aligned}$$

The initial and boundary conditions are :

$$\begin{aligned}
&N(y,0) = N_1(y) = V_{L\eta I}\left(y\wp(0),0\right), \quad 0\le y\le 1\\
&S(0) = S_1 = \tfrac{\wp(0)}{r_0}\\
&V(0,t) = 0, \quad t>0\\
&C_y(0,t) = 0, \quad t>0
\end{aligned}$$

$$C(1,t) = 1, \quad t > 0$$
$$S'(t) = V(1,t), \quad t > 0$$

Where $N_1(y) = V_L \eta_1(y\wp(0), 0)$ and $S_1 = \frac{\wp(0)}{r_0}$.This system above will be referred to as the direct problem.This is the system to be numerically solved repeatedly during the optimization procedure.

2.3 Numerical solution of the direct problem

We apply finite difference method to solve the PDEs system of tumor growth model. Functions $S(t)$ and $N(y,t)$ are found by simple time stepping, for example,

$$N(y,t_{n+1}) = N(y,t) + \tau * \left(\frac{S'}{S} - \frac{V}{S} N_y + N[a(C,\theta) - b(C,\theta)N] \right) \Big|_{t=t_n} \tag{2.4}$$

where τ is the time step, and t_n and $t_{n+1} = t_n + \tau$ are two consecutive times. The equation for $C(y,t)$ is solved by the Newton's method. For the solution of $V(y,t)$ we use a backward finite difference scheme and this leads to a linear system involving entries of V at the grid points. For each time step, we

1 update S (relabelled as S_{new}).

2 update N (relabelled as N_{new}) using S_{new}.

3 solve for C (relabelled as C_{new}) via Newton method using N_{new} and S_{new}.

4 solve for V (relabelled as V_{new}) using N_{new}, S_{new} and C_{new}.

The pseudocode to update S, N, and solve for V is as follows: Given S_I (initial dimensionless radius of a cell calculated through r_0), where q is the number of discretization points on the interval $[0,1]$ and $S_{new} = S_{old} + \tau V_q$.

- V_q is the last entry of vector calculated from $V(y,t)$ (from the boundary condition $S_0(t) = V(1,t)$).

- Find C_{new} by using Newton's method
- Find N_{new}.
- Solve a linear system to obtain V_{new}.

2.4 Experimental Results

In our numerical experiments, we set our parameters as following:

C_c	C_d	σ	V_l	δ	c_0	s_0	$\widehat{\beta}$
0.1	0.05	0.9	10^{-9}	0.5	1.4×10^{-3}	0.021	0.005

Table 2.1: Parameter table

Next, we plot results of some numerical simulations of the direct problem using the parameters values in Table 2.1.

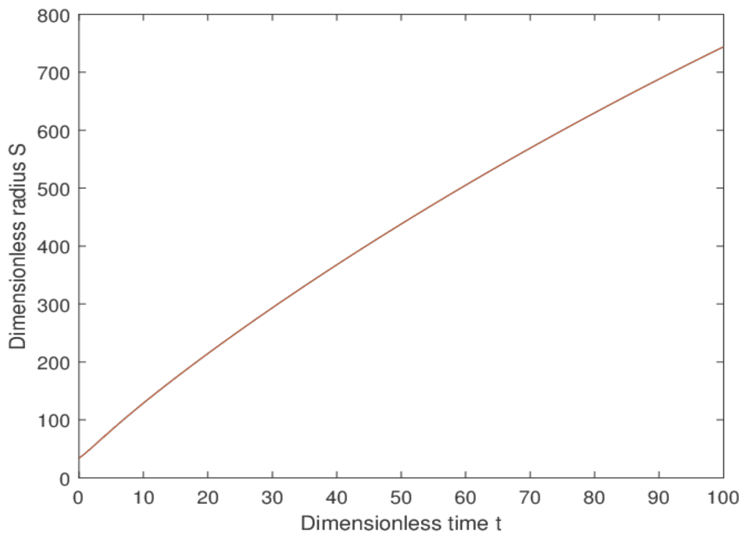


Figure 2.2: Evolution of the tumor radius S vs. dimensionless time t. where we used time step $\tau = 0.0005$ and $\alpha = 50$ grid points on $[0,1]$.

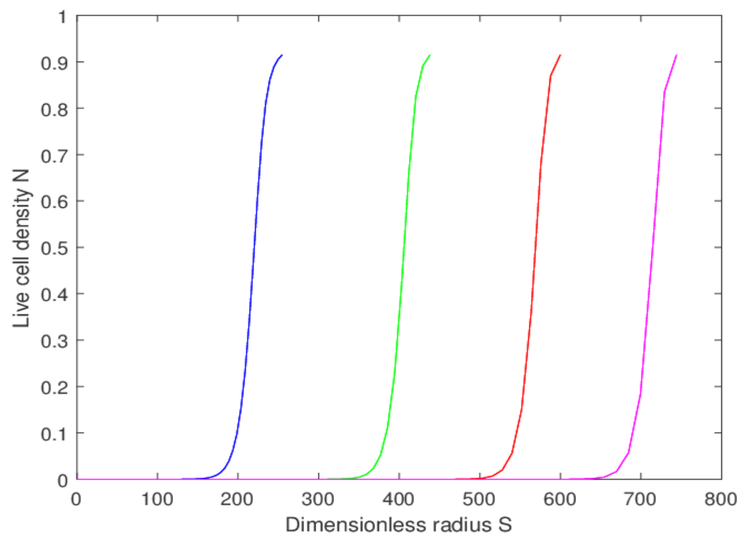


Figure 2.3: Evolution of the live cell density N vs. dimensionless radius s. where we used time step $\tau = 0.0005$ and $\alpha = 50$ grid points on $[0,1]$.

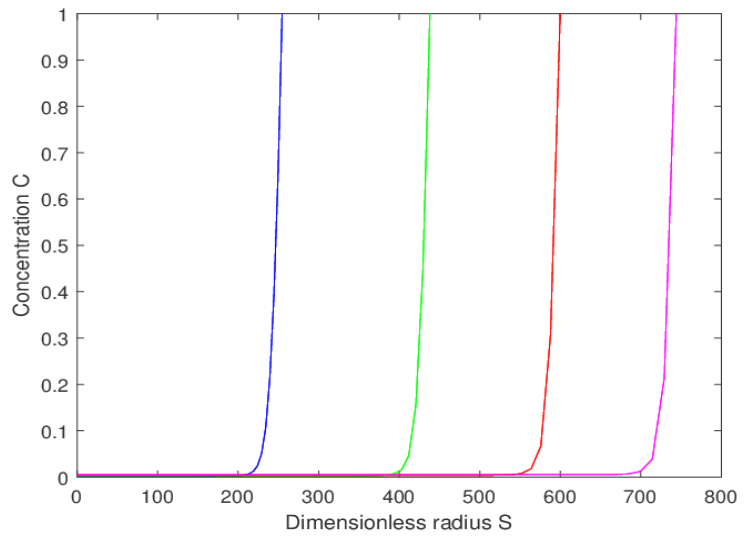


Figure 2.4: Evolution of concentration C vs. dimensionless radius s. where we used time step $\tau = 0.0005$ and $\alpha = 50$ grid points on $[0,1]$.

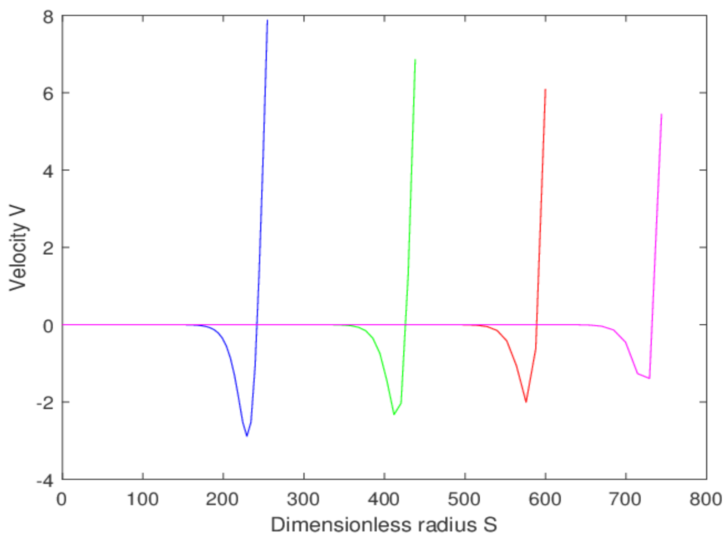


Figure 2.5: Evolution of velocity V vs. dimensionless radius s. where we used time step $\tau = 0.0005$ and $\alpha = 50$ grid points on $[0,1]$.

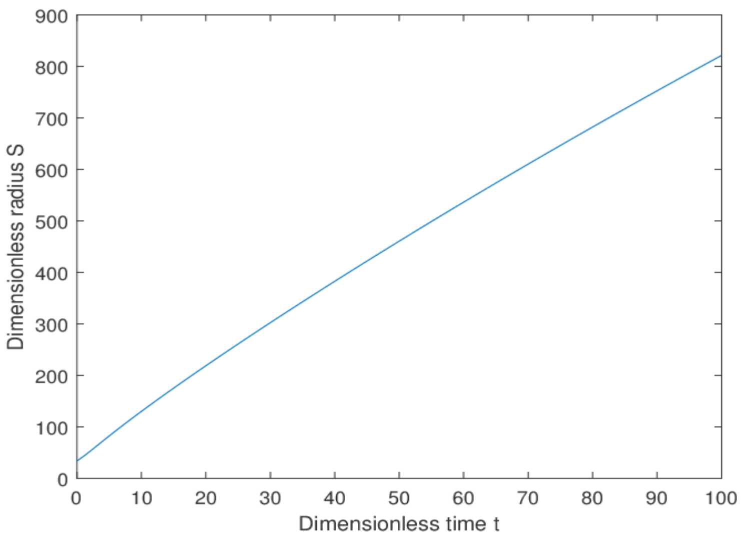


Figure 2.6: Evolution of the tumor radius S vs. dimensionless time t. where we used time step $\tau = 0.0005$ and $\alpha = 100$ grid points on $[0,1]$.

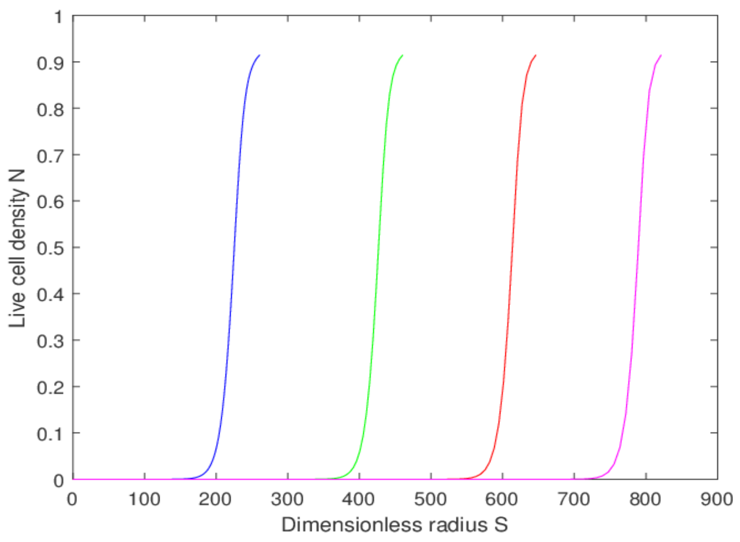


Figure 2.7: Evolution of the live cell density N vs. dimensionless radius s. where we used time step $\tau = 0.0005$ and $\alpha = 100$ grid points on $[0,1]$.

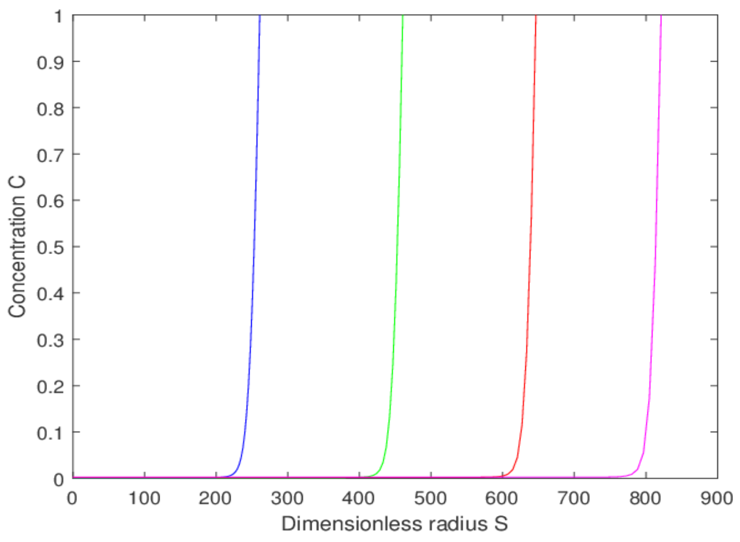


Figure 2.8: Evolution of concentration C vs. dimensionless radius s. where we used time step $\tau = 0.0005$ and $\alpha = 100$ grid points on $[0,1]$.

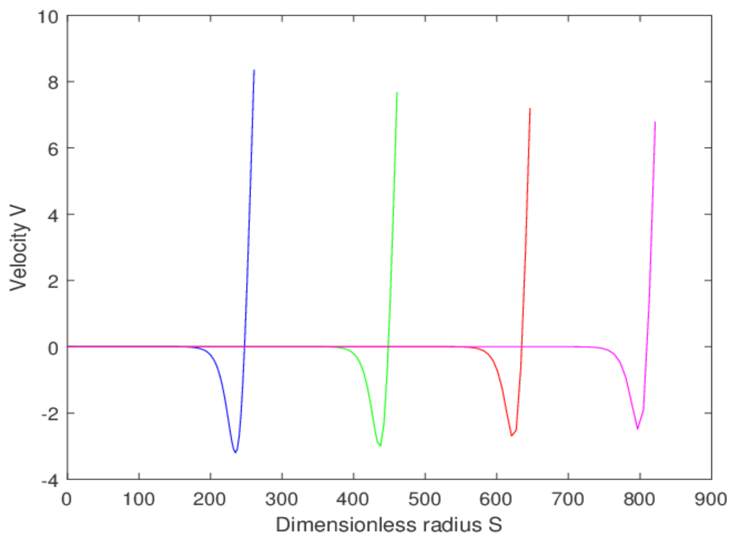


Figure 2.9: Evolution of velocity V vs. dimensionless radius s. where we used time step $\tau = 0.0005$ and $\alpha = 100$ grid points on $[0,1]$.

By observing the figures of dimensionless tumor radius figure 2.2 and 2.6, we can find that a slight kink as the growth rate decelerates a little before reaching the linear phase. This behavior is because of the time delay from when celss become quiescent to when they die. The live-cell density figure 2.3 and 2.7 show that the live-cell density is relatively constant in a small region beneath the cell surface, dropping sharply towards zero deeper into the tumour, reflecting a well-defined viable rim and a necrotic core. It should be stressed that such regions arise naturally from the model rather than being assumed a priori. Similar, the nutrient concentration figure 2.4 and 2.8 show that the nutrient concentration decreases sharply through the viable rim and tends to a constant level in the core due to the nearly complete necrosis in this region. By observing the figure 2.5 and 2.9 of velocity , we can get the conclusion that the velocity within the tumour decreases very rapidly from a positive value towards a negative minimum, before approaching zero in the necrotic core. The region of negative velocity reflects the fact that volume loss by cell death is greater there than the volume gain through mitosis.

Chapter 3

Inverse problem

In this chapter, we first give a very brief introduction to inverse problems and ill-posedness. Then, we will introduce the parameter identification problem rising from the tumor growth model discussed in Chapter 2. We will also discuss the objective function for the optimization formulation for the parameter identification problem.

3.1 Definition Of Inverse Problem

Keller [26] formulated the following very general definition of inverse problems, which is often cited in the literature: We call two problems inverses of one another if the formulation of each involves all or part of the solution of the other. Often, for historical reasons, one of the two problems has been studied extensively for some time, while the other is newer and not so well understood. In such cases, the former problem is called the direct problem, while the latter is called the inverse problem. In many cases one of the two problems is not well-posed in the following sense: Definition:(Hadamard) A problem is called well-posed if

- there exists a solution to the problem (existence)
- there is at most one solution to the problem (uniquenss)

- the solution depends continuously on the data (stability)

A problem which is not well-posed is called ill-posed. If one of two problems which are inverse to each other is ill-posed, we call it the inverse problem and the other one the direct problem. All inverse problems we will consider in the following are ill-posed.
If the data space is defined as set of solutions to the direct problem, existence of a solution to the inverse problem is clear. However, a solution may fail to exist if the data are perturbed by noise. This problem will be addressed below. Uniqueness of a solution to an inverse problem is often not easy to show. Obviously, it is an important issue. If uniqueness is is not guaranteed by the given data, then either additional data have to be observed or the set of admissible solutions has to be restricted using a-priori information on the solution. In other words, a remedy against non-uniqueness can be a reformulation of the problem.
Among the three Hadamard criteria, a failure to meet the third one is most delicate to deal with. In this case inevitable measurement and round-off errors can be amplified by an arbitrarily large factor and make a computed solution completely useless. Until the beginning of the last century it was generally believed that for natural problems the solution will always depend continuously on the data. If this was not the case, the mathematical model of the problem was believed to be inadequate. Therefore, these problems were called ill- or badly posed. Only in the second half of the last century it was realized that a huge number of problems arising in science and technology are ill-posed in any reasonable mathematical setting. This initiated a large amount of research in stable and accurate methods for the numerical solution of ill-posed problems. Today inverse and ill-posed problems are still an active area of research.

3.2 Formulation Of the Minimization Problem

In this section, we will use an inverse problem technique in order to estimate parameters(some of them unknown)that determines the behavior of a tumor's growth. We define the following vectors:

- $\phi = [N,V,C,S]^T$

- $p = [c_c, c_d, \sigma]^T$

where ϕ represents the solution of the direct problem for each choice of the vector of parameters p.
Let us assume that experimental information is available during the time interval $0 \le t \le T$.Then, the general problem we are interested in solving can be formulated as:
Find a vector of parameters p that generates data $\phi = [N, V, C, S]^T$ that is the best match to the experimental information over time $0 \le t \le T$.
For this purpose, we should construct an objective function which gives us a notion of distance between the experimental (real) data and the solution of the system of PDEs for each choice of parameters p.
We define the following functional:

$$J(N,S,p) = \frac{\mu_1}{2}\int_0^1\int_0^T [N(y,t) - N^*(y,t)]^2 dt dy + \frac{\mu_2}{2}\int_0^T [S(t) - S^*(t)]^2 dt \tag{3.1}$$

where $S(t)$ is the radius evolution obtained by solving the direct problem for a certain choice of p, S^* is the evolution measured experimentally (real data).$N(y,t)$ and $N^*(y,t)$ are the living cell concentrations for the direct problem solved with the parameters p and the real data (both of them in the domain $[0,1] \times [0,T]$). The positive constants μ_1 and μ_2 are introduced, to take into account the different order of magnitudes between N and S. In this way, these two parameters will give us some flexibility in order to choose an appropriate functional according to the experimental method used to obtain the data.
Let us define

$$\begin{bmatrix} N_t - N_y \frac{S'}{S} y + \frac{V}{S} N_y - N\left(a\left(C,p\right) - b\left(C,p\right)N\right) \\ V_y + \frac{2}{y} V - b\left(C,p\right) NS \\ C_{yy} + \frac{2}{y} C_y - K\left(C,p\right) NS^2 \\ V\left(1,\cdot\right) - S' \\ V\left(0,\cdot\right) \\ C\left(1,\cdot\right) - 1 \\ C_y\left(0,\cdot\right) \\ N\left(0,\cdot\right) - N_1 \\ S\left(0\right) - S_1 \end{bmatrix} \tag{3.2}$$

In this way we can rewrite the system of PDEs in the previous section as $E\left(\phi,p\right) = 0$.

The optimization problem that we consider the form:

$$\begin{aligned} &\text{minimize} \quad J\left(\phi,p\right) = J(S,N,p) \\ &\text{subject} \quad \text{to} \underset{p \in U_{ad}}{E\left(\phi,p\right)} = 0 \end{aligned}$$

Where J: $y \times U_{ad} \to R$ is the objective function and E :$y \times U_{ad} \to Z$ is a state equation , for y and z , Banach spaces and U_{ad} is a set of admissible points. We assume the following:

1. $U_{ad} \in R^m$ is a nonempty , closed and convex set.
2. J: $y \times U_{ad} \to R$ and E :$y \times U_{ad} \to Z$ are continuously Frechet-differentiable functions.
3. For each $p \in U_{ad}$ there exists an unique corresponding solution$\phi\left(p\right) \in y$ such that $E\left(\phi\left(p\right),p\right) = 0$. Thus there is an unique solution operator $p \in U_{ad} \mapsto \phi\left(p\right) \in y$.
4. The derivative $\frac{\partial E}{\partial \phi}\left(\phi\left(p\right),p\right) : y \to Z$ is a continuous linear operator and it is continuously invertible for all $p \in U_{ad}$.

Under these hypotheses $\phi(p)$ is continuously differentiable on $p \in U_{ad}$ by the implicit function theorem. Thus, it is reasonable to define the following so-called reduced problem

$$\begin{aligned} &\text{minimize} \widetilde{J}(p) = J(\phi(p), p) \\ &\text{subject} \quad \text{to} \underset{p \in U_{ad}}{E(\phi, p)} = 0 \end{aligned}$$

where $\phi(p)$ is given as the solution of $E(\phi(p), p) = 0$.

To find a minimum of continuously differentiable function $\widetilde{J}$, it will be necessary to compute the derivative of this reduced objective function.

Chapter 4

Optimization Methods

4.1 Gradient Method

Optimization refers to finding the maximum or minimum of a real-valued function, which is called objective function. Since locating the maximum of a function $f(x)$ is equivalent to locating the minimum of $-f(x)$, it suffices to consider minimization alone in developing computational methods. Methods for unconstrained optimization fall into two groups, depending on whether derivatives of the objective function $f(x)$ are used. If an algebraic function is known for $f(x)$, the derivatives can be easily determined by hand or basic algebraic computation in most cases. Derivative information should be used if it is possible, but there are many reasons why it might not be available. In particular, the objective function may be too complicated, too high dimensional, or not known in a form that may be differentiated.

In optimization problems, gradient method is an important algorithm to solve problems of the form

$$\min_{x \in R^n} f(x) \tag{4.1}$$

with the search directions defined by the gradient of the function at the current point. In this chapter we will introduce some optimization methods based on basic idea of gradient method such as the gradient descent, the conjugate gradient method and the nonlinear conjugate gradient method.

4.1.1 Gradient Descent

Gradient descent in [30] is a first-order iterative optimization algorithm to find the minimum of a function in a local domain. By using gradient descent, we take steps proportional to the negative of the gradient of the function at the current point. Since the gradient ∇f points in the direction of the steepest growth f , the opposite direction $-\nabla f$ is the line of steepest descent. How far should we go along this direction? Now that we have reduced the problem to minimizing along a line, let one of the one-dimensional methods decide how far to go. After the new minimum along the line of steepest descent is located, repeat the process, starting at that point. That is, find the gradient at the new point, and do a one-dimensional minimization in the new direction.

To explain the idea of gradient descent clearly, I will explain the idea depending on an easy example. Suppose we're going down a mountain. Firstly we are going down the mountain in any direction for a distance. Secondly we calculate the gradient of the current point to get a new better direction which can lead us to go down the mountain more quickly. Thirdly we are going down the mountain in this new for a distance. Then we just need to do step 2 and 3 again and again until we arrive at the foot of the mountain. Note: The foot of the mountain represents that the objective function is nearly zero. In addition new better directions represents the local best direction to go down the mountain.

```
Algorithm: Gradient descent
for $i = 0, 1, 2, ...$ Do the following steps:
    Step 1: $v = \nabla f(x_i)$
    Step 2: Minimize $f(x - sv)$ for scalar $s = s^*$
    Step 3: $x_{i+1} = x_i - s^* v$
end
```

Convergence of Steepest Descent is slower compared with the Newton Method for a good reason. Newton's method is solving an equation and is using the first and second derivatives. Steepest Descent is actually minimizing by following the downhill direction and is suing only first derivative information.

4.1.2 Conjugate Gradient Method

The conjugate gradient method in [29] is an iterative method for solving a linear system of equations

$$Ax = b \tag{4.2}$$

where A is an $n \times n$ matrix that is symmetric and positive definite. The problem can be stated equivalently as the following minimization problem:

$$f(x) = \frac{1}{2}x^T Ax - b^T x \tag{4.3}$$

Both problems have the same unique solution. This equivalence will allow us to interpret the Conjugate Gradient Method either as an algorithm for solving linear systems or as a technique for minimization of convex quadratic functions. We will note that the gradient of *phi* equals the residual of the linear system

$$\nabla f(x) = Ax - b \overset{def}{=} r(x) \tag{4.4}$$

One of the remarkable properties of the Conjugate Gradient Method is its ability to generate, a set of vectors with a property known as conjugacy. A set of nonzero vectors $\{p_0, p_1, ..., p_l\}$ is said to be conjugate with respect to the symmetric positive definite matrix A if

$$p_i^T A p_j = 0, for \quad all \quad i \neq j \tag{4.5}$$

The key observation is that the residual $r = b - Ax$ of the linear system is $-\nabla f(x)$,the direction of Gradient descent of the function f at the point x. Suppose we have chosen a search direction, denoted by vector d. To minimize $f(x)$ along that direction is to find the α that the function $h(\alpha) = f(x + \alpha d)$. We will set the derivative to zero to find the minimum:

$$0 = \nabla f \cdot d = (\alpha Ad - r)^T d \tag{4.6}$$

which implies that:

$$\alpha = \frac{r^T d}{d^T A d} = \frac{r^T r}{d^T A d}. \tag{4.7}$$

We conclude from this calculation that we could alternatively solve for the minimum of a paraboloid by using the Conjugate Gradient Method, but replacing

$$r_i = -\nabla f \tag{4.8}$$

and

$$\alpha_i = \alpha \qquad \text{that} \quad \text{minimizes} \quad f\left(x_{i-1} + \alpha d_{i-1}\right). \tag{4.9}$$

In fact, in looking at it in this way, notice that we have expressed conjugate gradient completely in terms of f. We can run the algorithm in this form for general f. Near regions where f has a parabolic shape, the method will move toward the bottom very quickly.

```
Algorithm: Conjugate Gradient Method
Let $(x_0)$ be the initial guess and set $d_0 = r_0 = -\nabla f$
for $i = 0, 1, 2, ...$ that minimizes $f\left(x_{i-1} + \alpha d_{i-1}\right)$
    Step 1: $x_i = x_{i-1} + \alpha_i d_{i-1}$
    Step 2: $r_i = -\nabla f\left(x_i\right)$
    Step 3: $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
    Step 4: $d_i = r_i + \beta_i d_{i-1}$
end
```

4.1.3 Nonlinear Conjugate Gradient Method

We have noted that the Conjugate Gradient Method can be viewed as a minimization algorithm for the convex quadratic function $f(x) = \frac{1}{2}x^TAx - b^Tx$. Similarly we can adapt the approach to minimize general convex functions, or even general nonlinear functions f. Since there are so many nonlinear methods based on Conjugate Gradient Method, we only simply introduce one Nonlinear Conjugate Gradient Method in [5], which is called The Fletcher-Reeves Method. Fletcher and Reeves showed that an extension of this kind is possible by making two simple changes in Algorithm of Conjugate Gradient Method. First, in place of the choice for the step length α_k, we need to perform a line search that identifies an approximate minimum of the nonlinear function f along p_k. Second, the residual r, which is simply the gradient of f in Algorithm of Conjugate Gradient Method, must be replaced by the gradient of the nonlinear objective f. These two changes give rise to the following algorithm for nonlinear optimization.

```
Algorithm: Fletcher-Reeves Method
Give $x_0$;
Evaluate $f_0 = f(x_0)$, $\nabla f_0 = \nabla f(x_0)$;
Set $p_0 = -\nabla f_0$, $k = 0$;
while $\nabla f_k \neq 0$
    Step 1: Compute $\alpha_k$ and set $x_{k+1} = x_k + \alpha_k p_k$;
    Step 2: Evaluate $\nabla f_{k+1}$;
    Step 3: $\beta_{k+1} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$;
    Step 4: $p_{k+1} = -\nabla f_{k+1} + \beta_{k+1} p_k$;
    Step 5: $k = k + 1$;
end
```

4.2 Optimization Methods Based on Newton's Method

In this section, we will introduce some optimization methods based on Newton's method including Quasi-Newton method, BFGS and L-BFGS.

4.2.1 Quasi-Newton Method

A standard alternative to Newton method is a class of line search methods where the search direction is defined by

$$d^{(j)} = -C_j \nabla f\left(x^{(j)}\right) \tag{4.10}$$

where C_j is updated in each iteration by a quasi-Newton updating formula in such a way that it has certain properties of the inverse of the true Hessian.

As long as C_j is symmetric positive definite, we have $\left(d^{(j)}\right)^T \nabla f\left(x^{(j)}\right) < 0$, that is $d^{(j)}$ is a desecent direction. To update this matrix we impose the well known secant equation:

$$B_{j+1}\left(\alpha_j d^{(j)}\right) = \nabla f\left(x^{(j+1)}\right) - \nabla f\left(x^{(j)}\right) \tag{4.11}$$

If we set

$$s^{(j)} = x^{(j+1)} - x^{(j)} \quad and \quad y^{(j)} = \nabla f\left(x^{(j+1)}\right) - \nabla f\left(x^{(j)}\right) \tag{4.12}$$

equation 4.11 becomes

$$B_{j+1}s^{(j)} = y^{(j)} \tag{4.13}$$

or equivalently

$$C_{j+1}y^{(j)} = s^{(j)} \tag{4.14}$$

This requirement , together with the requirement that C_{j+1} be symmetric positive definite, is not enough to uniquely determine C_{j+1}. To do that we further require that

$$C_{j+1} = \text{argmin}_C \left\| C - C_j \right\| \tag{4.15}$$

such that C_{j+1} in the sense of some matrix norm, be the closest to C_j among all symmetric positive definite matrices that satisfy the secant equation 4.14.

4.2.2 BFGS

BFGS method in [7] is currently thought as the most effective and the most popular quasi-Newton update formula. The success of the BFGS algorithm depends on how well the updating formula for C_j approximates the inverse of the true Hessian at the current iteration. Many previous experiments have shown that the method has very strong self-correcting properties so that if, at some iterations, the matrix contains bad curvature information, it often takes only a few updates to correct these inaccuracies. For this reason, BFGS method generally works very well and once close to a minimizer, it usually attains superlinear convergence.

The most popular update formula is

$$C_{j+1}^{BFGS} = \left(I - \rho_j s^{(j)} \left(y^{(j)}\right)^T\right) C_j \left(I - \rho_j y^{(j)} \left(s^{(j)}\right)^T\right) + \rho_j s^{(j)} \left(s^{(j)}\right)^T \tag{4.16}$$

where $\rho_j = \left(\left(y^{(j)}\right)^T s^{(j)}\right)^{-1}$.

```
Algorithm BFGS
Input: $x^{(0)}, \delta, C_0$
  j=0
  while true do
    $d^{(j)} = -C_j \nabla f\left(x^{(j)}\right)$
    $\alpha_j = Linesearch\left(x^{(j)}, f\right)$
    $x^{(j+1)} = x^{(j)} + \alpha_j d^{(j)}$
    Compute $C_{j+1}$ from 4.16 and 4.12
    $j = j+1$
    if $\left\|\nabla f\left(x^{(j)}\right)\right\| \leq \delta$ then
      stop
    end if
  end while
Output: $x^{(j)}, f\left(x^{(j)}\right), \nabla f\left(x^{(j)}\right)$
```

4.2.3 L-BFGS

Aiming at dealing with the shortcomings of BFGS that requires a lot of storage space, the basic idea of L-BFGS in [8] is to only store the information of the past m iterations to reduce the demand for data storage space. A less computationally intensive method when n is large is the Limited Memory BFGS method(L-BFGS). Instead of updating and storing the entire approximated inverse Hessian C_j, the L-BFGS method never explicitly forms or stores the matrix. The first m iterations, L-BFGS and BFGS generate the same search directions.

It can also be stated that L-BFGS method has the further advantage that it only uses relatively new information. In BFGS method, the inverse Hessian contains information from all previous iterations. This may be problematic if the objective function is very different in nature in different regions.

In some cases, L-BFGS method uses as many or even fewer function evaluations to find the minimizer.

This is remarkable considering that even when using the same number of function evaluations, L-BFGS runs significantly faster than full BFGS if n is large.

```
Algorithm: Direction finding in L-BFGS
  $q = \gamma_j \nabla f\left(x^{(j)}\right)$, with $\gamma_j = \left(\left(s^{(j-1)}\right)^T y^{(j-1)}\right)\left(\left(y^{(j-1)}\right)^T y^{(j-1)}\right)^{-1}$
  for $i = (j-1) : (-1) : (j-m)$ do
    $\alpha_i = \rho_i \left(s^{(i)}\right)^T q$
    $q = q - \alpha_i y^{(i)}$
  end for
  for $i = (j-m) : 1 : (j-1)$ do
    $\beta = \rho_i \left(y^{(i)}\right)^T r$
    $r = r + s^{(i)} (\alpha_i - \beta)$
  end for Output: $d^{(j)} = -r$
```

4.2.4 Conjugate Gradient Trust Region Method

In this subsection, we will introduce a brief algorithm than combines the trust region paradigm with the inexact Newton ideas.We solve the scaled trust region problem

$$\min_{\|d\|_C \leq \Delta} \quad \phi(d) \tag{4.17}$$

where the quadratic model is

$$\phi(d) = \nabla f(x)^T d + \frac{1}{2} d^T \nabla^2 f(x) d \tag{4.18}$$

Here the C-norm is

$$\|d\|_C = \sqrt{(d^T C d)} \tag{4.19}$$

This is a method in that the approximate solution of the trust region problem lies on a piecewise linear path with the CG iterations as nodes.[27] As long as CG is performing properly nodes are added to the path until the path intersects the trust region boundary. If a direction of indefiniteness is found, then that direction is followed to the boundary. In this way a negative curvature direction, if found in the course of the CG iteration, can be exploited.

The inputs to Algorithm trcg are the current point x, the objective f, the forcing term , and the current trust region radius . The output is the approximate solution of the trust region problem d. This algorithm is not the whole story, as once the trust region problem is solved approximately, one must use $f(x_c + d)$ to compute r and then make a decision on how the trust region radius should be changed.

```
Algorithm $Trcg(d,x,f,M,,,K_max)$
  1.$r = -\nabla f(x), \rho_0 = \|r\|_2^2, k = 1, d = 0$
  2.Do while $\sqrt{\rho_{k-1}} \geq \eta \|\nabla f(x)\|_2$ and $k < k_max$
    (1) $z = Mr$
    (2) $\tau_{k-1} = z^T r$
    (3) if $k = 1$ then $\beta = 0$ and $p = z$
      else $\beta = \frac{\tau_{k-1}}{\tau_{k-2}}, p = z + \beta p$
    (4) $\omega = \nabla^2 f(x) p$
      if $p^T \omega \leq 0$ then
        find $\tau$ such that $\|d + \tau p\|_C = \Delta$
        $d = d + \tau p$ ;return
    (5) $\alpha = \frac{\tau_{k-1}}{p^T \omega}$
    (6) $r = r - \alpha\omega$
    (7) $\rho_k = r^T r$
    (8) $\widehat{d} = d + \alpha p$
    (9) If $\left\|\widehat{d}\right\|_C > \Delta$ then
      find $\tau$ such that $\|d + \tau p\|_C = \Delta$
      $d = d + \tau p$; return
    (10) $d = \widehat{d}; k = k + 1$
```

Algorithm cgtrust is based on the solution of the trust region problem from trcg.

```
Algorithm $cgtrust(x, f, \tau)$
  1.Initialize $\Delta, M, \eta, k_{\max}$
  2.Do forever
    (1) Let $x_c = x$. Compute $\nabla f(x_c)$.
    (2) Use $Trcg(d, x, f, M, , , K_max)$ to solve the trust region subproblem. Set $x_t = x + d$.
    (3) Solve the trust region subproblem with algorithm trcg
    (4) Update
```

4.3 Derivative Free Methods

4.3.1 Pattern Search Method

Sometimes it is not convenient or not possible to know the first or second derivatives of the objective function in an unconstrained nonlinear problem. In the case we can use pattern search methods which need only the ability to return the value of $f(x)$ for some input point x. For this reason they are also known as derivative-free, direct search or black box optimization methods. Pattern search methods can also be applied when the objective function is differentiable, but in that case we are ignoring the useful information about the first and second derivatives. Therefore pattern search methods are typically applied only when the derivatives are not available.

Many pattern search methods have been developed over the years. Pattern search methods follow the general form of most optimization methods: given an initial guess at a solution x_0 and an initial choice of a step length parameter $\Delta_0 > 0$. [29]Pattern search methods only require simple, as opposed to sufficient, decrease on the objective function.

```
Algorithm General Pattern search method
For $k = 0, 1, ...$
    (1) Check for convergence
    (2) Compute $f(x_k)$
    (3) Determine a step $S_k$ using Exploratory moves $(\Delta_k, P_k)$
    (4) If $f(x_k) > f(x_k + s_k)$,then $x_{k+1} = x_k + x_k$. Otherwise $x_{k+1} = x_k$.
    (5) Update $(\Delta_k, P_k)$
```

4.3.2 The Nelder Mead Method

The Nelder Mead simplex algorithm in [9] maintains a simplex S of approximations to an optimal point. In this algorithm the vertices $\{x_j\}_{j=1}^{N+1}$ are sorted according to the objective function values

$$f(x_1) \leq f(x_2) \leq \cdots \leq f(x_{N+1}) \tag{4.20}$$

x_1 is called the best vertex and x_{N+1} the worst. If several vertices have the same objective value as x_1, the best vertex is not uniquely defined, but this ambiguity has little effect on the performance of the algorithm.

The algorithm attempts to replace the worst vertex x_{N+1} with a new point of the form

$$x(\mu) = (1+\mu)\bar{x} - \mu x_{N+1} \tag{4.21}$$

where $\bar{x}$ is the centroid of the convex hull of $\{x_i\}_{i=1}^{N}$

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4.22}$$

The value of μ is selected from a sequence $-1 < \mu_{ic} < 0 < \mu_{oc} < \mu_r < \mu_c$ by rules that we called

Algorithm nelder. Our formulation of the algorithm allows for termination if either $f(x_{N+1})f(x_1)$ is sufficiently small or a user-specified number of function evaluations has been expended.

```
Algorithm nelder($S, f, \tau, k_{max}$)

  1. Evaluate $f$ at the vertices of $S$ and sort the vertices of $S$.
  2. Set $f_{count} = N+1$.
  3. While $f(x_{N+1}) - f(x_1) > \tau$
    (a) Compute $\bar{x}, x(\mu_r)$ and $f_r = f(x(\tau_r))$. $f_{count} = f_{count} + 1$.
    (b) Reflect: If $f_{count} = k_{max}$ then exit. If $f(x_1) \le f_r \le f(x_N)$, replace $x_{N+1}$ with $x(\tau_r)$ and go to step 3(g).
    (c) Expand: If $f_{count} = k_{max}$ then exit. If $f_r < f(x_1)$ then compute $f_e = f(x(\tau_e))$. $f_{count} = f_{count} + 1$.
      If $f_e < f_r$, replace $x_{N+1}$ with $x(\tau_e)$;
      otherwise replace $x_{N+1}$ with $x(\tau_r)$ and go to step 3(g).
    (d) Outside Contraction: If $f_{count} = k_{max}$ then exit. $f(x_N) \le f_r \le f(x_{N+1})$, compute $f_c = f(x(\tau_{oc}))$.
      $f_{count} = f_{count} + 1$.
      If $f_c \le f_r$ replace $x_{N+1}$ with $x(\tau_{oc})$ and go to step 3(g); otherwise go to step 3(f).
    (e) Inside Contraction: If $f_{count} = k_{max}$ then exit.If $f_r \le f(x_{N+1})$ compute $f_c = f(x(\tau_{ic}))$.
      $f_{count} = f_{count} + 1$.
      If $f_c \le f(x_{N+1})$, replace $x_{N+1}$ with $x_(\tau_{ic})$ and go to step 3(g); otherwise go to step 3(f).
    (f) Shrink: If $f_{count} \le k_{max} - N$, exit. For $2 \le i \le N+1$: set $x_i = x_1 - (x_i - x_1)/2$; compute $f(x_i)$.
    (g) Sort: Sort the vertices of $S$ so that $f(x_1) \le f(x_2) \le \cdots \le f(x_{N+1})$.
```

4.4 Gradient Projection Method

Gradient project methods [6] are methods for solving bound constrained optimization problems. In solving bound constrained optimization problems, active set methods face criticism because the working set changes slowly; at each iteration, at most one constraint is added to or dropped from the working set. If there are k_0 constraints active at the initial W_0, but k constraints active at the solution, then at least $|kk_0|$ iterations are required for convergence. This property can be a serious disadvantage in large problems if the working set at the starting point is vastly different from the active set at the solution. As a result, researchers have developed algorithms that allow the working set to undergo radical changes at each iteration and to interior-point algorithms that do not explicitly maintain a working set.

The gradient-projection algorithm is the prototypical method that allows large changes in the working set at each iteration. Given x_k, this algorithm searches along the piecewise linear path

$$P\left[x_k - \alpha \nabla f\left(x_k\right)\right], \alpha \geq 0 \tag{4.23}$$

where P is the projection onto the feasible set. A new point

$$x_{k+1} = P\left[x_k - \alpha_k \nabla f\left(x_k\right)\right] \tag{4.24}$$

is obtained when a suitable $_k > 0$ is found. For bound-constrained problems, the projection can be easily computed by setting

$$\left[P\left(x\right)\right]_i = mid\left\{x_i, l_i, u_i\right\} \tag{4.25}$$

where mid $\{\cdot\}$ is the middle (median) element of a set. The search for $_k$ has to be done carefully since the function

$$\phi\left(\alpha\right) = f\left(P\left[x_k - \alpha_k \nabla f\left(x_k\right)\right]\right) \tag{4.26}$$

If properly implemented, the gradient-projection method is guaranteed to identify the active set at a solution in a finite number of iterations. After it has identified the correct active set, the gradient-projection algorithm reduces to the steepest-descent algorithm on the subspace of free variables. As a result, this method is invariably used in conjunction with other methods with faster rates of convergence.

Chapter 5

Numerical Results

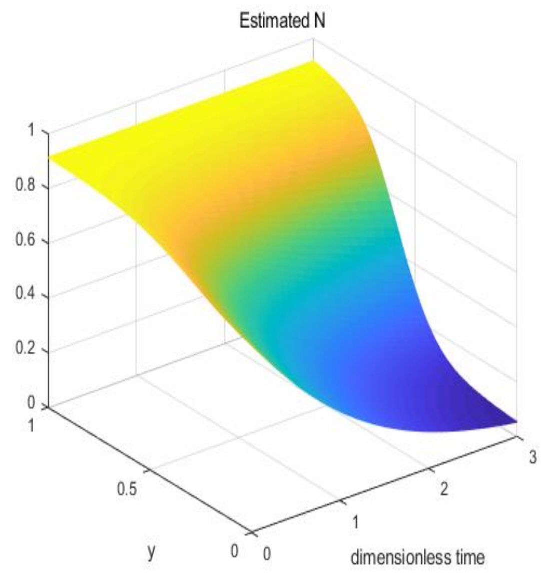
In this chapter, we will apply different optimization methods to solve the inverse problem associated to the tumor growth model including derivative free methods and gradient based methods. We compare the accuracy and efficiency of the optimization methods.

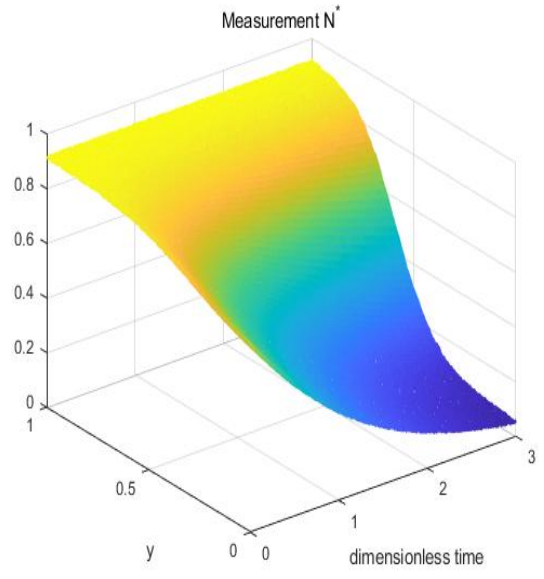
5.1 Derivative Free Methods

In this section, we apply pattern search method and the NelderMead method to get the solution of the inverse problem of the tumor growth model including the estimated N, S and estimated parameters(C_c, C_d, σ).

5.1.1 Pattern Search Method

We present the results of some numerical simulations using pattern search method in this subsection. In Figures 5.4, 5.5, 5.6 we present results of three simulations with 1%, 5%, and 10% noise levels. Top left image corresponds to the simulated N (obtained by solving the direct problem by using the identified values of the parameters). Top right image shows the noisy data fed to the optimization routine. Bottom row image shows data S^* (with added noise) and the simulated S. Pattern search method is very effective for this particular problem.





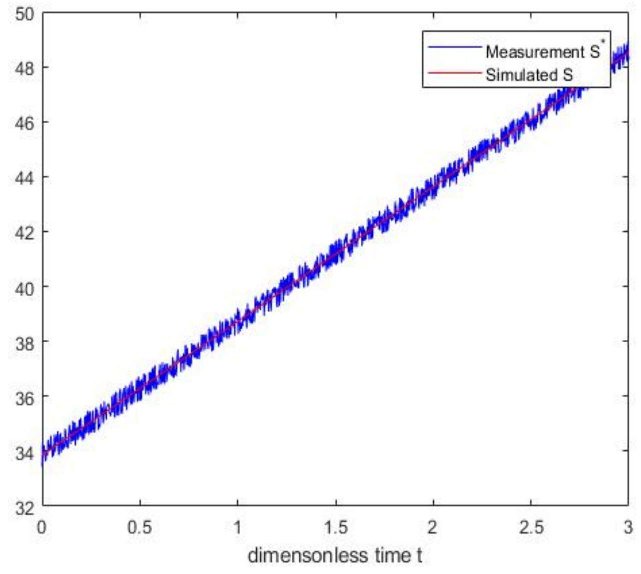
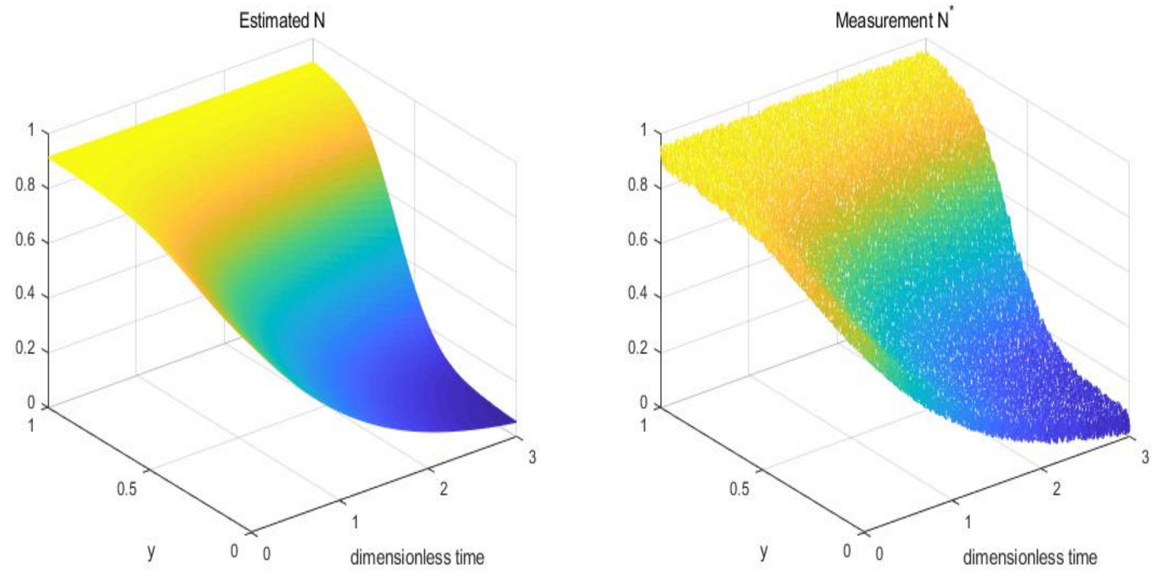


Figure 5.1: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=1%



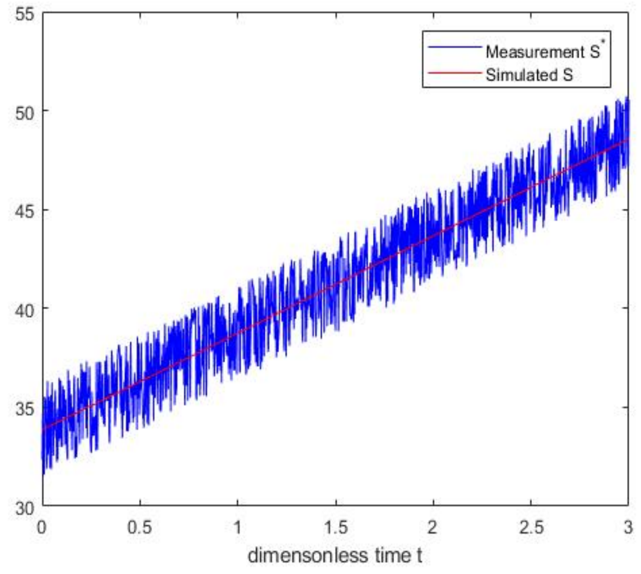
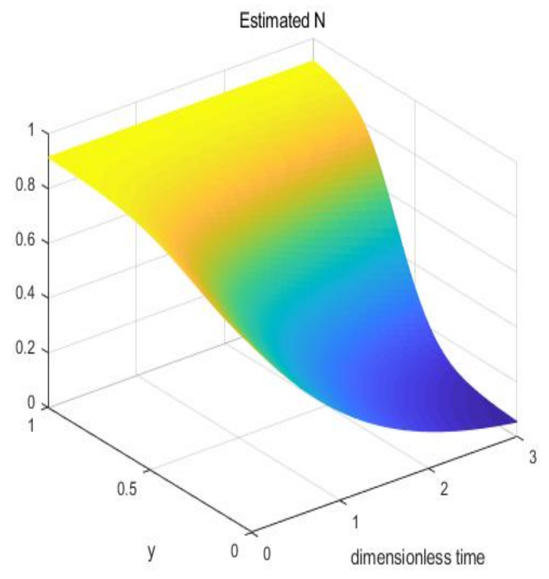
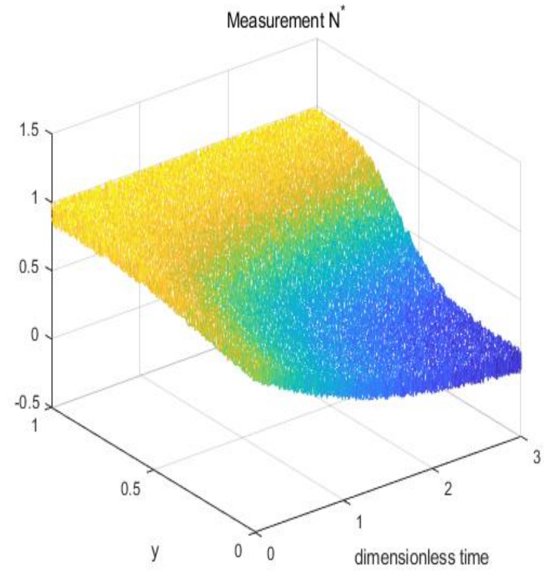


Figure 5.2: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=5%





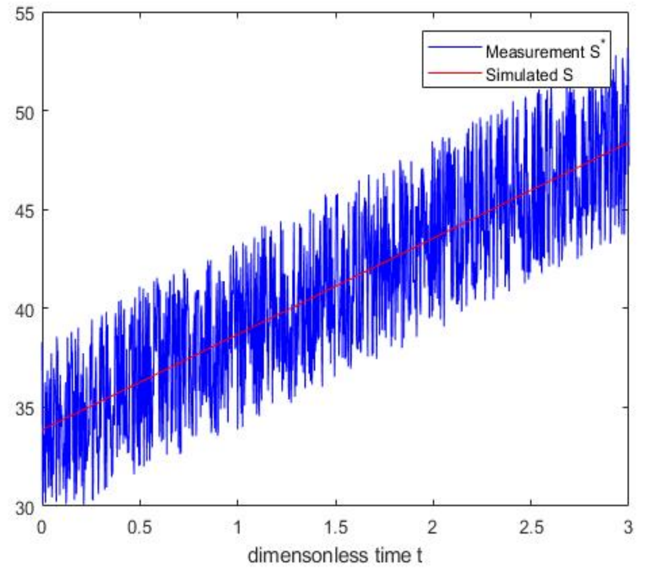


Figure 5.3: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=10%

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.0991, 0.0491, 0.8985)$	$(0.88\%, 1.67\%, 0.15\%)$	$2292.28s$
5%	$(0.1028, 0.0518, 0.9001)$	$(2.80\%, 3.60\%, 0.01\%)$	$2370.48s$
10%	$(0.0948, 0.04483, 0.8939)$	$(5.18\%, 10.32\%, 0.67\%)$	$2293.10s$

Table 5.1: Exact parameters=$(0.1, 0.05, 0.9)$, grid points=40, time step=0.005

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.1000, 0.0500, 0.9005)$	$(0.01\%, 0.18\%, 0.06\%)$	$8326.84s$
5%	$(0.0957, 0.0451, 0.8880)$	$(4.22\%, 9.78\%, 1.32\%)$	$8014.07s$
10%	$(0.1050, 0.0544, 0.9053)$	$(5.09\%, 8.81\%, 0.59\%)$	$9116.46s$

Table 5.2: Exact parameters=$(0.1, 0.05, 0.9)$, grid points=80, time step=0.0025

Tables 5.1 and 5.2 show the summaries of two sets of simulations with varying number of discretization points in the mesh (and time steps).Pattern search method is very effective for our problem. As the tables show that the method is quite robust with respect to relatively high noise level of 10%. Simulation times are quite reasonable for a derivative-free method, and they stay roughly the same during the simulations with varying levels of noise.

5.1.2 The Nelder Mead Method

We present the results of some numerical simulations using Nelder-Mead method in this subsection. In Figures 5.4, 5.5, 5.6 we present results of three simulations with 1%, 5%, and 10% noise levels. Top left image corresponds to the simulated N (obtained by solving the direct problem by using the identified values of the parameters). Top right image shows the noisy data fed to the optimization routine. Bottom row image shows data S^* (with added noise) and the simulated S.

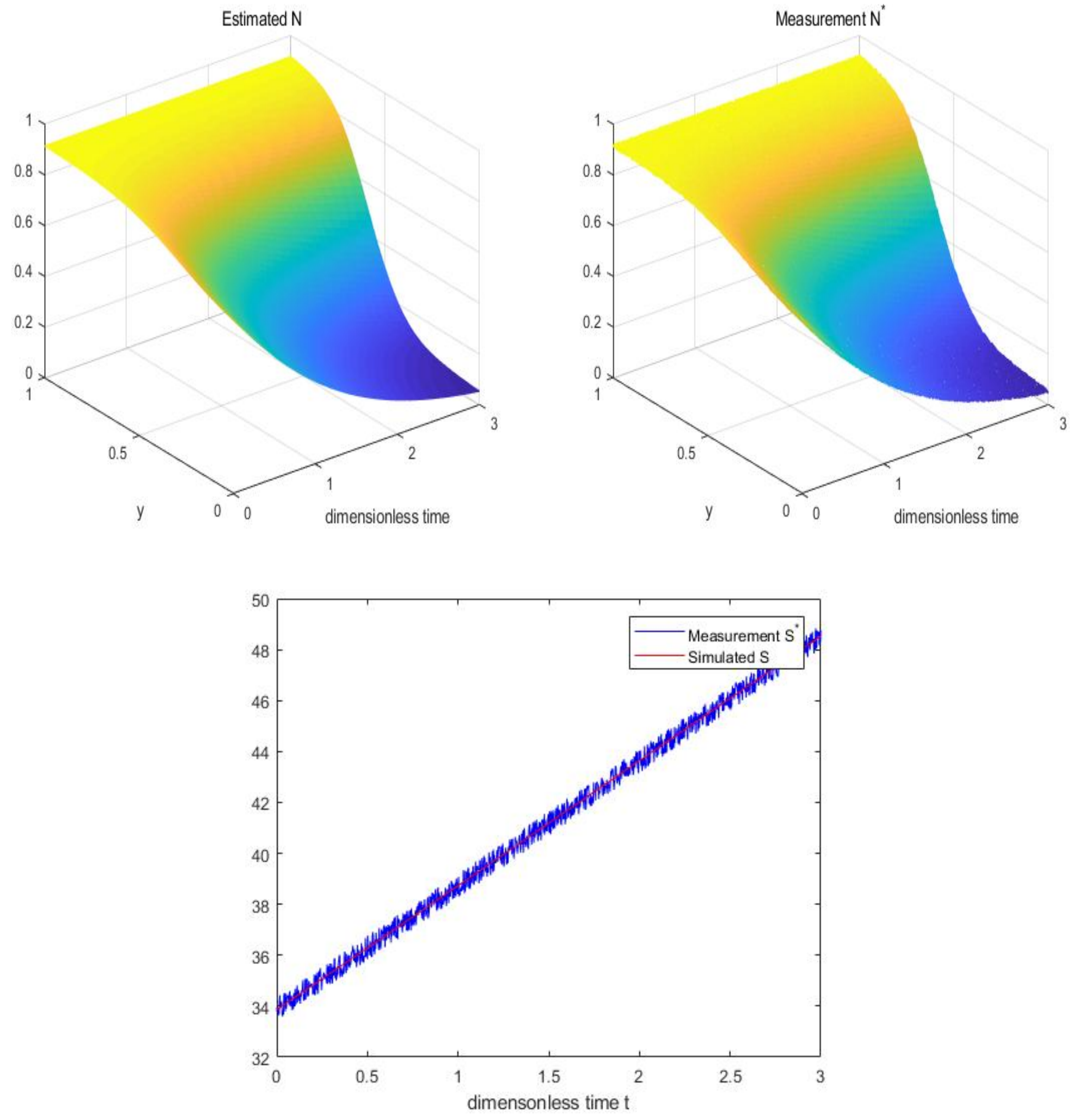
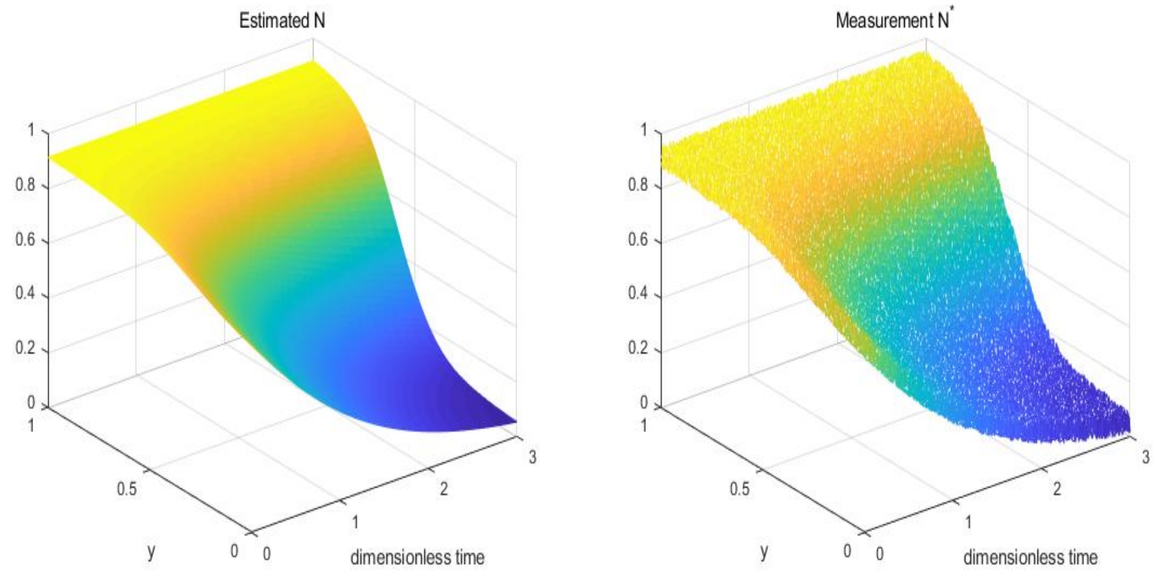


Figure 5.4: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=1%



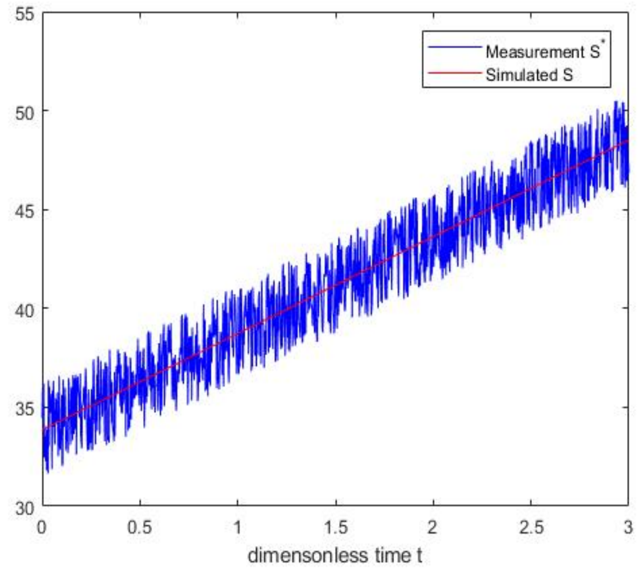


Figure 5.5: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=5%

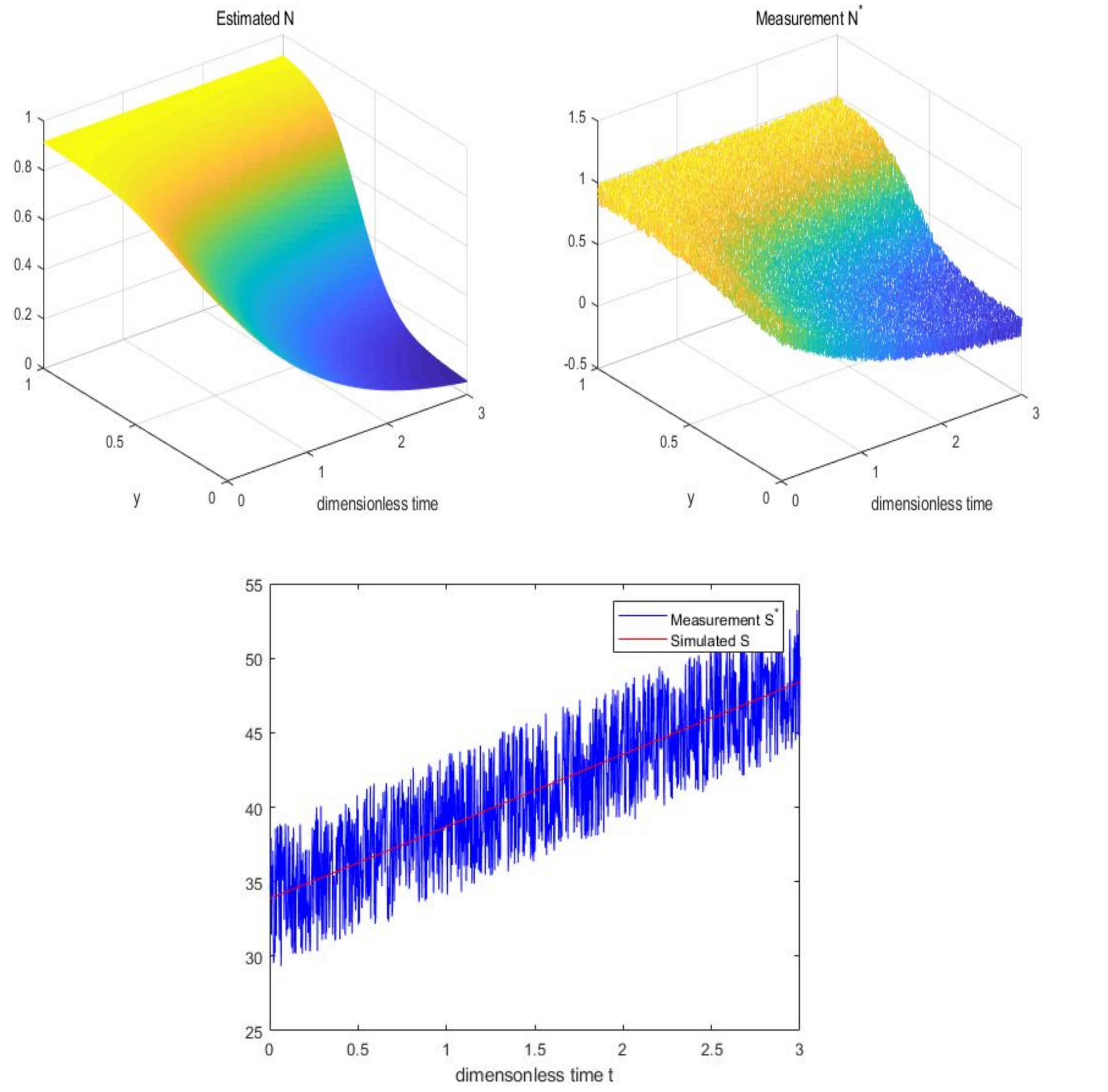


Figure 5.6: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=10%

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.0988, 0.0489, 0.8987)$	$(1.17\%, 2.05\%, 0.14\%)$	$65.12s$
5%	$(0.1062, 0.0565, 0.9116)$	$(6.21\%, 13.16\%, 1.29\%)$	$69.35s$
10%	$(0.1012, 0.0505, 0.8998)$	$(1.29\%, 1.05\%, 0.01\%)$	$46.28s$

Table 5.3: Exact parameters=$(0.1, 0.05, 0.9)$, grid points=40, time step=0.005

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.0994, 0.0496, 0.9000)$	$(0.53\%, 0.68\%, 0.01\%)$	$215.53s$
5%	$(0.1004, 0.0507, 0.9018)$	$(0.46\%, 1.49\%, 2.05\%)$	$161.59s$
10%	$(0.1022, 0.050, 0.8945)$	$(2.27\%, 0.76\%, 0.60\%)$	$150.87s$

Table 5.4: Exact parameters=$(0.1, 0.05, 0.9)$, grid points=80, time step=0.0025

Tables 5.3 and 5.4 show the summaries of two sets of simulations with varying number of discretization points in the mesh (and time steps).The Nelder Mead Method is more effective for our problem. As the tables show that the method is quite robust with respect to relatively high noise level of 10%. Simulation times are quite reasonable for a derivative-free method, and they stay roughly the same during the simulations with varying levels of noise. Comparing with Pattern Search Method, Nelder Mead Method works better for our problem between two derivative free methods.

5.2 Gradient Method

In this section, we apply Conjugated gradient trust region method and Gradient projection method to get the solution of the inverse problem of the tumor growth model including the estimated N, S and estimated parameters(C_c, C_d, σ).

5.2.1 Conjugated Gradient Trust Region Method

We present the results of some numerical simulations using Conjugated Gradient Trust Region Method in this subsection. In Figures 5.7, 5.8, 5.9 we present results of three simulations with 1%, 5%, and 10% noise levels. Top left image corresponds to the simulated N (obtained by solving the direct problem by using the identified values of the parameters). Top right image shows the noisy data fed to the optimization routine. Bottom row image shows data S^* (with added noise) and the simulated S.

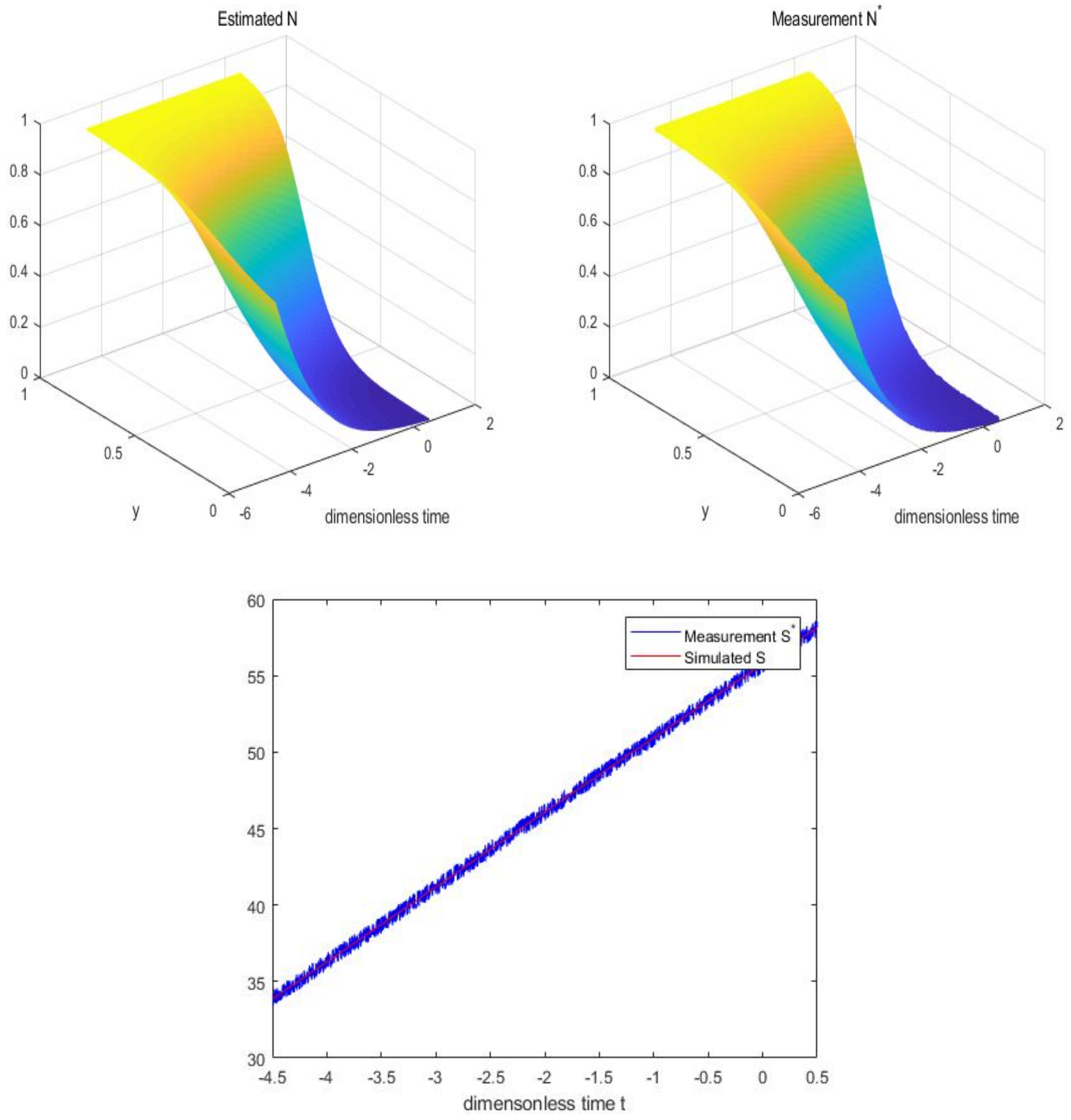


Figure 5.7: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=1%

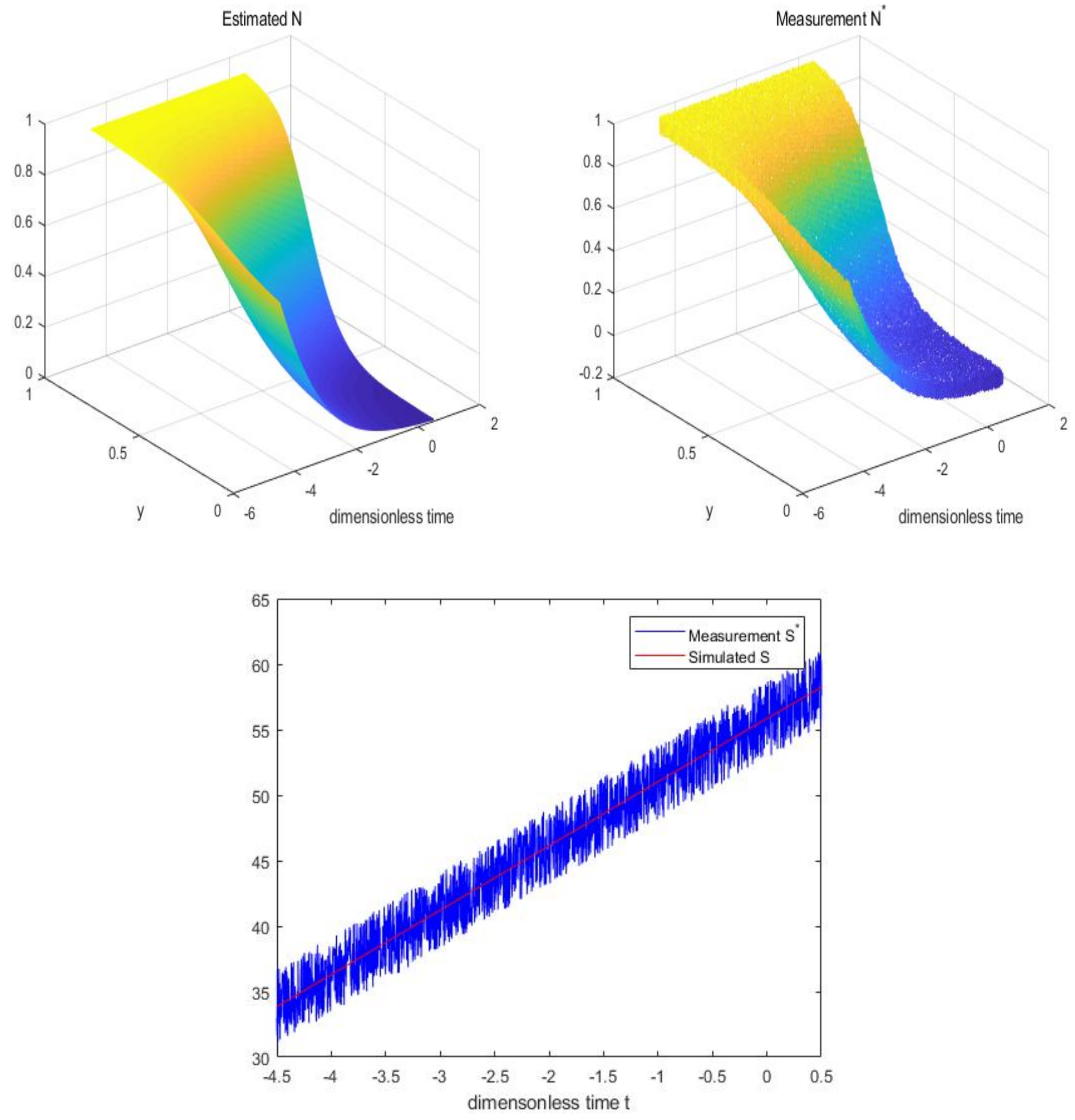
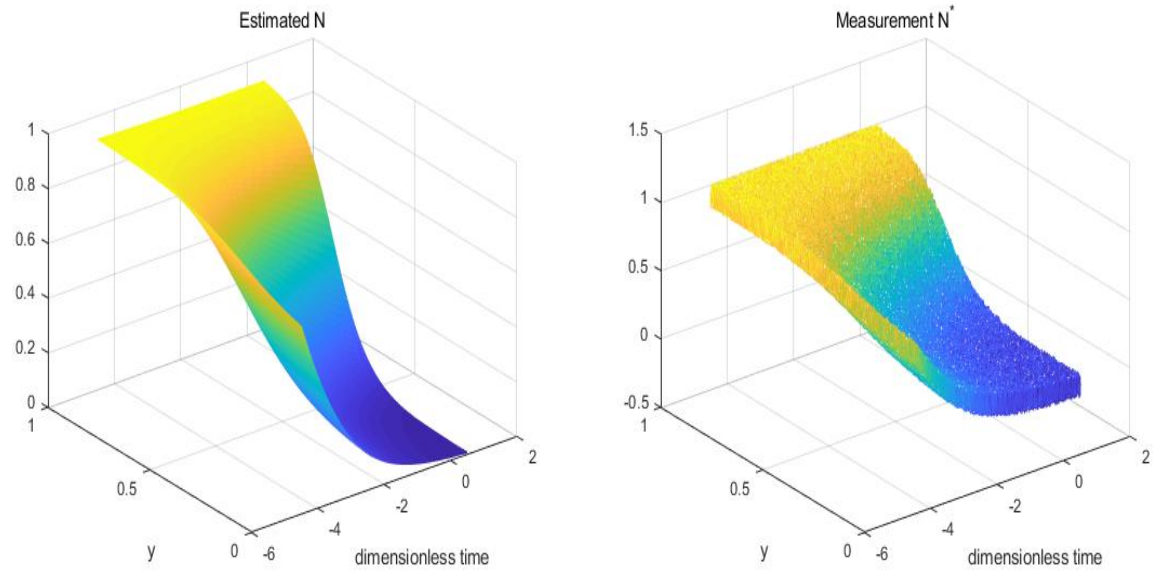


Figure 5.8: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=5%



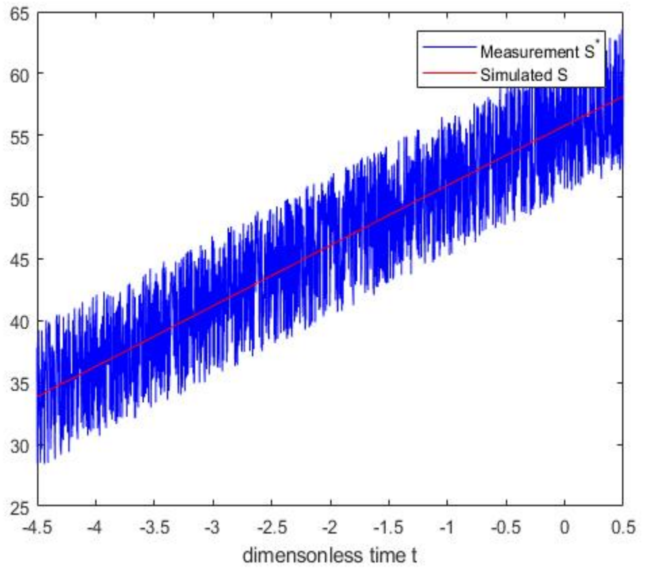


Figure 5.9: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=10%

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.1001, 0.0502, 0.9004)$	$(0.15\%, 0.46\%, 0.04\%)$	$380.00s$
5%	$(0.0984, 0.0487, 0.8978)$	$(1.55\%, 2.48\%, 0.24\%)$	$297.68s$
10%	$(0.1120, 0.0626, 0.9220)$	$(12.07\%, 25.23\%, 2.45\%)$	$334.48s$

Table 5.5: Exact parameters=$(0.1, 0.05, 0.9)$,grid points=40,time step=0.005

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.1000, 0.0501, 0.9007)$	$(0.01\%, 0.37\%, 0.08\%)$	$543.00s$
5%	$(0.0965, 0.0468, 0.8947)$	$(3.48\%, 6.24\%, 0.58\%)$	$1214.29s$
10%	$(0.0958, 0.0427, 0.8801)$	$(4.11\%, 14.53\%, 2.20\%)$	$1404.43s$

Table 5.6: Exact parameters=$(0.1, 0.05, 0.9)$,grid points=80,time step=0.025

Tables 5.5 and 5.6 show the summaries of two sets of simulations with varying number of discretization points in the mesh (and time steps). Conjugated Gradient Trust Region Method works well for our problem. As the tables show that the method is quite robust with respect to relatively high noise level of 10%. With the increase of grid points and time step, the relative error becomes smaller. Simulation times are quite reasonable for a derivative-free method, and they stay roughly the same during the simulations with varying levels of noise.

5.2.2 Gradient Projection Method

We present the results of some numerical simulations using Gradient Projection Method in this subsection. In Figures 5.10, 5.11, 5.12 we present results of three simulations with 1%, 5%, and 10% noise levels. Top left image corresponds to the simulated N (obtained by solving the direct problem by using the identified values of the parameters). Top right image shows the noisy data fed to the optimization routine. Bottom row image shows data S^* (with added noise) and the simulated S.

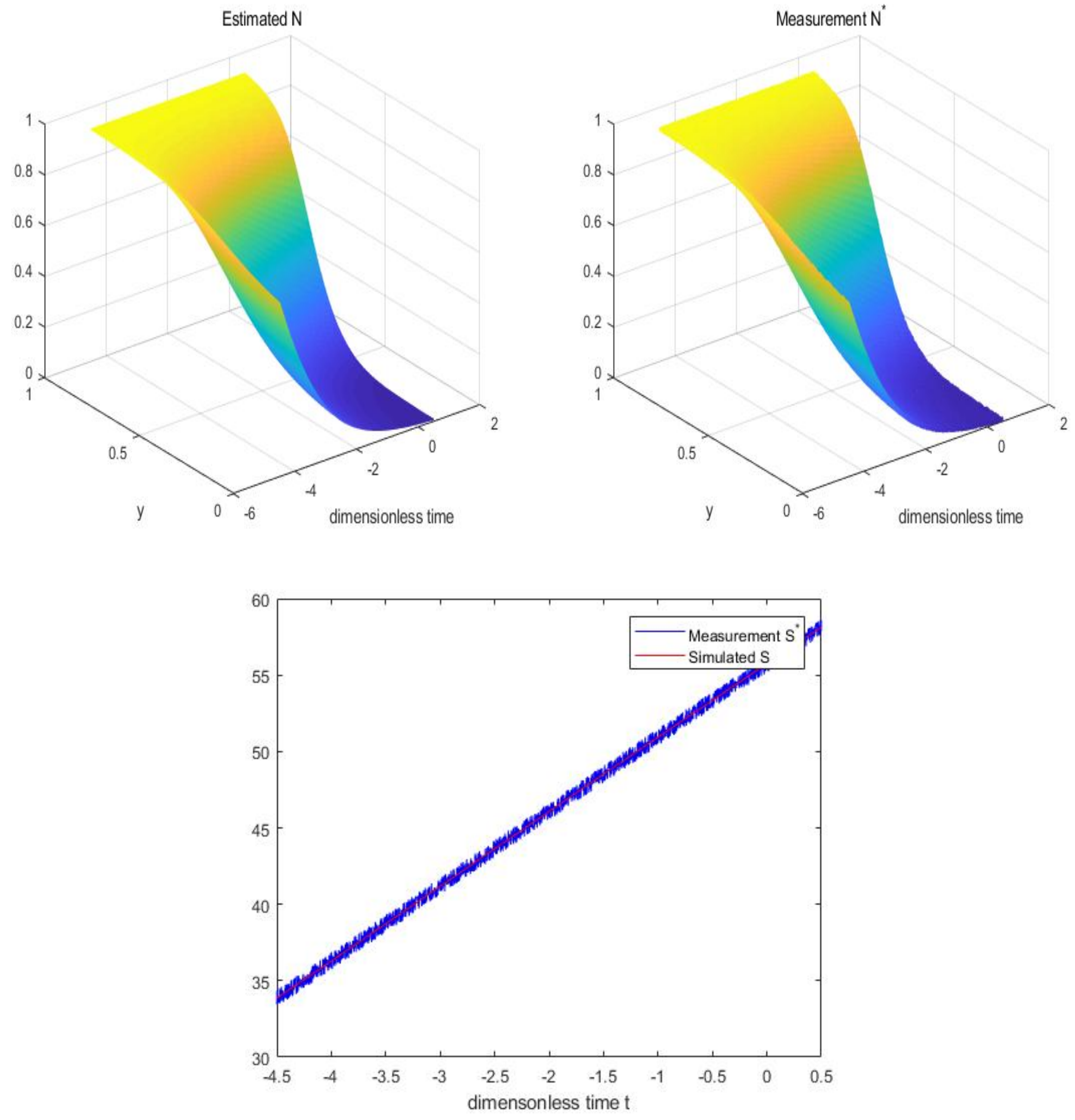
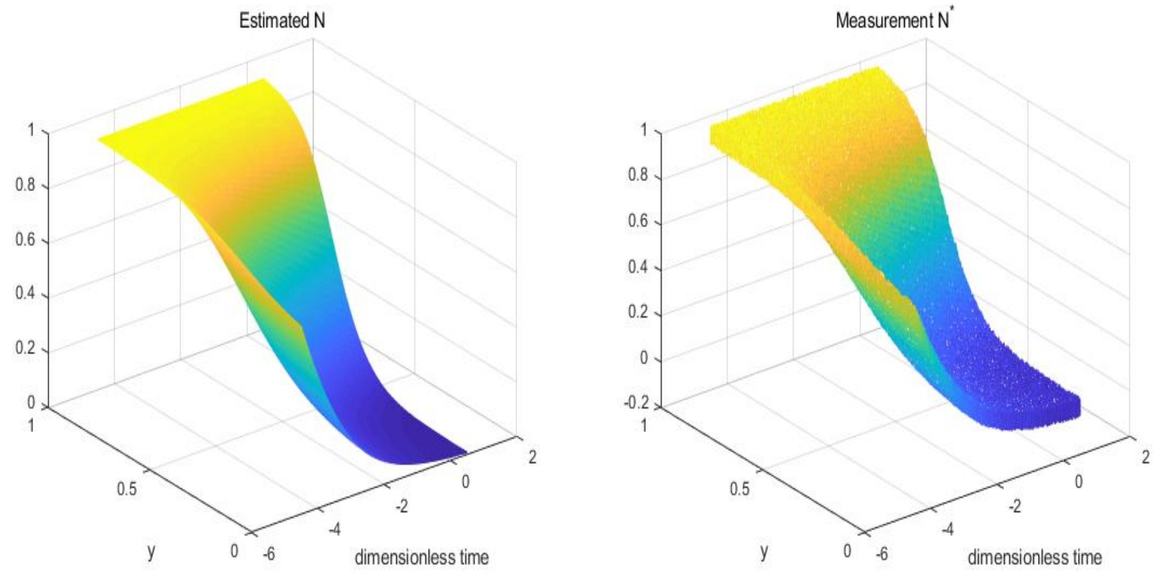


Figure 5.10: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=1%



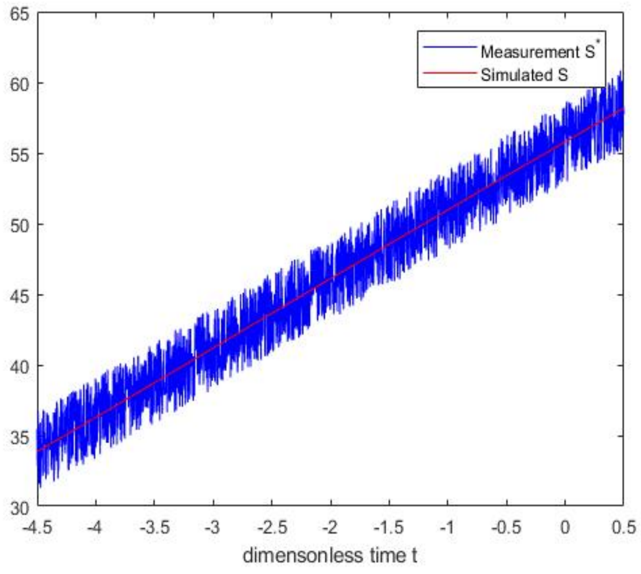


Figure 5.11: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=5%

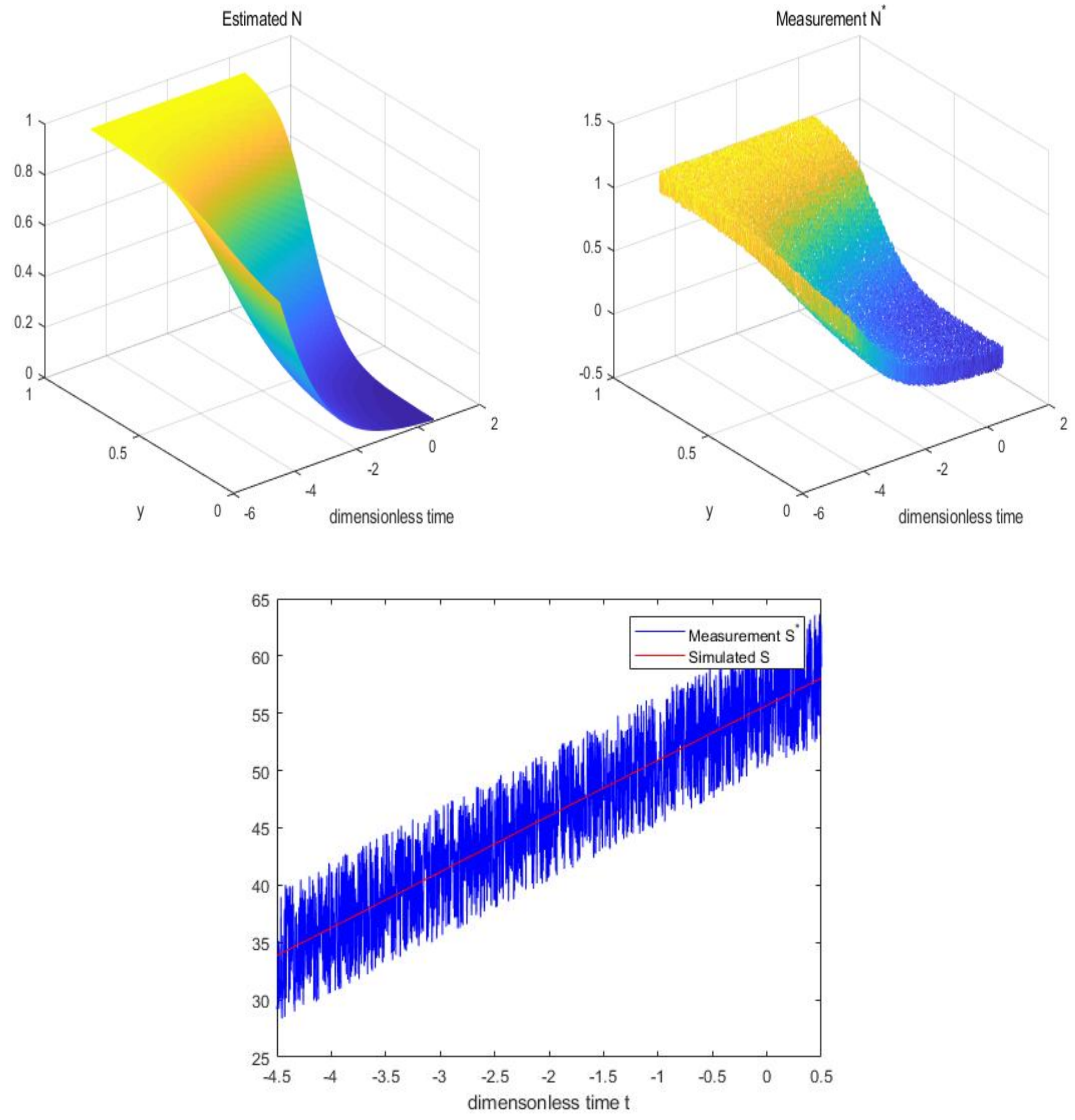


Figure 5.12: Exact parameters=$(0.1, 0.05, 0.9)$, Grid points=80, Time step=0.0025, Noise level=10%

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.0999, 0.0499, 0.8992)$	$(0.07\%, 0.04\%, 0.08\%)$	$4665.83s$
5%	$(0.0966, 0.0472, 0.8976)$	$(3.32\%, 5.45\%, 0.25\%)$	$4587.15s$
10%	$(0.0902, 0.0437, 0.8974)$	$(9.74\%, 12.45\%, 0.27\%)$	$4849.53s$

Table 5.7: Exact parameters=$(0.1, 0.05, 0.9)$, grid points=40, time step=0.005

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.0997, 0.0498, 0.8994)$	$(0.22\%, 0.25\%, 0.05\%)$	$17498.93s$
5%	$(0.0987, 0.0492, 0.8995)$	$(1.25\%, 1.50\%, 0.05\%)$	$19968.62s$
10%	$(0.1013, 0.0516, 0.9021)$	$(1.39\%, 3.35\%, 0.24\%)$	$17813.50s$

Table 5.8: Exact parameters=$(0.1, 0.05, 0.9)$, grid points=80, time step=0.025

Tables 5.7 and 5.8 show the summaries of two sets of simulations with varying number of discretization points in the mesh (and time steps).Gradient Projection Method is not very efficient for our problem. As the tables show that the method is quite robust with respect to relatively high noise level of 10%. With the increase of grid points and time step, the relative error becomes smaller. Simulation times are quite reasonable for a derivative-free method, and they stay roughly the same during the simulations with varying levels of noise. Comparing with Conjugated Gradient Trust Region Method, Gradient Projection Method costs much more time on getting estimated data.

5.3 Analysis

In this section, we compare the results for each method and discuss the benefits and potential problems of each method as well as initial guess selection.

Firstly, let us discuss about the efficiency of those four optimization methods on our tumor growth model by comparing CPU use time of each method. Without any doubt, the Nelder-mead method is the most efficient among those four methods while conjugated gradient trust region method also does well in efficiency. However, pattern search method and Gradient projection method spend over over ten times time on getting the estimated data. In addition, it is obvious that each method spends more time on getting the results with the increase of time steps and grid points.

Secondly we need to compare the accuracy of each method. For accuracy, the Gradient projection method works the best and the relative error of each parameters required by Gradient projection method is below 0.1% when the noise level is 1%. For the three parameters in our model, the estimated c_d has the highest relative error among these three parameters. We think that the reason for the case should be that the exact value of c_d is much smaller than the other two parameters. In addition, the relative errors between exact parameters and estimated parameters is smaller with the increase of time steps and grid points.

Thirdly, we will discuss the initial guess selection of each method. For initial guess selection, gradient free methods works better because the initial guess selection in gradient free methods is more free while we need to choose the initial guess much more close to the exact solution when we apply gradient methods to our model. The reason for this case should be that the calculation of the gradient is more demanding for the data when the computer applies optimization method to the objective function.

Chapter 6

Tumor growth model with an uncertain parameter

The parameter c_d which is the (non-dimensionalized) half saturation concentration in the model depends on temperature and PH levels of the environment. Therefore, we consider a model where c_d is a random parameter whose distribution is known beforehand. For example, c_d could be uniformly distributed, i.e.

$$\xi = c_d \sim \mathscr{U}(c_0 - q, c_0 + q)$$

or follows a normal distribution

$$\xi \sim \mathscr{N}(c_0, \sigma_\xi^2)$$

where the mean value is fixed at c_0 in both cases. We can write the model equation system in the direct problem in an operator form

$$\mathbf{F}(\mathbf{U}, \mathbf{P}, \xi) = \mathbf{0} \tag{6.1}$$

where $\mathbf{U} = [N, C, V, S]$ is the vector of state variables and $\mathbf{P} = [c_c, \sigma]$ is a vector of control variables (consists of the parameters to be identified), and ξ is the random variable.

6.1 Inverse problem with an uncertain parameter

Since the model involves a random parameter, we modify the objective function accordingly. The expectation of the objective function $J(\mathbf{U},\mathbf{P})$ in is given by

$$\hat{J}(\mathbf{P}) = \int J(\mathbf{U},\mathbf{P})p(\xi)\,d\xi \tag{6.2}$$

where $p(\xi)$ is the probability density function of ξ. The inverse problem is to find parameter $\hat{\mathbf{P}}$ such that

$$\hat{J}(\hat{\mathbf{P}}) = \min \hat{J}(\mathbf{P}).$$

6.2 Numerical solution of the problem

Numerical solution procedure will now be obtained with one additional step: evaluation of the objective function values will now involve use of a Monte Carlo method. An approximation of the integral (6.2) using Monte Carlo method is given by

$$\hat{J}(\mathbf{P}) \approx \frac{1}{M}\sum_{i=1}^{M} \hat{J}(\mathbf{U}(\xi_i),\mathbf{P}) \tag{6.3}$$

where $\{\xi_i\}_{i=1}^{M}$ is the sequence of samples of ξ. The algorithm for the solution of the optimization problem is as follows: Let $\mathbf{P}_0$ be an initial guess of the parameters $[c_c,\sigma]$.

Step 1. Solve the direct problem (6.1) to obtain the state vector $\mathbf{U}_n$.

Step 2. Evaluate the objective function $\hat{J}_n = \hat{J}(\mathbf{U}_n, P)$ (and its gradient) by using finite difference approximations and Monte Carlo method.

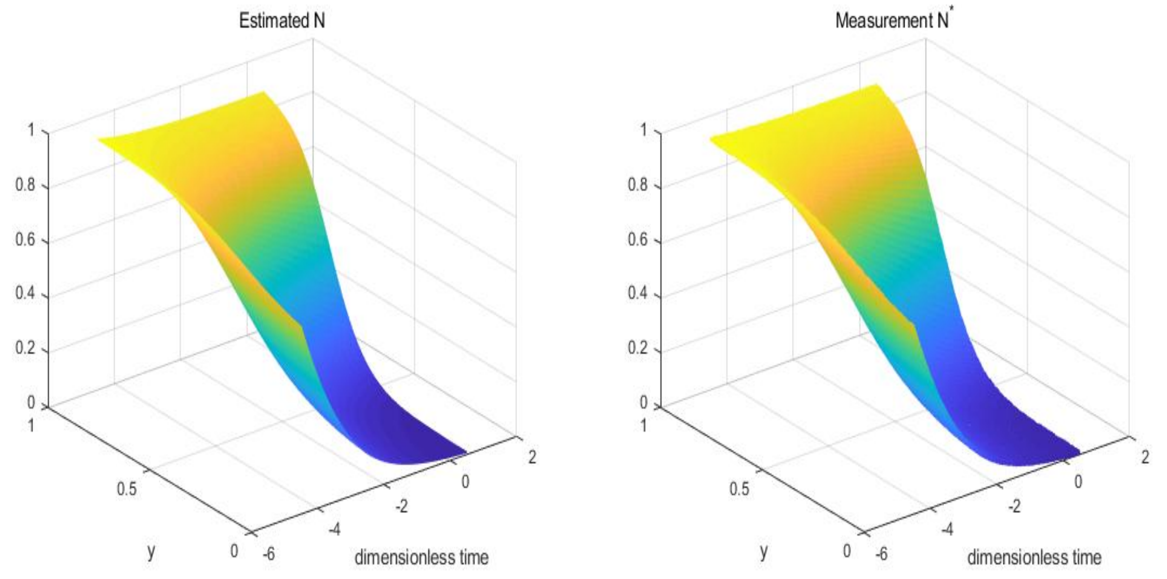
Step 3. Perform the optimization step (use either derivative free or gradient based methods) and get $\mathbf{P}_{n+1}$. Terminate if the stopping criteria are fulfilled, otherwise go to Step 1.

6.2.1 Results of the numerical experiments using derivative free methods

We test the method by using uniformly and normally distributed random parameter ξ in our numerical experiments by Pattern search method and the Nelder Mead method. For the uniform distribution case, we take the mean value of ξ (or c_d) to be 0.05 and $\xi \sim \mathscr{U}(0.03, 0.07)$. For our next experiment, we take $\xi \sim \mathscr{N}(0.05, \sigma_\xi^2)$ with several different values of the variance σ_ξ in order to observe how the distribution of the ξ affects the identification of the parameters.

Pattern Search Method

We present the results of some numerical simulations for normal distribution and uniform distribution using pattern search method in this subsection. In Figures 6.1, 6.2, 6.3 we present results of three simulations with 1%, 5%, and 10% noise levels for normal distribution. In Figures 6.4, 6.5, 6.6 we present results of three simulations with 1%, 5%, and 10% noise levels for uniform distribution. Top left image corresponds to the simulated N (obtained by solving the direct problem by using the identified values of the parameters). Top right image shows the noisy data fed to the optimization routine. Bottom row image shows data S^* (with added noise) and the simulated S. Pattern search method is very effective for this particular problem.



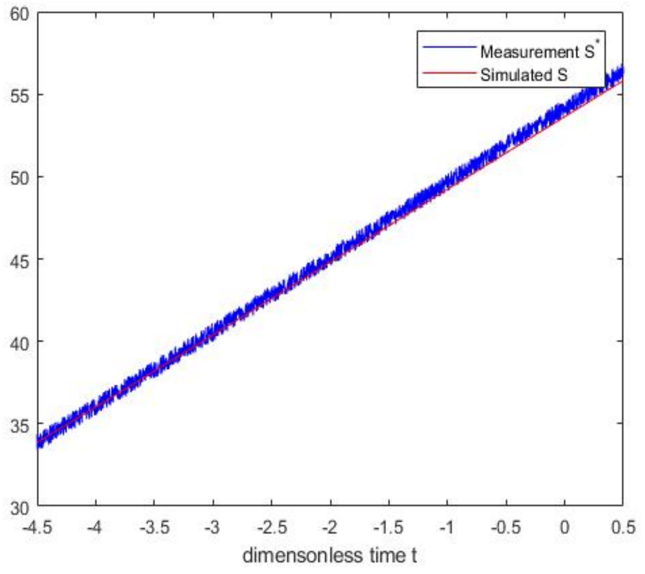
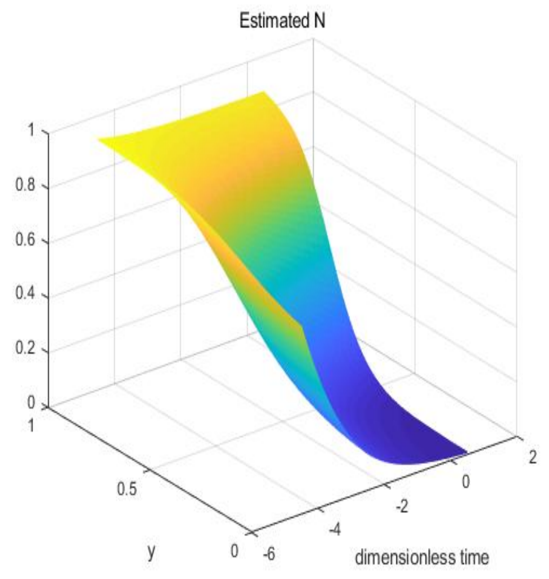
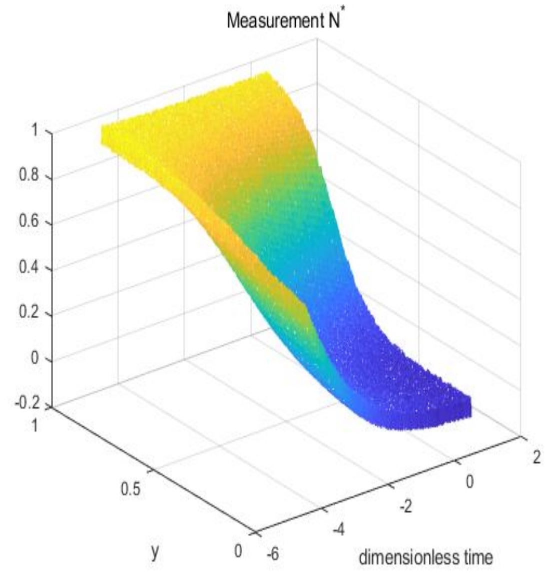


Figure 6.1: Normal distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=1%





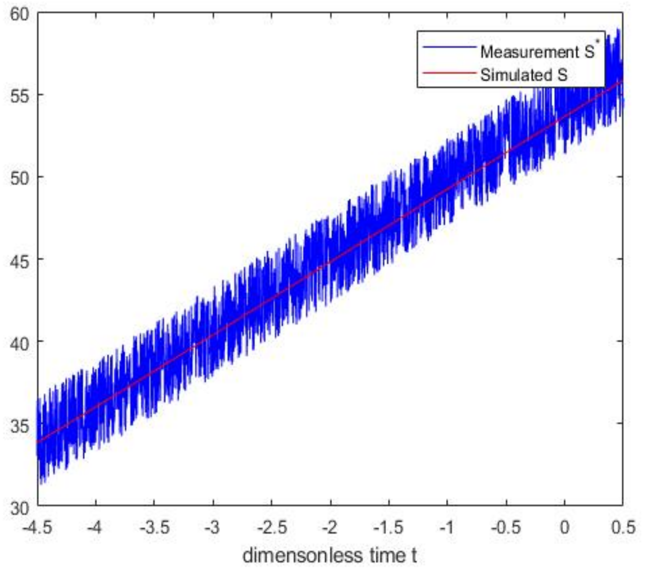


Figure 6.2: Normal distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=5%

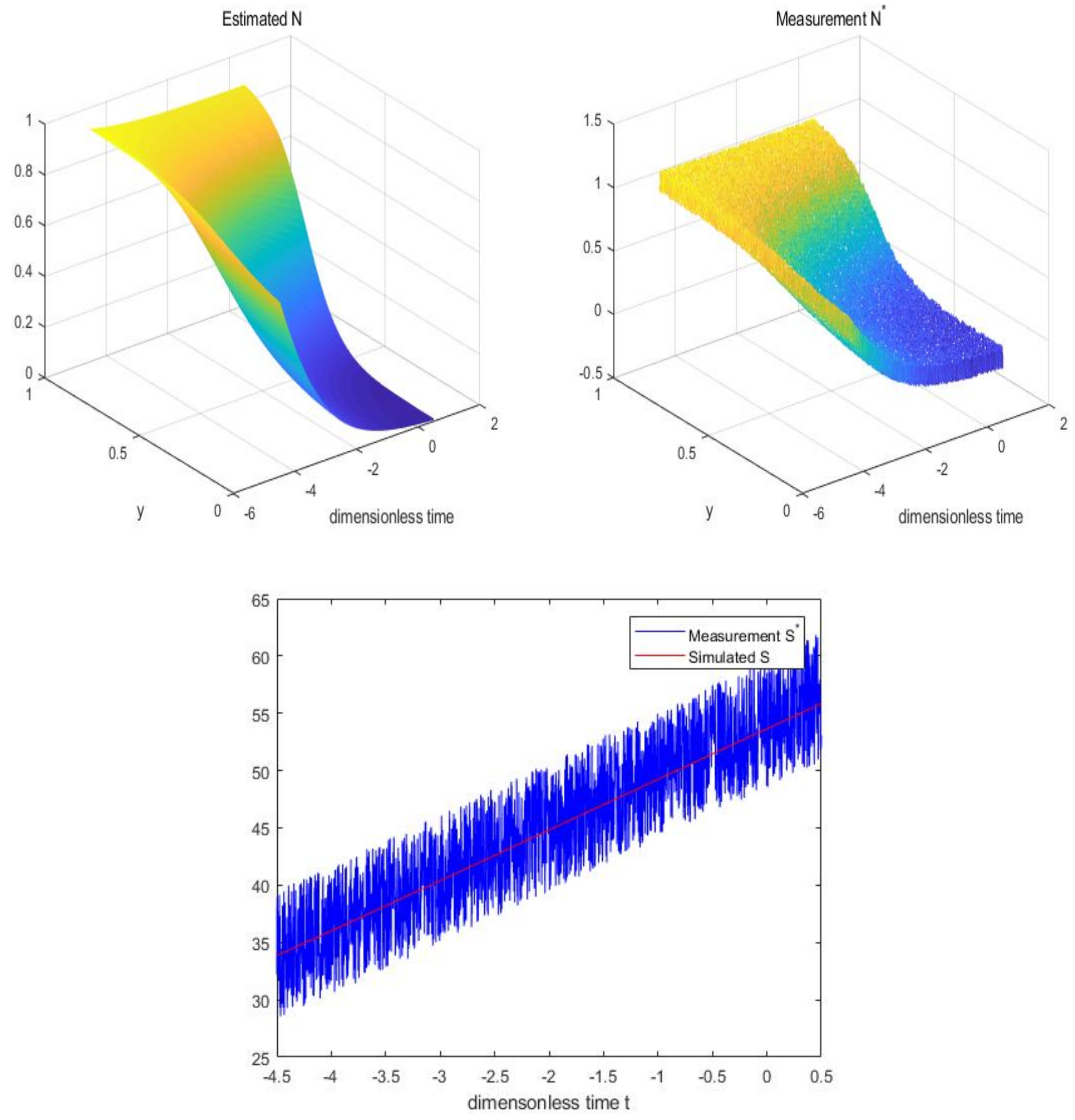


Figure 6.3: Normal distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=10%

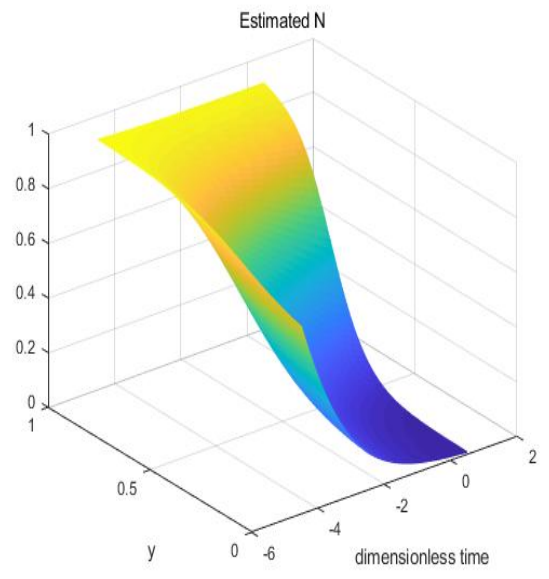
Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.0985, 0.9052)$	$(1.46\%, 0.58\%)$	$8741.59s$
5%	$(0.1610, 0.9687)$	$(61.01\%, 7.63\%)$	$8601.85s$
10%	$(0.0857, 0.8789)$	$(14.21\%, 2.34\%)$	$12788.01s$

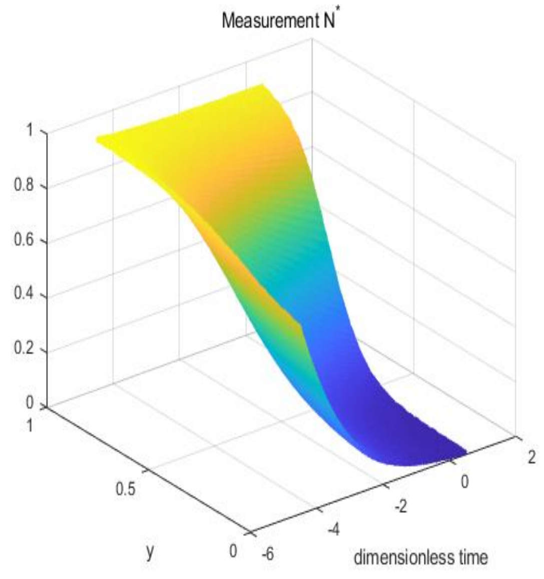
Table 6.1: Exact parameters=$(0.1, 0.9)$, Grid points=40, Time step=0.005, Normal distribution

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.0974, 0.8632)$	$(2.50\%, 4.07\%)$	$30354.50s$
5%	$(0.0975, 0.8642)$	$(2.48\%, 3.97\%)$	$37300.50s$
10%	$(0.0956, 0.8615)$	$(4.33\%, 4.27\%)$	$39793.29s$

Table 6.2: Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Normal distribution

Tables 6.1 and 6.2 show the summaries of two sets of simulations with varying number of discretization points in the mesh (and time steps).Pattern search method is very effective for the normal distribution. As the tables show that the method is quite robust with respect to relatively high noise level of 10%. Simulation times are quite reasonable for a derivative-free method, and they stay roughly the same during the simulations with varying levels of noise. By the results of our numerical experiments, we can find that the accuracy of estimated c_c is not stable with the change of noise level. The reason for the case should also result from the small value of exact c_c. We can also find that the estimated values of N and S are close to the measurement values of N and S. It means that Pattern search method can also be applied in this kind of model with random parameters for normal distribution.





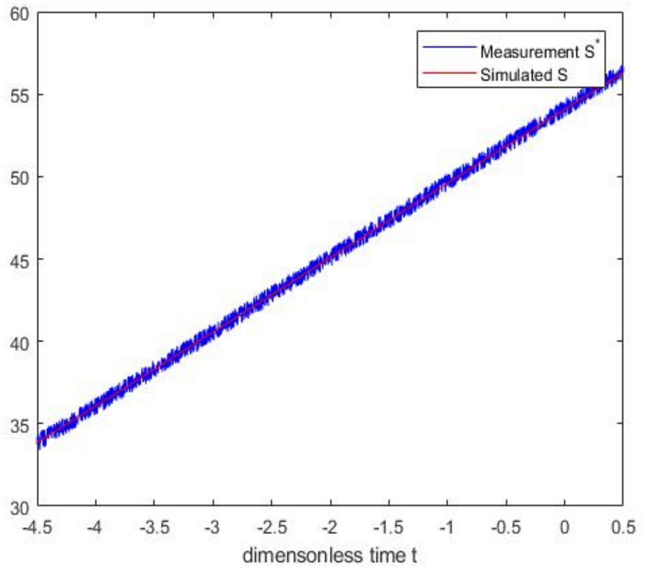
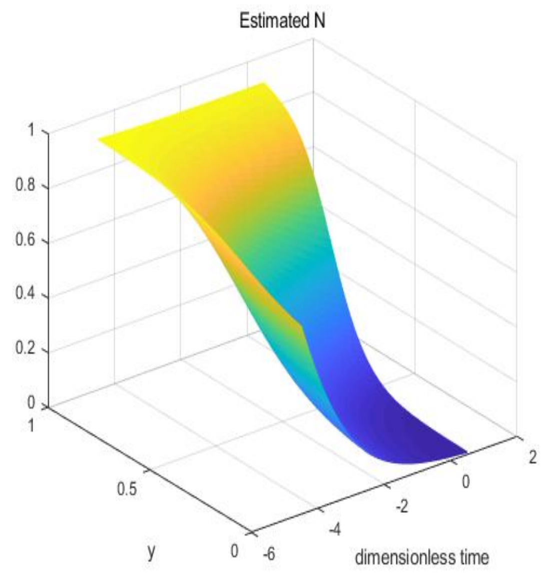
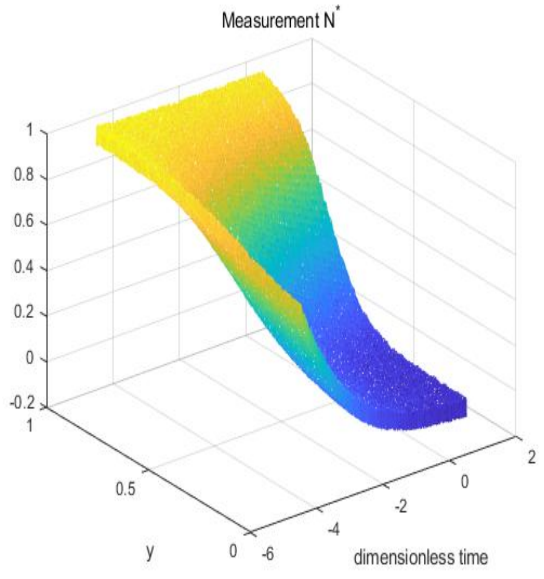


Figure 6.4: Uniform Distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=1%





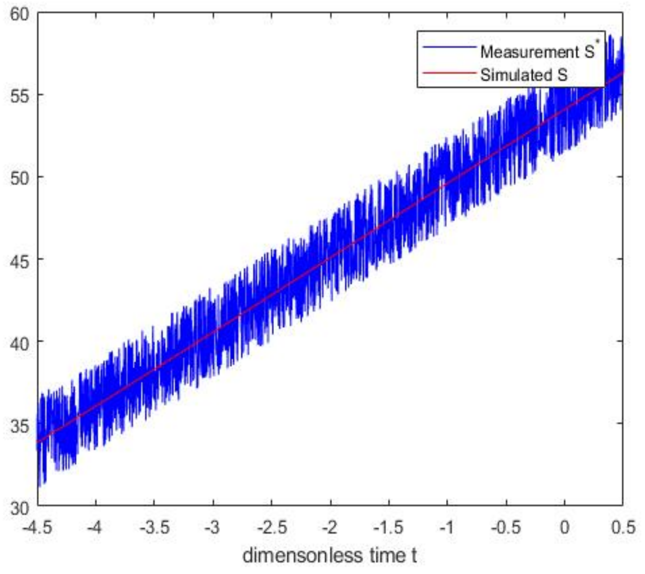
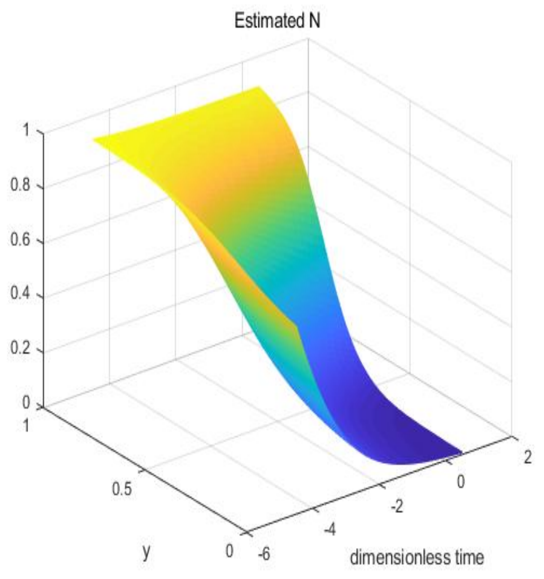
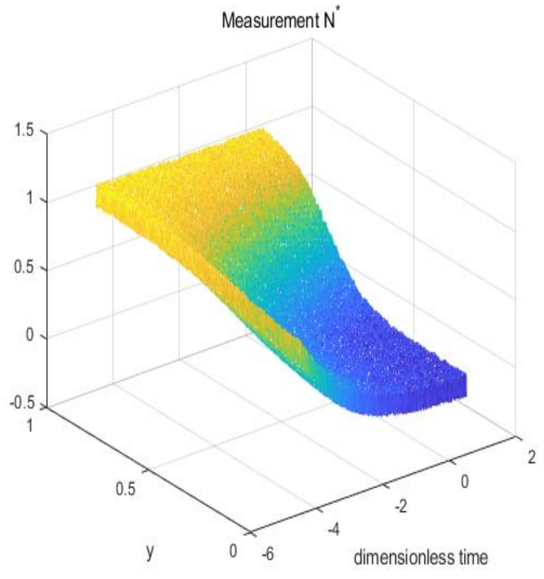


Figure 6.5: Uniform Distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=5%





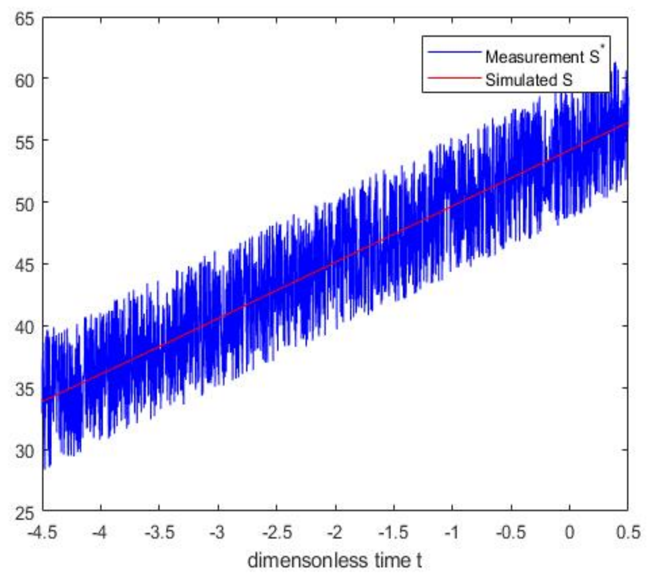


Figure 6.6: Uniform Distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=10%

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.1131, 0.9062)$	$(13.12\%, 0.69\%)$	$7840.67s$
5%	$(0.0975, 0.8828)$	$(2.50\%, 1.90\%)$	$8313.04s$
10%	$(0.1131, 0.9218)$	$(13.12\%, 2.43\%)$	$7086.78s$

Table 6.3: Exact parameters=$(0.1, 0.9)$, Grid points=40, Time step=0.005, Uniform distribution

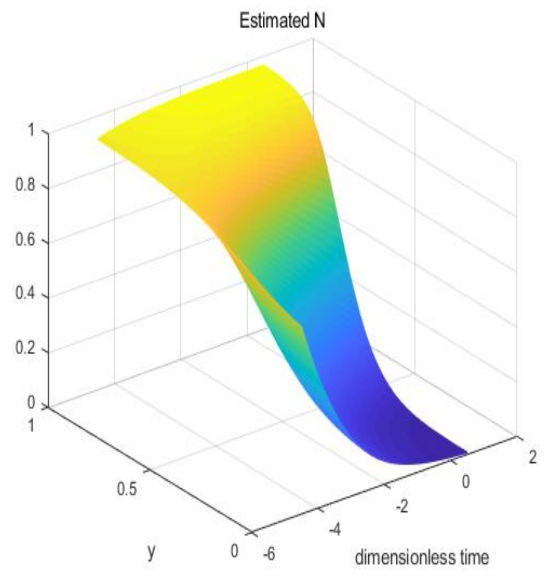
Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.1209, 0.9221)$	$(20.93\%, 2.45\%)$	$30995.20s$
5%	$(0.1109, 0.9243)$	$(10.93\%, 2.48\%)$	$32836.14s$
10%	$(0.0936, 0.8945)$	$(6.37\%, 0.60\%)$	$43990.42s$

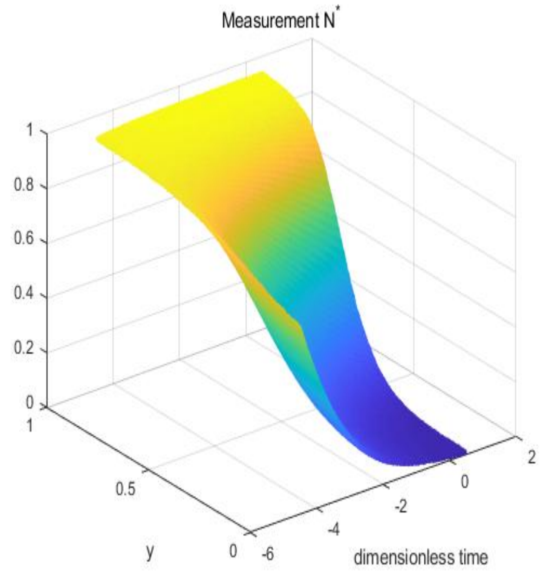
Table 6.4: Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Uniform distribution

Tables 6.3 and 6.4 show the summaries of two sets of simulations with varying number of discretization points in the mesh (and time steps).Pattern search method is also very effective for the uniform distribution. As the tables show that the method is quite robust with respect to relatively high noise level of 10%. Simulation times are quite reasonable for a derivative-free method, and they stay roughly the same during the simulations with varying levels of noise. However comparing with the results for the normal distribution, the accuracy for normal distribution is better than the accuracy for uniform distribution.

The Nelder Mead Method

We present the results of some numerical simulations for normal distribution and uniform distribution using the Nelder Mead method in this subsection. In Figures 6.7, 6.8, 6.9 we present results of three simulations with 1%, 5%, and 10% noise levels for normal distribution. In Figures 6.10, 6.11, 6.12 we present results of three simulations with 1%, 5%, and 10% noise levels for uniform distribution. Top left image corresponds to the simulated N (obtained by solving the direct problem by using the identified values of the parameters). Top right image shows the noisy data fed to the optimization routine. Bottom row image shows data S^* (with added noise) and the simulated S.





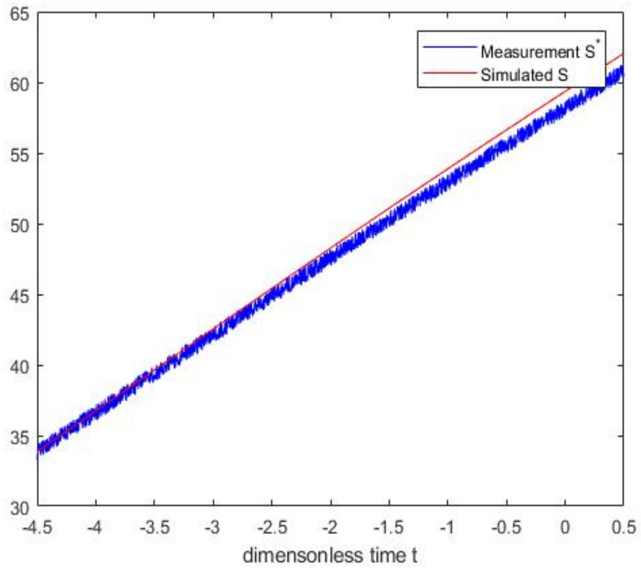
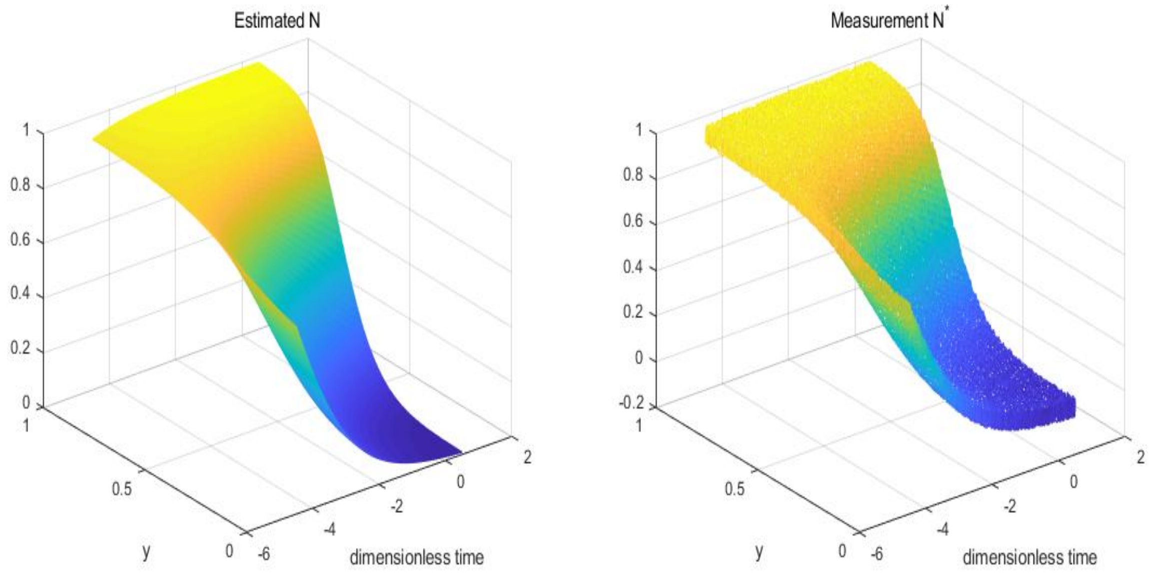


Figure 6.7: Normal Distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=1%



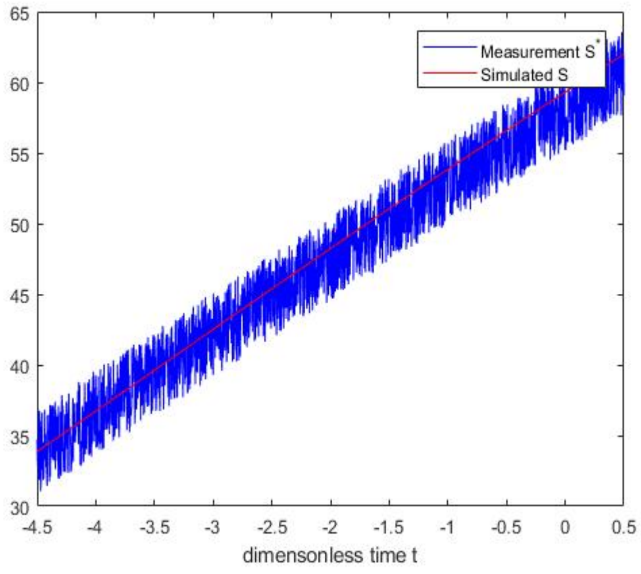
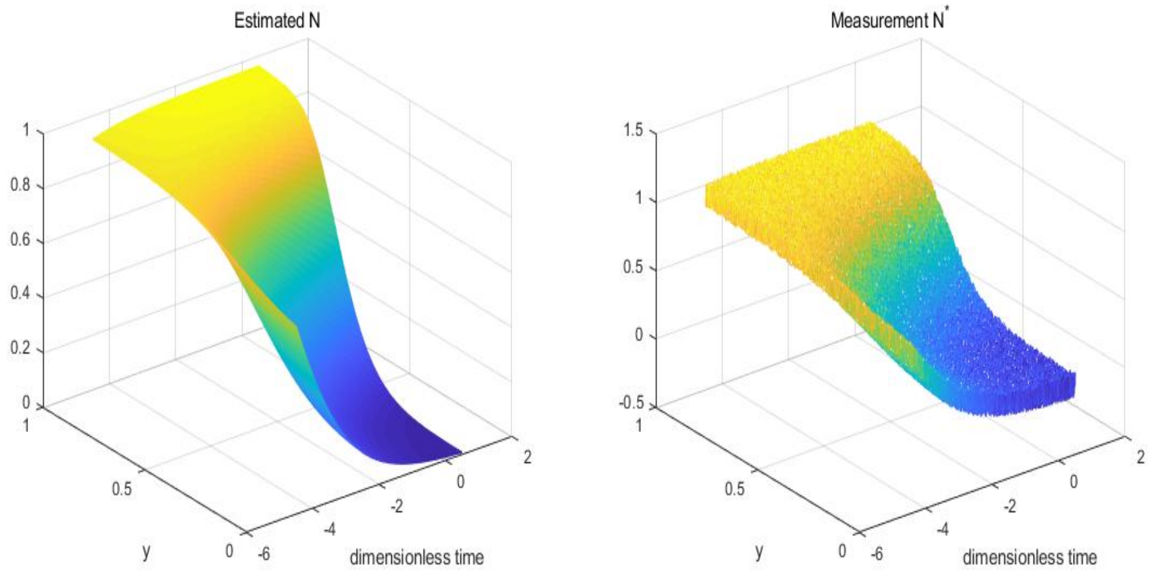


Figure 6.8: Normal Distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=5%



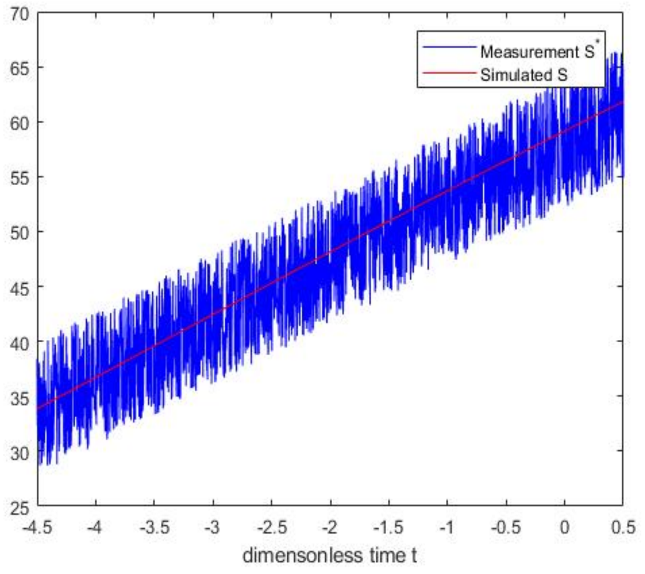


Figure 6.9: Normal Distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=10%

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.1040, 0.9531)$	$(4.00\%, 5.90\%)$	$20257.15s$
5%	$(0.1011, 0.9485)$	$(1.11\%, 5.40\%)$	$20231.10s$
10%	$(0.1265, 0.9647)$	$(26.54\%, 7.19\%)$	$19630.46s$

Table 6.5: Exact parameters=$(0.1, 0.9)$, Grid points=40, Time step=0.005, Normal distribution

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.0969, 0.9510)$	$(3.04\%, 5.67\%)$	$74117.46s$
5%	$(0.1280, 0.9684)$	$(28.08\%, 7.60\%)$	$70582.00s$
10%	$(0.1027, 0.9452)$	$(2.71\%, 5.02\%)$	$90744.48s$

Table 6.6: Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Normal distribution

Tables 6.5 and 6.6 show the summaries of two sets of simulations with varying number of discretization points in the mesh (and time steps).The Nelder Mead Method is very effective for the normal distribution. As the tables show that the method is quite robust with respect to relatively high noise level of 10%. Simulation times are quite reasonable for a derivative-free method, and they stay roughly the same during the simulations with varying levels of noise. By the results of our numerical experiments, we can find that the accuracy of estimated c_c is also not stable with the change of noise level by the Nelder Mead Method. The reason for the case should also result from the small value of exact c_c. We can also find that the estimated values of N and S are close to the measurement values of N and S.

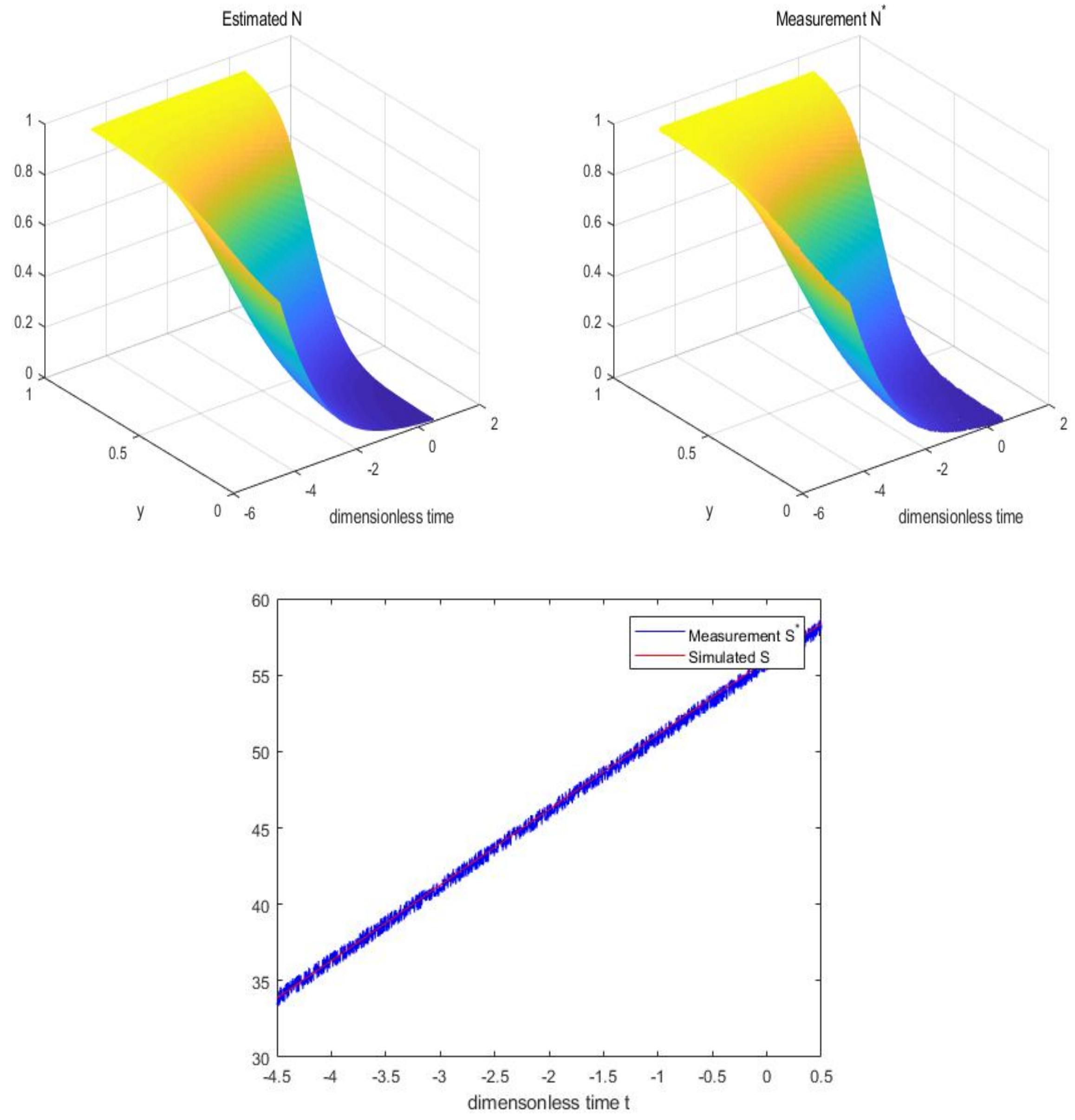
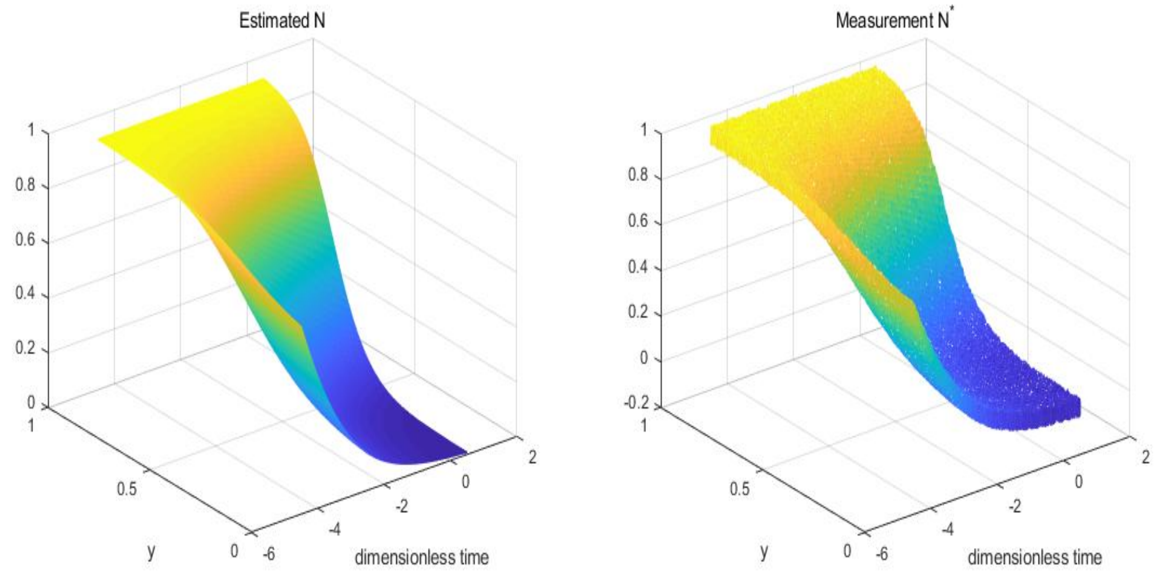


Figure 6.10: Uniform Distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=1%



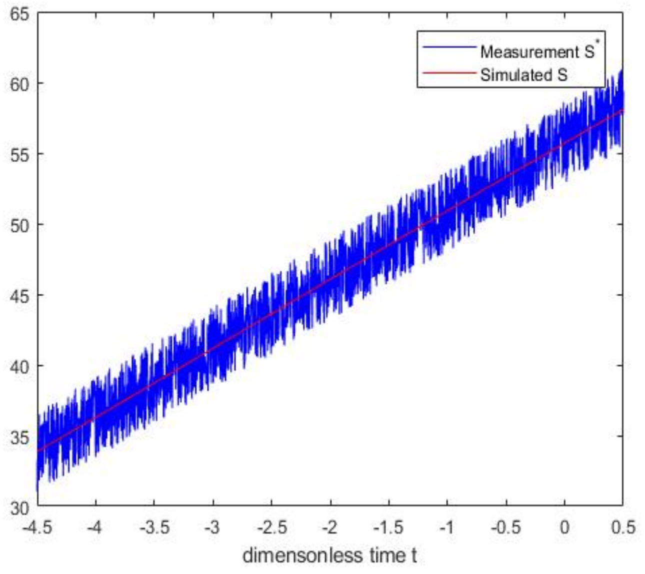
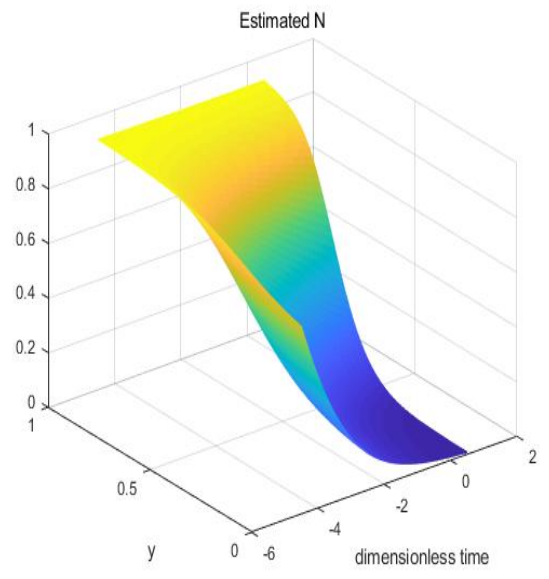
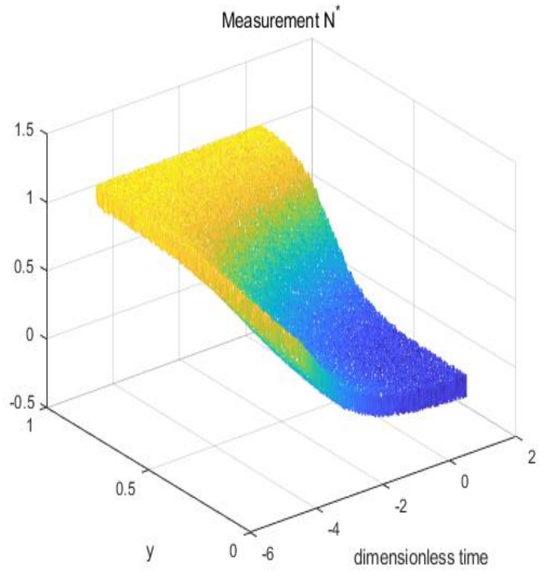


Figure 6.11: Uniform Distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=5%





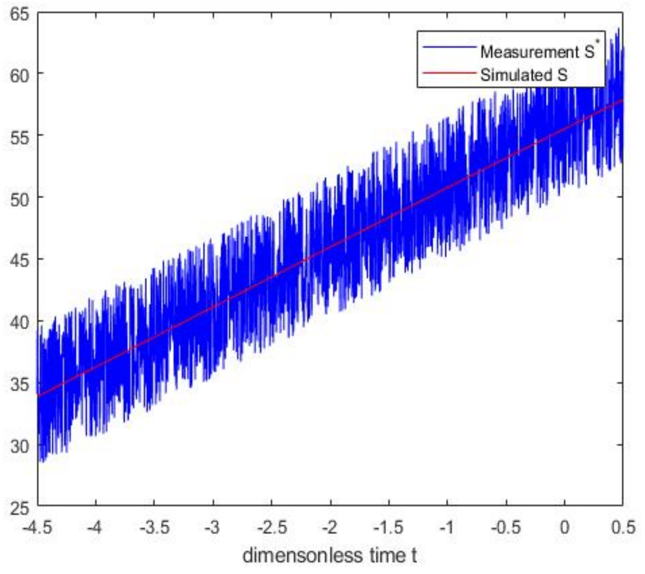


Figure 6.12: Uniform Distribution, Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Noise level=10%

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.1024, 0.8924)$	$(2.41\%, 0.84\%)$	$19580.59s$
5%	$(0.1060, 0.9056)$	$(6.05\%, 0.63\%)$	$19283.51s$
10%	$(0.1039, 0.8921)$	$(3.99\%, 0.88\%)$	$19481.40s$

Table 6.7: Exact parameters=$(0.1, 0.9)$, Grid points=40, Time step=0.005, Uniform distribution

Noise Level	Estimated parameters	Relative error	CPU time
1%	$(0.1015, 0.8856)$	$(1.56\%, 1.60\%)$	$76235.00s$
5%	$(0.1047, 0.9029)$	$(4.74\%, 0.32\%)$	$73875.62s$
10%	$(0.1038, 0.8922)$	$(3.83\%, 0.87\%)$	$74158.00s$

Table 6.8: Exact parameters=$(0.1, 0.9)$, Grid points=80, Time step=0.0025, Uniform distribution

Tables 6.7 and 6.8 show the summaries of two sets of simulations with varying number of discretization points in the mesh (and time steps).The Nelder Mead Method is also very effective for the uniform distribution. As the tables show that the method is quite robust with respect to relatively high noise level of 10%. Simulation times are quite reasonable for a derivative-free method, and they stay roughly the same during the simulations with varying levels of noise. However comparing with the results for the normal distribution, the accuracy for normal distribution is worse than the accuracy for uniform distribution.

6.3 Analysis and Future Work

By the experimental results, we can find that derivative-free methods are very effective for the tumor growth model with random parameters. By comparing CPU time of Pattern search method and the Nelder Mead Method, these two derivative-free methods are similar in efficiency. For accuracy, the Nelder Mead Method works better for the uniform distribution while Pattern search method works better for the normal distribution. Although the results by these two methods have a certain error, the errors are acceptable. For the initial guess, both of two derivative-free methods need an initial guess very close to the exact solution. In this thesis, we only apply derivative-free methods in the tumor growth model with random parameters. In the future, we can also apply more derivative methods such as Gradient projection method and Conjugated gradient trust region method to the model. In addition, we can let more parameters in the model be random and make more numerical experiments.

Bibliography

[1] D.A. Knopoff, D.R. Fernndez, G.A. Torres, C.V. Turner, Adjoint method for a tumor growth PDE-constrained optimization problem, Computers and Mathematics with Applications. 66 (2013) 1104-1119.

[2] D.A. Knopoff, D.R. Fernndez, G.A. Torres, C.V. Turner, A mathematical method for parameter estimation in a tumor growth model, Computers and Mathematics with Applications. 36 (2017) 733-748.

[3] A. K. Louis. Inverse und schlecht gestellte Probleme. Teubner Verlag, Stuttgart,(1989).

[4] F. Kappel and A. Kuntsevich. An implementation of Shors Ralgorithm,Computational Optimization and Applications, 15(2000) 193-205.

[5] J. Borwein and A. S. Lewis. Convex Analysis and Nonlinear Optimization:Theory and Examples. Springer, 2000.

[6] J. V. Burke, A. S. Lewis, and M. L. Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J. Optimization, 15 (2005) 751-779.

[7] J. V. Burke, A. S. Lewis, and M. L. Overton. Two numerical methods for optimizing matrix stability. Linear Algebra Appl., 351–352 117-145, 2002.

[8] J. Nocedal. Updating quasi-Newton matrices with limited storage.Mathematics of Computation, 35(1980)773-782.

[9] J. Nocedal and S. Wright. Numerical Optimization. Springer, 2nd edition, 2006.

[10] D. F. Shanno. Conditioning of quasi-Newton methods for function minimization. Mathematics of Computation, 24(1970) 647-656.

[11] N. Z. Shor. Minimization Methods for Non-Differentiable Functions. Springer Verlag, 1985.

[12] J. Tromp, C. Tape, Q. Liu. Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels", Geophys. J. Int. 160(2005), 195–216.

[13] Y. Cao, S. Li, L. Petzold, R. Serban, Adjoint sensitivity analysis for differential algebraic equations: The adjoint DAE system and its numerical solution, SIAM J. Sci. Comput. 23(2003), 1076–1089.

[14] E.L. Bearer, J.S. Lowengrub, H.B. Frieboes, Y.L. Chuang, F. Jin, S.M. Wise, M. Ferrari, D.B. Agus, V. Cristini, Multiparameter computational modeling of tumor invasion, Cancer Research 69(10)(2009),4493-4501.

[15] N. Bellomo, N.K. Li, P.K. Maini, On the foundations of cancer modelling: selected topics, speculations and perspectives, Mathematical Models and Methods in Applied Sciences 18(04)(2008),593-646.

[16] L. Preziosi, G. Vitale, A multiphase model of tumor and tissue growth including cell adhesion and plastic reorganization, Mathematical Models and Methods in Applied Sciences 21(09)(2011),1901-1932.

[17] H.P. Greenspan, Models for the growth of a solid tumor by diffusion, Studies in Applied Mathematics 51(4)(1972),317-340.

[18] C. Hogea, C. Davatzikos, G. Biros, An image-driven parameter estimation problem for a reactiondiffusion glioma growth model with mass effects, Journal of Mathematical Biology 56(6)(2008),793-825.

[19] J.P. Ward, J.R. King, Mathematical modelling of avascular-tumour growth, Mathematical Medicine and Biology 14(1)(1997),3969

[20] J.P. Ward, J.R. King, Mathematical modelling of drug transport in tumour multicell spheroids and monolayer cultures, Mathematical Biosciences 181(2)(2003),177-207.

[21] B. Perthame, J.P. Zubelli, On the inverse problem for a size-structured population model, Inverse Problems 23(3)(2007) 1037-1052.

[22] R.P. Araujo, D.L.S. McElwain, A history of the study of solid tumour growth: the contribution of mathematical modelling, Bulletin of Mathematical Biology 66(5)(2004),1039-1091.

[23] J.A. Adam, A simplified mathematical model of tumor growth, Mathematical Biosciences 81(2)(1986),229-244.

[24] A. Monazzam, R. Josephsson, C. Blomqvist, J. Carlsson, B. Lngstrm, M. Bergstrm, Application of the multicellular tumour spheroid model to screen PET tracers for analysis of early response of chemotherapy in breast cancer, Breast Cancer Research 9(2007), R45.

[25] H.W. Engl, M. Hanke, A. Neubauer, Regularization of Inverse Problems, in: Mathematics and its Applications, vol. 375, Kluwer Academic Publishers Group, Dordrecht, 1996.

[26] A. Kirsch, An Introduction to the Mathematical Theory of Inverse Problems, in: Applied Mathematical Sciences, vol. 120, Springer-Verlag, New York, 1996.

[27] M. Hinze, R. Pinnau, M. Ulbrich, S. Ulbrich, Optimization with PDE Constraints, in: Mathematical Modelling: Theory and Applications, vol. 23, Springer, New York, 2009.

[28] M. Bertero, M. Piana, Inverse problems in biomedical imaging: modeling and methods of solution, Complex Systems in Biomedicine (2006) 133.

[29] Peter Glynn, Stephen M.Robinson, Numerical optimization,Springer series in operations research, ISBN 0-387-98793-2.

[30] Timothy Sauer, Numerical Analysis ,George Mason University, ISBN 0-321-26898-9.