Bioluminescence Tomography: Sensitivity to Additive Gaussian Noise

Ana Jocelijn Cavero Roeper
s3056910
Leiden University

A thesis submitted for the degree of
MSc Mathematics
30th July 2022

Specialization: Education
External thesis supervisor: dr. Tristan van Leeuwen
Leiden thesis supervisor: dr. Hermen Jan Hupkes

Abstract

Bioluminescence Tomography is a growing field in optical imaging. Governed by the diffusion approximation of the Radiative Transfer Equation, Bioluminescence Tomography corresponds to the inverse problem of reconstructing the source function $q(x)$ from measurements $\Phi(x)$. This inverse problem is ill-posed in the sense of Hadamard. The discrete form of the forward problem is given by $A^{-1}\mathbf{q} = \mathbf{\Phi}$. Although it seems like obtaining $\mathbf{q}$ from $\mathbf{\Phi}$ is straightforward, through numerical experiments it is shown that additive Gaussian noise has a significant effect on reconstruction. Even for the simplest source function tested, i.e. $q(x) = x^2$, regular inversion only suffices in the absence of noise. Both the Truncated Singular Value Decomposition solution and Tikhonov regularization provide a more adequate reconstruction of the source function in the case of additive noise. This is also seen for a source function made up of two Gaussian peaks. Although both methods are able to distinguish the two peaks in a similar fashion, Truncated Singular Value Decomposition may result in misinterpretation of extra peaks due to its forced sinusoidal form. Finally, in the case of a 2D image source function reconstruction we note that the regular inversion performs better than expected but that overall Tikhonov regularization provides the best reconstruction of the source function. Nevertheless, Tikhonov regularization is not perfect and may result in oversmoothing. Further research can be done to improve the reconstruction of source function $q(x)$ from noisy data.

Contents

Chapter 1

Background

A brief introduction to bioluminescence tomography and optical imaging in general is given in this chapter.

1.1 Optical Imaging

X-ray radiography, ultrasonography, X-ray computed tomography (CT) and magnetic resonance imaging (MRI) make up the list of the most common imaging procedures in medicine. Optical imaging is an emerging new addition to this list. [1]

Optical molecular imaging uses light released from either internal sources or administered agents to obtain information about biological processes on a microscopic scale. The optical photons provide nonionizing and safe radiation which in turn provide biochemical information since the optical spectrum is related to molecular conformation. According to [1] optical imaging is highly beneficial in comparison to the other imaging modalities, the results summarized in table 1.

Characteristics	X-ray Imaging	Ultrasound	MRI	Optical Imaging
Soft-tissue contrast	Poor	Good	Excellent	Excellent
Spatial resolution	Excellent	Good	Good	Mixed
Function	None	Good	Excellent	Good
Nonionizing radiation	No	Yes	Yes	Yes
Data acquisition	Fast	Fast	Slow	Fast
Cost	Low	Low	High	Low

Table 1.1: Comparison of Several Imaging Modalities as Presented in [1]

1.2 Bioluminescence Imaging and Fluorescence Imaging

Optical molecular imaging encompasses many different imaging methods, the most common including bioluminescence and fluorescence imaging. Bioluminescence occurs as a result of an enzyme-substrate reaction, namely when the substrate luciferin is oxidized using the catalyst luciferase. Luciferin is an organic substance present naturally in, for instance, fireflies. In the case of fluorescence imaging, an excitation of the molecule fluorophore is required, which results in the production of a lower energy light. Both are non-invasive modalities that allow for the in vivo study of biological processes. However, in fluorescence imaging the excitation required for the reemission of light means that phototoxicity is possible. Bioluminescence imaging (BLI) on the other hand, does not have this issue. It does, however, have a couple of disadvantages compared to fluorescence imaging.

BLI requires the use of a substrate, which fluorescence imaging (FLI), in turn, does not. Furthermore, image acquisition is slower: in FLI it takes seconds whereas in BLI it takes minutes. One of the major limitations to BLI is that it can only image about 1-2cm depth of tissue. Note, however, that this is not a limitation with respect to FLI, which only has an imaging depth of about 1cm. The major advantage of BLI is that it's signal-to-noise ratio is higher. Moreover, BLI is highly sensitive and specific. Like FLI, it has the ability to simultaneously image multiple organisms. According to [2], this technique has allowed researchers to study among others disease progression, protein-protein interaction and treatment efficacy.

1.2.1 Bioluminescence Tomography

Tomography is a branch of imaging that targets reconstruction from indirect measurement of an object under consideration. [3] In this case, Bioluminescence Tomography (BLT) uses an inversion algorithm to determine the distribution of internal bioluminescent sources which enables quantitative monitoring of pathological and physiological changes. [2]

BLT is an optical molecular imaging method that is based on the light propagation model in biological tissues governed by the Boltzmann equation. According to [3] and [2], BLT has been widely applied in preclinical studies. However, the model used is a highly simplified photon propagation model. Moreover, the inverse problem is inherently ill-posed in the sense of Hadamard. This limits the quality of BLT reconstruction to some extent. Nevertheless, BLT has high sensitivity, low cost, and noninvasive characteristics, which makes it a very appealing imaging modality. [2]

This thesis will delve into the inverse problem of reconstructing an image from (noisy) measurements of BLT, beginning with the derivation of the forward problem from the radiative transfer equation (RTE), or Boltzmann equation in chapter 2. From there the inverse problem is formulated and the ill-posedness of the (inverse) problem will be discussed. In chapter 3, a numerical implementation of reconstruction techniques is presented along side numerical experiments to test the quality of these techniques. The results are discussed in chapter 4.

Chapter 2

The Mathematical Model

In this chapter, the forward (diffusion) problem in bioluminescence tomography is derived from the radiative transfer equation. The underlying assumptions used to simplify the model are explained and the corresponding boundary conditions are presented. An analytic solution to the forward problem is given. This chapter concludes with a discussion of the inverse problem and its ill-posedness. For the numerical implementation see chapter 3.

2.1 The Forward Problem

Consider the volume element depicted in Figure 2.1. ds is the differential length element of the cylinder in the direction of photon propagation $\hat{s}$ and dA is the differential area element perpendicular to $\hat{s}$. Next, consider an energy change in the volume element within the differential solid angle $d\Omega$ around $\hat{s}$. Due to the

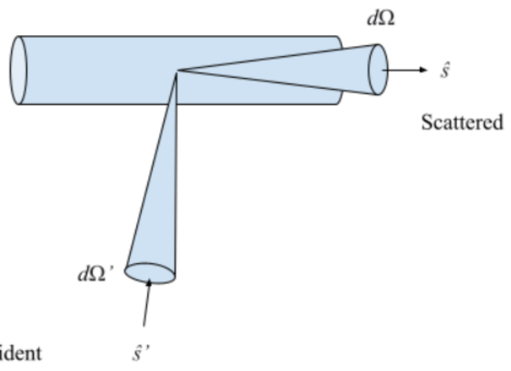


Figure 2.1: Schematic diagram of directional vectors and differential solid angles.

conservation of energy we can write the change in energy in the volume element within the solid angle element per unit time in terms of 4 contributing factors namely divergence (dP_{div}), extinction (dP_{ext}), scattering (dP_{sca}) and the source (dP_{src}), i.e.

$$dP = -dP_{div} - dP_{ext} + dP_{sca} + dP_{src} \tag{2.1}$$

All but the last term can be written in terms of the specific intensity I as given in [1].The contribution from the source dP_{src} is denoted by q. By rewriting the equation to avoid negative terms we get the well known Boltzmann equation:

$$\frac{1}{c}\frac{\partial I}{\partial t} + \hat{s} \cdot \nabla I(r,t,\hat{s}) + (\mu_a + \mu_s) I(r,t,\hat{s}) = \mu_s \int_{\mathbb{S}^2} k(\hat{s} \cdot \hat{s}') I(r,\hat{s}') d\Omega' + q(r,t,\hat{s}) \qquad r \in \Omega \tag{2.2}$$

Definitions of the important terms/symbols used in the RTE and derivation of the forward problem are summarized in the following table:

Symbol	Definition
$\mathbb{S}^2$	unit sphere
Ω	domain
r	position
$\hat{s}$	unit scattered direction vector in $\mathbb{S}^2$
dΩ	differential solid angle element around direction $\hat{s}$
$\hat{s}'$	unit incident direction vector
I	specific intensity at the position r
μ_a	absorption coefficient
μ_s	scattering coefficient
k	scattering kernel function a.k.a. phase function
t	time
c	speed of light through medium
q	source distribution function

Table 2.1: Definition of important terms/symbols

Note from the scattering term $(\mu_s \int_{\mathbb{S}^2} k(\hat{s} \cdot \hat{s}')I(r, \hat{s}')d\Omega')$ a phase function k. The phase function is defined as 'the angular distribution of light intensity scattered by a particle at a given wavelength.' [4] It can be thought of as a probability density function representing the probability that light with direction $\hat{s}'$ is scattered into dΩ around direction $\hat{s}$. [1] The phase function k is symmetrical with respect to interchange of its arguments and respects the normalized condition:

$$\int k(\hat{s}' \cdot \hat{s})d\hat{s}' = 1$$

In [3], Weimin Han and Ge Wang present the Hanyey-Greenstein scattering kernel as a commonly used scattering kernel function given by

$$k_{HG}(s) = \frac{1}{4\pi}\frac{1 - g_a^2}{(1 + g_a^2 - 2g_a s)(3/2)} \qquad -1 \leq s \leq 1$$

where parameter $g_a \in (-1, 1)$ is a measure for anisotropy[1], i.e. $g_a = 0$ corresponds to isotropic scattering (even in all directions).Furthermore, in the steady-state, i.e. time-independent case, we assume

$$\frac{1}{c}\frac{\partial I}{\partial t} = 0$$

For this to hold we require a time-invariant light source, i.e. a continuous wave light beam at constant power. Nevertheless, for a pulsed light source this can still be applicable to "time-integrated physical quantities such as specific energy deposition."[1]

We begin the derivation of our forward problem with equation (2.2). Since the RTE is computationally expensive accurate numerical simulation remains difficult.Note that the RTE contains no less than 6 independent variables! We thus proceed by applying the diffusion approximation.

2.2 The Diffusion Approximation

The Boltzmann equation, also known as the Radiative Transfer Equation (RTE), models photon transport in biological tissue. Nonetheless, at wavelengths of 400 to 750nm light scattering dominates over absorption

[1]The dependence on direction of measurement

in the biological tissues.[5] Therefore, the light transport in this situation can be depicted accurately by the diffusion approximation of the specific intensity, namely:

$$I(r,\hat{s},t) = \frac{1}{4\pi}\Phi(r,t) + \frac{3}{4\pi}J(r,t)\cdot\hat{s} \tag{2.3}$$

where $\Phi(r,t)$ is the fluence rate and $J(r,t)$ is the current density, or flux. The goal is to use (2.3) to obtain the diffusion approximation of the RTE. We begin by introducing several new terms, which will be used in the derivation of the diffusion approximation, noted in table 2.2 below.

Symbol	Term	Equation
$\Phi(r,t)$	fluence rate	$\Phi(r,t) = \int_{\mathbb{S}^2} I(r,t,\hat{s})\mathrm{d}\Omega$
$J(r,t)$	current density, flux	$J(r,t) = \int_{\mathbb{S}^2} \hat{s}I(r,t,\hat{s})\mathrm{d}\Omega$
g	scattering anisotropy	$g = \int_{\mathbb{S}^2} (\hat{s}\cdot\hat{s}')k(\hat{s}\cdot\hat{s}')\mathrm{d}\Omega$
μ_T	interaction coefficient	$\mu_T = \mu_a + \mu_s$
μ_T'	transport interaction coefficient	$\mu_T = \mu_a + \mu_s'$
μ_s'	reduced scattering coefficient	$\mu_s' = \mu_s(1-g)$
l_T'	transport mean free path	$l_T' = \frac{1}{\mu_T'}$

Table 2.2: Terms used in derivation of diffusion approximation

2.2.1 The Scalar Differential Equation

To apply the diffusion approximation of I, we integrate the RTE over the full 4π solid angle and substitute (2.3) to obtain a scalar differential equation. Consider the terms in the RTE as first through fifth from left tot right. In that case, the first term becomes:

$$\int_{\mathbb{S}^2} \frac{1}{c}\frac{\partial I}{\partial t}\mathrm{d}\Omega = \frac{1}{c}\frac{\partial \Phi}{\partial t}$$

Recall that for the case of steady state this term is zero. The second term becomes:

$$\int_{\mathbb{S}^2} \hat{s}\cdot\nabla I(r,t,\hat{s})d\Omega = \int_{\mathbb{S}^2} \nabla\cdot\hat{s}I(r,t,\hat{s})\mathrm{d}\Omega = \nabla\cdot\int_{\mathbb{S}^2}\hat{s}I(r,t,\hat{s})d\Omega = \nabla\cdot J(r,t)$$

We shall see in the vector formulation of the diffusion approximation (see 2.2.2) that

$$J(r,t) = -D\nabla\Phi(r,t)$$

Thus the second term becomes:

$$-\nabla\cdot(D\nabla\Phi(r,t))$$

The third term of the RTE becomes:

$$\mu_T\int_{\mathbb{S}^2} I(r,t,\hat{s})d\Omega = \mu_T\Phi(r,t)$$

where $\mu_T = \mu_a + \mu_s$.
The fourth term becomes:

$$\begin{aligned}\mu_s\int_{\mathbb{S}^2}\int_{\mathbb{S}^2} I(r,\hat{s}')k(\hat{s}\cdot\hat{s}')\mathrm{d}\Omega'\mathrm{d}\Omega &= \mu_s\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}\left(\frac{1}{4\pi}\Phi(r,t)+\frac{3}{4\pi}J(r,t)\right)k(\hat{s}\cdot\hat{s}')\mathrm{d}\Omega'\mathrm{d}\Omega \\ &= \mu_s\frac{1}{4\pi}\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}\Phi(r,t)k(\hat{s}\cdot\hat{s}')\mathrm{d}\Omega'\mathrm{d}\Omega + \mu_s\frac{3}{4\pi}\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}J(r,t)k(\hat{s}\cdot\hat{s}')\mathrm{d}\Omega'\mathrm{d}\Omega\end{aligned}$$

We evaluate

$$\begin{aligned}\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}\Phi(r,t)k(\hat{s}\cdot\hat{s}')\mathrm{d}\Omega'\mathrm{d}\Omega &= \Phi(r,t)\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}k(\hat{s}\cdot\hat{s}')\mathrm{d}\Omega'\mathrm{d}\Omega\\ &= \Phi(r,t)\int_{\mathbb{S}^2}\mathrm{d}\Omega\\ &= 4\pi\Phi(r,t)\end{aligned}$$

and

$$\begin{aligned}\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}J(r,t)k(\hat{s}\cdot\hat{s}')\mathrm{d}\Omega' &= |J(r,t)|\int_{\mathbb{S}^2}\left[\int_{\mathbb{S}^2}k(\hat{s}\cdot\hat{s}')\mathrm{d}\Omega\right]\cos\hat{s}'\mathrm{d}\Omega'\\ &= |J(r,t)|\int_{\mathbb{S}^2}\cos\hat{s}'\mathrm{d}\Omega' = 0\end{aligned}$$

So the fourth term becomes:

$$\mu_s\Phi(r,t)$$

For the fifth term we have:

$$\int_{\mathbb{S}^2}q(r,t,\hat{s})\mathrm{d}\Omega = \frac{1}{4\pi}\int_{\mathbb{S}^2}q(r,t)\mathrm{d}\Omega = q(r,t)$$

Thus we have that

$$\begin{aligned}-\nabla\cdot(D\nabla\Phi(r,t)) + \mu_T\Phi(r,t) &= \mu_s\Phi(r,t) + q(r,t)\\ -\nabla\cdot(D\nabla\Phi(r,t)) + (\mu_T-\mu_s)\Phi(r,t) &= q(r,t)\end{aligned}$$

and the resulting equation is:

$$-\nabla\cdot(D\nabla\Phi(r,t)) + \mu_a\Phi(r,t) = q(r,t) \tag{2.4}$$

Note that since we assume steady-state, the first term = 0 and thus implies that Φ is independent of time, i.e. $\Phi = \Phi(r)$.

2.2.2 The Vector Differential Equation

Next we have the derivation of the equality $J(r,t) = -D\nabla\Phi(r,t)$. This comes from the vector differential equation of the diffusion approximation of the RTE. To obtain this the RTE is multiplied by the direction $\hat{s}$ before evaluation.
The first term becomes:

$$\int_{\mathbb{S}^2}\frac{1}{c}\frac{\partial I}{\partial t}\hat{s}\mathrm{d}\Omega = \frac{1}{c}\frac{\partial J}{\partial t}$$

Since we assume that the fractional change in $J(r,t)$ within the transport mean free path l'_T is small, this time-dependent term becomes negligible. More specifically we assume that

$$\left(\frac{l'_T}{c}\right)\left(\frac{1}{|J(r,t)|}\left|\frac{\partial J(r,t)}{\partial t}\right|\right) \ll 1$$

where $l'_T = \frac{1}{\mu'_T}$ thus we have that

$$\left|\frac{\partial J(r,t)}{c\partial t}\right| \ll \mu'_T|J(r,t)|$$

and thus this term is negligible.
Then the second term becomes:

$$\begin{aligned}\int_{\mathbb{S}^2} \hat{s}(\hat{s}\cdot\nabla I(r,t,\hat{s}))\mathrm{d}\Omega &= \int_{\mathbb{S}^2} \hat{s}(\hat{s}\cdot\nabla(\frac{1}{4\pi}\Phi(r,t)+\frac{3}{4\pi}J(r,t)\cdot\hat{s}))\mathrm{d}\Omega \\ &= \frac{1}{4\pi}\int_{\mathbb{S}^2}\hat{s}(\hat{s}\cdot\nabla\Phi(r,t))\mathrm{d}\Omega + \frac{3}{4\pi}\int_{\mathbb{S}^2}\hat{s}(\hat{s}\cdot\nabla(J(r,t)\cdot\hat{s}))\mathrm{d}\Omega \\ &= \frac{1}{4\pi}\frac{4\pi}{3}\nabla\Phi(r,t)\end{aligned}$$

The third term of the RTE becomes:

$$\mu_T\int_{\mathbb{S}^2}\hat{s}I(r,t,\hat{s})\mathrm{d}\Omega = \mu_T J(r,t)$$

where $\mu_T = \mu_a + \mu_s$.
For the fourth term we have:

$$\begin{aligned}\mu_s\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}\hat{s}(I(r,t,\hat{s})k(\hat{s}\cdot\hat{s},)\mathrm{d}\Omega')\mathrm{d}\Omega &= \mu_s\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}\hat{s}((\frac{1}{4\pi}\Phi(r,t)+\frac{2}{4\pi}J(r,t)\cdot\hat{s}')k(\hat{s}\cdot\hat{s},)\mathrm{d}\Omega')\mathrm{d}\Omega \\ = \mu_s\frac{1}{4\pi}\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}\hat{s}(\Phi(r,t)k(\hat{s}'\cdot\hat{s})\mathrm{d}\Omega')\mathrm{d}\Omega &+ \mu_s\frac{3}{4\pi}\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}\hat{s}((J(r,t)\cdot\hat{s}')k(\hat{s}'\cdot\hat{s})\mathrm{d}\Omega')\mathrm{d}\Omega\end{aligned}$$

We call the first integral (i) and the second (ii). The first integral (i) becomes:

$$\begin{aligned}\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}\hat{s}(\Phi(r,t)k(\hat{s}'\cdot\hat{s})\mathrm{d}\Omega')\mathrm{d}\Omega &= \Phi(r,t)\int_{\mathbb{S}^2}\hat{s}(\int_{\mathbb{S}^2}k(\hat{s}'\cdot\hat{s})\mathrm{d}\Omega')\mathrm{d}\Omega \\ &= \Phi(r,t)\int_{\mathbb{S}^2}\hat{s}\mathrm{d}\Omega = 0\end{aligned}$$

And the second integral (ii) becomes:

$$\int_{\mathbb{S}^2}\int_{\mathbb{S}^2}\hat{s}((J(r,t)\cdot\hat{s}')k(\hat{s}'\cdot\hat{s})\mathrm{d}\Omega')\mathrm{d}\Omega = \int_{\mathbb{S}^2}(\int_{\mathbb{S}^2}\hat{s}k(\hat{s}'\cdot\hat{s})\mathrm{d}\Omega)(J(r,t)\cot\hat{s}')\mathrm{d}\Omega'$$

Using the identity $\hat{s} = \hat{s}'(\hat{s}\cdot\hat{s}') + \hat{s}'\times(\hat{s}\times\hat{s}')$ we can proceed by splitting the internal integral into two integrals.[2] For the first, we apply the definition of scattering anisotropy and obtain that

$$\int_{\mathbb{S}^2}\hat{s}'(\hat{s}\cdot\hat{s}')k(\hat{s}\cdot\hat{s}')\mathrm{d}\Omega = \hat{s}'g$$

and for the second we use that $k(\hat{s}\cdot\hat{s}')$ is azimuthally symmetric about $\hat{s}$, which means that $\int_{\mathbb{S}^2}\hat{s}k(\hat{s}'\cdot\hat{s})\mathrm{d}\Omega$ is parallel with $\hat{s}'$, i.e. the cross-product is zero:

$$\int_{\mathbb{S}^2}\hat{s}'\times(\hat{s}\times\hat{s}')k(\hat{s}\cdot\hat{s}')\mathrm{d}\Omega = \hat{s}'\times((\int_{\mathbb{S}^2}\hat{s}k(\hat{s}'\cdot\hat{s})\mathrm{d}\Omega)\times\hat{s}') = 0$$

We thus obtain for the fourth term that

$$\mu_s\frac{3}{4\pi}g\int_{\mathbb{S}^2}\hat{s}'(J(r,t)\cdot\hat{s}')\mathrm{d}\Omega' = \mu_s\frac{3}{4\pi}\frac{4\pi}{3}gJ(r,t) = \mu_s gJ(r,t)$$

The final term of the RTE becomes:

$$\int_{\mathbb{S}^2}\hat{s}q(r,t,\hat{s})\mathrm{d}\Omega = \frac{1}{4\pi}q(r,t)\int_{\mathbb{S}^2}\hat{s}\mathrm{d}\Omega = 0$$

[2]This identity is an application of the vector triple product, also known as Lagrange's formula. Lagrange's formula states that $a\times(b\times c) = (a\cdot c)\cdot b - (a\cdot b)\cdot c$. Taking $a = c = \hat{s}'$ and $b = \hat{s}$ we get the desired result. (Notice that $\hat{s}\cdot\hat{s} = 1$.)

Hence the vector differential equation is:

$$\begin{aligned}\frac{1}{3}\nabla\Phi(r,t) + \mu_T J(r,t) &= \mu_s g J(r,t)\\ (\mu_T - \mu_s g)J(r,t) &= -\frac{1}{3}\nabla\Phi(r,t)\\ (\mu_a + \mu_s(1-g))J(r,t) &= -\frac{1}{3}\nabla\Phi(r,t)\end{aligned}$$

Rewriting this we have that

$$J(r,t) = -D\nabla\Phi(r,t) \tag{2.5}$$

where

$$\begin{aligned}D &= \frac{1}{3(\mu_a + \mu_s')}\\ \mu_s' &= \mu_s(1-g)\end{aligned}$$

Equation (2.5) is referred to as Fick's Law and D is called the diffusion coefficient.

2.2.3 Explaining Underlying Assumptions

The diffusion equation (2.4) does not depend on vector $\hat{s}$, thus has 4 instead of 6 degrees of freedom. Two assumptions are made to obtain the diffusion equation:

1. The expansion of the radiance is limited to first-order spherical harmonics
2. The fractional change in $J(r,t)$ in one l_T' is much less than 1

The first assumption means that the radiance is nearly isotropic. That means that the magnitude does not vary according to the direction of measurement. The interpretation of the second assumption is that the photon current is "temporally broadened relative to the transport mean free time". Both assumptions hold true in the case of multiple scattering events. Consequently, these two approximations can be translated into a single condition namely that $\mu_t' \gg \mu_a$. In other words, the photons must have gone through a sufficient number of scattering events before being absorbed. Another requirement is that the observation point must be sufficiently far from the sources and the boundaries. In order to improve accuracy we can also apply boundary conditions.

2.3 Deriving the boundary condition

2.3.1 The Refractive-Index Matched Case

To obtain the boundary condition for the diffusion approximation (2.4) we first consider the case of a refractive-index matched situation, such as soft tissue in water. In this case we have that

$$I(r,\hat{s},t) = 0 \qquad\qquad \text{for } \hat{s}\cdot v = 0$$

In other words the scattered direction is orthogonal to v, which is the unit vector normal to the interface pointing into the scattering medium. If we define the z-axis to be along this unit vector, we have that $\hat{s}\cdot v = \cos\theta_i$, where θ_i is the polar angle of $\hat{s}$. (Note that this is a straightforward derivation from the scalar projection formula.) Since radiance $I \geq 0$ an equivalent boundary condition would be

$$\int_{\hat{s}\cdot v} I(r,\hat{s},t)\hat{s}\cdot v\mathrm{d}\Omega = 0 \tag{2.6}$$

Now we can use the diffusion approximation of I to get:

$$\begin{aligned}\int_{\hat{s}\cdot v>0}(\frac{1}{4\pi}\Phi(r,t) + \frac{3}{4\pi}J(r,t)\cdot\hat{s})\hat{s}\cdot v\mathrm{d}\Omega &= \int_{\hat{s}\cdot v>0}\frac{1}{4\pi}\Phi(r,t)\hat{s}\cdot v\mathrm{d}\Omega + \int_{\hat{s}\cdot v>0}\frac{3}{4\pi}(J(r,t)\cdot\hat{s})\hat{s}\cdot v\mathrm{d}\Omega\\ &= \frac{1}{4\pi}\Phi(r,t)\int_{\hat{s}\cdot v>0}\hat{s}\cdot v\mathrm{d}\Omega + \frac{3}{4\pi}\times\int_{\hat{s}\cdot v>0}(J(r,t)\cdot\hat{s})\hat{s}\cdot v\mathrm{d}\Omega\end{aligned}$$

Next we split $J(r,t)$ into its directional components J_x, J_y and J_z. Also, for a smooth boundary we have that $\int_{\hat{s}\cdot v>0} \hat{s}\cdot v \mathrm{d}\Omega = \pi$. Hence the above integrals reduce to

$$\begin{aligned}
&= \frac{1}{4}\Phi(r,t) + \frac{3}{4\pi} \times \int_0^{\frac{\pi}{2}} \int_0^{2\pi} (J_x \sin\theta_i \cos\beta + J_y \sin\theta_i \cos\beta + J_z \cos\theta_i) \sin\theta_i \cos\theta_i \mathrm{d}\beta \mathrm{d}\theta_i \\
&= \frac{1}{4}\Phi(r,t) + \frac{3}{4\pi} \times 2\pi \int_0^{\frac{\pi}{2}} J_z \cos\theta_i \cos\theta_i \sin\theta_i \mathrm{d}\theta_i = \frac{1}{4}\Phi(r,t) + \frac{3}{4\pi}\frac{2\pi}{3} J_z \\
&= \frac{1}{4}\Phi(r,t) + \frac{1}{2} J(r,t)\cdot v
\end{aligned}$$

Convention has us multiplying by 4 to have the final boundary condition as

$$\Phi(r,t) + 2J(r,t)\cdot v = 0 \tag{2.7}$$

However, the interface between soft tissue and air is refractive-index-mismatched and thus we need to adapt the boundary condition. We make use of the Fresnel reflections to adapt our refractive-index-matched boundary conditions to the mismatched case.

2.3.2 Fresnel Reflection

The Fresnel reflection formulas describe the reflection of a portion of incident light observed at a discrete interface due to differences in the refractive indices of the two media (in this case soft tissue and air). Consider the diagram shown in Figure (2.2). Consider an incident beam at $\hat{s}$. The angle made with the

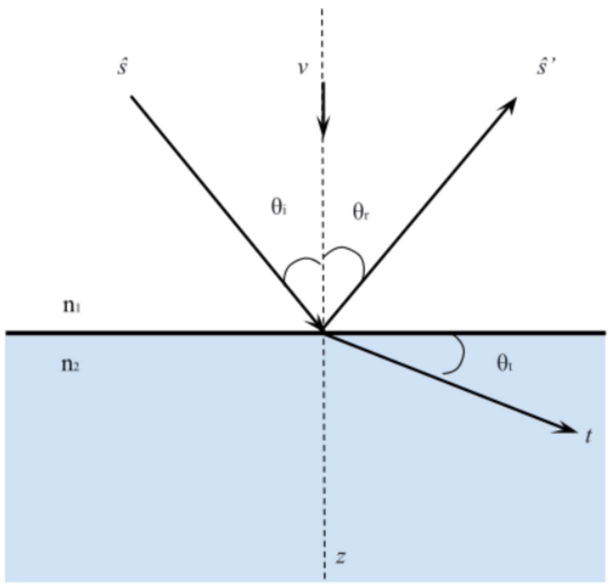


Figure 2.2: 2-D Diagram of angles used in Fresnel Equations. Note that v, $\hat{s}$ and $\hat{s}'$ refer to unit directions.

vector v normal to the boundary is called the incident angle θ_i. Due to the difference in respective refractive indices of the two media, namely n_1 and n_2, some of the light is scattered ($\hat{s}'$) and some is transmitted (t). The fraction of incident light that is reflected is called reflectance.

Reflectance varies depending on the polarization of light. A commonly used coordinate system used to describe the polarization of light is the plane of incidence. The plane of incidence is the plane in which the light propagates before and after reflection or refraction as shown in Figure (2.3). The electric field of polarized light can be either parallel (p) or *senkrecht* (s), meaning perpendicular in German, to the plane of incidence.

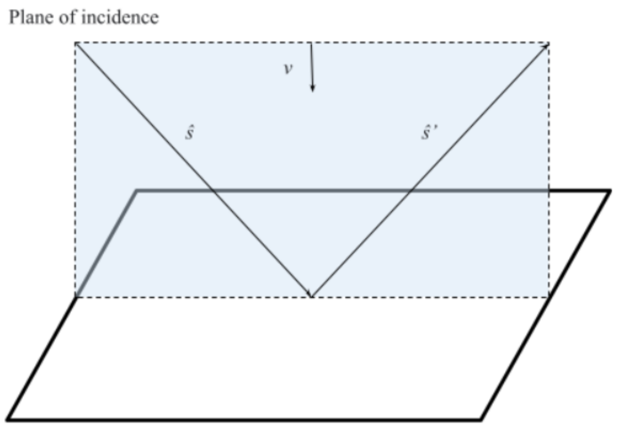


Figure 2.3: Diagram depicting the plane of incidence. Note that v, $\hat{s}$ and $\hat{s}'$ refer to unit directions.

Fresnel defines the reflectance for p- and s-polarized light, assuming that the materials are non-magnetic[3], as follows:

$$R_p = \left| \frac{n_1 \cos\theta_t - n_2 \cos\theta_i}{n_1 \cos\theta_t + n_2 \cos\theta_i} \right|$$
$$R_s = \left| \frac{n_1 \cos\theta_i - n_2 \cos\theta_t}{n_1 \cos\theta_i + n_2 \cos\theta_t} \right|$$

where n is the refractive index for medium 1 and 2 respectively and angles θ_i and θ_t are given in Figure Since we are concerned with natural light, we assume it to be unpolarized. In other words there is an equal amount of s- and p-polarized light and the effective reflectivity is simply the average of the two. In other words, we define the effective reflectance R as the average of R_s and R_p:

$$R = \frac{1}{2}(R_p + R_s)$$

2.3.3 The Refractive-Index Mismatched Case

For the refractive-index-mismatched case we replace the boundary condition equation (2.6) with the following condition instead:

$$\int_{\hat{s}\cdot v>0} I(r,\hat{s},t)\hat{s}\cdot v \mathrm{d}\Omega = \int_{\hat{s}\cdot v<0} R(\hat{s}\cdot v) I(r,\hat{s},t)\hat{s}\cdot v \mathrm{d}\Omega$$

where R is the effective reflectance defined by Fresnel and $\hat{s}\cdot v = \cos\theta_i$.

Once again we substitute the diffusion approximation of the specific intensity I as given in equation (2.3) and evaluate the integrals. The left-hand side is obtained following the method shown in section 2.3.1 and is thus

$$\frac{1}{4}\Phi(r,t) + \frac{1}{2}J(r,t)\cdot v$$

Similarly the right-hand side becomes:

$$\frac{1}{4}R_\phi\Phi(r,t) - \frac{1}{2}R_j J(r,t)\cdot v$$

[3]The assumption of non-magnetic media is a good approximation for optical frequencies and for transparent media at other frequencies.

with

$$\begin{aligned}
R_\phi &:= \int_0^{\frac{\pi}{2}} 2\sin\theta_i \cos\theta_i R(\cos\theta_i)\mathrm{d}\theta_i \\
R_j &:= \int_0^{\frac{\pi}{2}} 3\sin\theta_i (\cos\theta_i)^2 R(\cos\theta_i)\mathrm{d}\theta_i
\end{aligned}$$

Thus, in the diffusive regime, we replace equation (2.7) with

$$\frac{1}{4}\Phi(r,t) + \frac{1}{2}J(r,t)\cdot v = \frac{1}{4}R_\phi\Phi(r,t) - R_j\frac{1}{2}J(r,t)\cdot v$$

This can be rewritten as

$$\begin{aligned}
\frac{1}{4}\Phi(r,t) - \frac{1}{4}R_\phi\Phi(r,t) &= -\frac{1}{2}J(r,t)\cdot v - R_j\frac{1}{2}J(r,t)\cdot v \\
\Phi(r,t) - R_\phi\Phi(r,t) &= -2J(r,t)\cdot v - R_j 2J(r,t)\cdot v \\
\Phi(r,t)(1-R_\phi) &= (1+R_j)(-2J(r,t)\cdot v) \\
\Phi(r,t) &= \frac{1+R_j}{1-R_j}(-2J(r,t)\cdot v)
\end{aligned}$$

In the case of the diffusion approximation we also assume that the ratio of the fluence rate ($\Phi(r,t)$) to the normal component of the flux ($J(r,t)\cdot v$) at the surface to be large. Thus we can define an effective reflection coefficient by

$$\frac{\Phi(r,t)}{3J(r,t)\cdot v} = \frac{2}{3}\frac{1+R_j}{1-R_\phi} := \frac{2}{3}\frac{1+R_e}{1-R_e}$$

to get that [6, p. 2731]

$$R_e = \frac{R_\phi + R_j}{2 - R_\phi + R_j}$$

as follows:

$$\begin{aligned}
\frac{2}{3}\frac{1+R_j}{1-R_\phi} &= \frac{2}{3}\frac{1+R_e}{1-R_e} \\
(1+R_e)(1-R_\phi) &= (1-R_e)(1+R_j) \\
1 - R_\phi + R_e - R_eR_\phi &= 1 + R_j - R_e - R_eR_j \\
2R_e - R_eR_\phi + R_eR_j &= R_\phi + R_j \\
R_e(2 - R_\phi + R_j) &= R_\phi + R_j \\
R_e &= \frac{R_\phi + R_j}{2 - R_\phi + R_j}
\end{aligned}$$

The effective reflection coefficient represents the fraction of the emittance that is reflected and becomes the irradiance, in mathematical terms

$$R_e = \frac{\int_{\hat{s}\cdot v<0} R(\hat{s}\cdot v)I(r,\hat{s},t)\hat{s}\cdot v\mathrm{d}\Omega}{\int_{\hat{s}\cdot v<0} I(r,\hat{s},t)\hat{s}\cdot v\mathrm{d}\Omega} \tag{2.8}$$

thus

$$\int_{\hat{s}\cdot v<0} R(\hat{s}\cdot v)I(r,\hat{s},t)\hat{s}\cdot v\mathrm{d}\Omega = R_e\int_{\hat{s}\cdot v<0} I(r,\hat{s},t)\hat{s}\cdot v\mathrm{d}\Omega$$

Finally, we can write that

$$\frac{1}{4}\Phi(r,t) + \frac{1}{2}J(r,t)\cdot v = \frac{1}{4}R_e\Phi(r,t) + R_e\frac{1}{2}J(r,t)\cdot v \tag{2.9}$$

Using the expression (2.5) we rewrite (2.9) as follows:

$$\begin{aligned}
\frac{1}{4}\Phi(r,t) + \frac{1}{2}J(r,t)\cdot v &= \frac{1}{4}R_e\Phi(r,t) + R_e\frac{1}{2}J(r,t)\cdot v \\
\Phi(r,t) + 2J(r,t)\cdot v &= R_e\Phi(r,t) + R_e 2J(r,t)\cdot v \\
\Phi(r,t) - 2D\nabla\Phi\cdot v &= R_e\Phi(r,t) - R_e 2D\nabla\Phi(r,t)\cdot v \\
\Phi(r,t) - R_e\Phi(r,t) &= R_e 2D\nabla\Phi\cdot v + 2D\nabla\Phi(r,t)\cdot v \\
\Phi(r,t)(1-R_e) &= (1+R_e)(2D\nabla\Phi\cdot v) \\
\Phi(r,t) &= \left(\frac{1+R_e}{1-R_e}\right)2D\nabla\Phi(r,t)\cdot v
\end{aligned}$$

and get the final diffusion approximation boundary condition for the refractive index mismatched case:

$$\Phi(r,t) - 2AD\nabla\Phi(r,t)\cdot v = 0 \tag{2.10}$$

where

$$A = \frac{1+R_e}{1-R_e}$$

If the surrounding medium is air, for which the refractive index η is approximately one, then R_e can be approximated by

$$R(\eta)_e \approx -1.4399\eta^{-2} + 0.7099\eta^{-1} + 0.6681 + 0.0636\eta.$$

[6]

2.4 Summary

We assume that the system is in steady-state. Then the diffusion model is given by:

$$-\nabla\cdot(D\nabla\Phi(r)) + \mu_a\Phi(r) = q(r) \tag{2.11}$$

with boundary condition

$$\Phi(r) + 2AD\nabla\Phi(r)\cdot v = 0 \tag{2.12}$$

where

$$A = \frac{1+R_e}{1-R_e}$$

and R_e is the directionally varying refractive parameter approximated by

$$R_e \approx -1.4399\eta^{-2} + 0.7099\eta^{-1} + 0.636\eta$$

for some refractive index η. For the inverse problem, we have that $\Phi(r)$ on the boundary $d\Omega$ is the measured quantity and μ_a, A, and D are known quantities. We want to solve for $q(r)$, the source distribution function. We begin by tackling the forward problem, to get an idea of the measurements Φ given a source function q; we assume that $q(r)$ is known and solve for $\Phi(r)$.

2.5 Solving the Forward Model

We begin the analysis by assuming a single spatial dimension x where $x \in [0,1]$ and solving the forward model. We take $D(x)$ and $A(x)$ to be constant and scale them to 1. Likewise we scale such that $\mu_a = 1$. We thus have a second order in-homogeneous differential equation of the form:

$$-\Phi''(x) + \Phi(x) = q(x) \tag{2.13}$$

where q is given and Φ is unknown.

2.5.1 Homogeneous Boundary Conditions

We begin with solving the homogeneous case $q(x) = 0$. We have that

$$\Phi''(x) - \Phi(x) = 0 \tag{2.14}$$

with boundary conditions

$$\Phi'(0) = -2\Phi(0) \tag{2.15}$$
$$\Phi'(1) = 2\Phi(1) \tag{2.16}$$

We find that the general solution to (2.14) is given by

$$\Phi(x) = C_1 \cos(kx) + C_2 \sin(kx)$$

We thus have that

$$\begin{aligned}\Phi'(x) &= -C_1 k \sin(kx) + C_2 k \cos(kx)\\ \Phi''(x) &= -C_1 k^2 \cos(kx) - C_2 k^2 \sin(kx)\end{aligned}$$

Thus for (2.14) to hold we have that $-k^2 = 1 \Rightarrow k = \pm i$. The boundary conditions give us that

$$\begin{aligned}\Phi'(0) &= kC_2\\ -2\Phi(0) &= -kC_1\\ kC_2 &= -2C_1\\ C_1 &= \frac{-k}{2}C_2\end{aligned}$$

However,

$$\begin{aligned}\Phi'(1) &= -C_1 k \sin(k) + C_2 k \cos(k)\\ 2\Phi(1) &= 2C_1 \cos(k) + 2C_2 \sin(k)\end{aligned}$$

would imply that

$$\begin{aligned}-C_1 k \sin(k) + C_2 k \cos(k) &= 2C_1 \cos(k) + 2C_2 \sin(k)\\ C_2 k \cos(k) - 2C_2 \sin(k) &= 2C_1 \cos(k) + C_1 k \sin(i)\\ C_1 &= \frac{k\cos(k) - 2\sin(k)}{2\cos(k) - k\sin(k)} C_2\end{aligned}$$

In other words, we obtain that $C_2 = 0 = C_1$ the trivial solution.

2.5.2 The Inhomogeneous Case

To solve the inhomogeneous equation (2.13) we consider the related eigenvalue problem:

$$-\nabla^2 \phi_n(x) + \phi_n = \lambda \phi_n \tag{2.17}$$

with boundary conditions

$$\phi_n'(0) + 2\phi_n(0) = 0, \quad \phi_n'(1) - 2\phi_n(1) = 0.$$

Using the ansatz

$$\phi_n(x) = a_n \cos(k_n x) + b_n \sin(k_n x)$$

we have that

$$\begin{aligned}\phi_n(x) &= a_n \cos(k_n x) + b_n \sin(k_n x)\\ \phi_n'(x) &= -a_n k \sin(k_n x) + b_n k_n \cos(k_n x)\\ \phi''(x) &= -a_n k_n^2 \cos(k_n x) - b_n k_n^2 \sin(k_n x)\end{aligned}$$

substituting back into equation (2.17) we find that

$$\begin{aligned} k_n^2\phi_n + \phi_n &= \lambda\phi_n \\ \lambda &= k_n^2 + 1 \end{aligned}$$

Using the boundary conditions we obtain

$$\begin{aligned} \phi_n'(0) + 2\phi_n(0) &= 0 \\ b_n k_n + 2a_n &= 0 \Rightarrow b_n k_n = -2a_n \\ \phi_n(1) - 2\phi_n(1) &= 0 \\ -a_n k_n \sin(k_n) + b_n k_n \cos(k_n) - 2a_n \cos(k_n) - 2b_n \sin(k_n) &= 0 \end{aligned}$$

Combining the two we get that k_n is a solution of

$$\begin{aligned} (\frac{1}{2}b_n k_n^2 - 2b_n)\sin(k_n) + 2b_n k_n \cos(k_n) &= 0 \\ b_n[(\frac{1}{2}k_n^2 - 2b)\sin(k_n) + 2k_n \cos(k_n)] &= 0 \\ (\frac{1}{2}k_n^2 - 2b)\sin(k_n) + 2k_n \cos(k_n) &= 0 \end{aligned}$$

$$\begin{aligned} 2k_n \cos(k_n) &= (2 - \frac{1}{2}k_n^2)\sin(k_n) \\ 2k_n = (2 - \frac{1}{2}k_n^2)\frac{\sin(k_n)}{\cos(k_n)} &= (2 - \frac{1}{2}k_n^2)\tan(k_n) \\ 0 &= (2 - \frac{1}{2}k_n^2)\tan(k_n) - 2k_n \end{aligned}$$

shown in Figure (2.4).

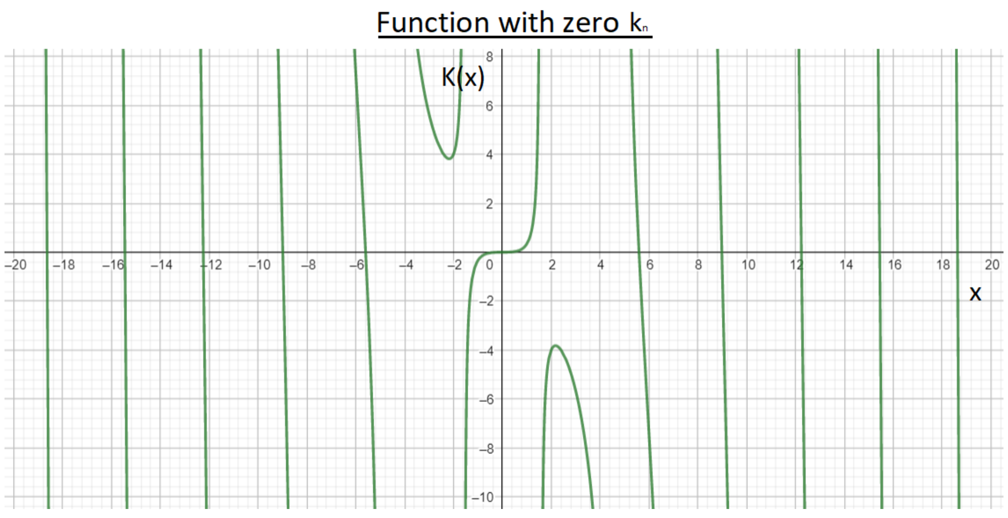


Figure 2.4: Function $K(x) = (2 - \frac{1}{2}x^2)\tan(x) - 2x$ with zeros $x^* = k_n$.

The eigenfunctions are now determined up to a scaling factor

$$\begin{aligned}
\phi_n(x) &= a_n \cos(k_n x) + b_n \sin(k_n x)\\
\phi_n(x) &= -\frac{1}{2} b_n k_n \cos(k_n x) + b_n \sin(k_n x)\\
2\phi_n(x) &= b_n k_n \cos(k_n x) + 2b_n \sin(k_n x)\\
\frac{2}{b_n}\phi_n(x) &= -k_n \cos(k_n x) + 2\sin(k_n x).
\end{aligned}$$

hence, the solution of (2.13) is of the form

$$\Phi(x) = \sum_n c_n \phi_n(x),$$

where we assume that $\int_0^1 |\phi_n|^2 dx = 1$. Thus

$$\begin{aligned}
q(x) &= \sum_n c_n(-\phi_n''(x) + \phi_n(x))\\
q(x) &= \sum_n c_n(k_n^2 + 1)\phi_n(x)
\end{aligned}$$

and we conclude that c_n is given by

$$c_n = \frac{\int_0^1 \phi_n(x) q(x) dx}{k_n^2 + 1}$$

2.6 The Inverse Problem and Well-Posedness

In the case of bioluminescence tomography, we deal not with the forward problem, but the inverse problem. That is, we want to obtain the source function $q(x)$ from measured data $\Phi(x)$. We note that a well-posed problem in the sense of Hadamard satisfies the following three criteria:

- There **exists** a solution u,
- The solution u is **unique**, and
- The solution is **stable**, i.e. the solution depends continuously on the data

We note that the solution to our forward problem is an infinite Fourier sum: we have a linear relation between the data and the source function which we can interpret as a linear operator K such that $Kq = \Phi$, i.e.

$$Kq = \sum_n \frac{\langle \phi_n(x), q(x)\rangle}{k_n^2 + 1}\phi_n(x)$$

We note that the operator K is of the form

$$K = \sum_n \frac{\sigma_n \langle \cdot, v_n\rangle}{u_j}$$

where $\{(u_i, v_i, \sigma_i)\}_{i=1}^{\infty}$ is the singular system of K. Thus $\frac{1}{k_n^2+1}$ are the eigenvalues of operator K. Since k_n grows as $n \to \infty$, we have that $\lim_{n\to\infty}\{\frac{1}{k_n^2+1}\} = 0$ so K is a compact operator. Therefore, the pseudo-inverse of can be expressed as

$$K^{\dagger} = \sum_n (k_n^2 + 1)\langle \phi_n(x), \cdot\rangle \phi_n(x)$$

Here, the Picard condition lets us know whether the problem is ill-posed. The Picard condition states that given a compact operator $K : \mathcal{U} \to \mathcal{F}$ and $f \in \mathcal{F}$, f is in the range of K iff

$$\sum_n \frac{|\langle f, u_j \rangle_{\mathcal{F}}|^2}{\sigma_n^2} < \infty$$

In other words, the inverse of K is only well-defined if the Fourier coefficients ($\langle f, u_j \rangle_{\mathcal{F}}$) decay fast enough, in this case faster than the singular values ($\sigma_n = 1/(k_n^2 + 1)$). Moreover, since in application we deal with approximate data, the collected data is discrete, finite and prone to noise, then Φ may also not be in the range of K for this reason and then a solution to the inverse problem doesn't exist.

2.6.1 The discrete inverse problem

In chapter 3 we discretize the problem to get a discrete forward problem of the form

$$A^{-1}\mathbf{q} = \mathbf{\Phi} \tag{2.18}$$

where $\mathbf{q}$ is the discrete source function and $\mathbf{\Phi}$ are the discrete measurements, both in vector form. A is a finite $n \times n$ real matrix. Note that this matrix equation immediately gives us the solution to the inverse problem of finding the source function $\mathbf{q}$ given data $\mathbf{\Phi}$, assuming that we have all measurements Φ_i, i.e. the solution exists and is unique. Thus, in the case of complete noiseless data, under the aforementioned assumptions, reconstruction of $\mathbf{q}$ is quite simple and well-posed. However, in practice, measurements Φ_i can only be made at the boundary and are subject to noise; in this case we expect the reconstruction of $\mathbf{q}$ to be more problematic. In our experiments presented in section 4 we assume complete measurements. Even so, we can still show that the problem is ill-posed.

Once again we take note of the discrete forward problem 2.18. Recall that in the inverse problem we wish to obtain source function $\mathbf{q}$ from measurements $\mathbf{\Phi}$. Since A^{-1} is invertible, a solution to the inverse problem exists and is given by $\tilde{\mathbf{q}} = A\mathbf{\Phi}$. However, the problem arises when looking at the stability of the solution, i.e. whether the errors in the data get amplified too much. The relative error of our problem is given by

$$\frac{||\mathbf{q} - \mathbf{q}^\delta||}{||\mathbf{q}||} \leq \kappa(A) \frac{||\mathbf{\Phi} - \mathbf{\Phi}^\delta||}{||\mathbf{\Phi}||}$$

and $\kappa(A)$ is the condition number of A, $\mathbf{\Phi}^\delta$ are measurements $\mathbf{\Phi}$ subject to noise and $\mathbf{q}^\delta$ is the noisy reconstruction of source function $\mathbf{q}$. We see that for large number of discretization points the condition number of A increases significantly (matrix A is ill-conditioned) causing errors to blow up.[4] Thus we expect that the reconstruction of $\mathbf{q}$ is highly sensitive to errors in measurement and thus we conclude that the discrete inverse problem is also ill-posed. In the next chapter the discretization is presented as well as methods to deal with this instability, including their implementation in python.

[4]An explicit definition of matrix A is given in equation 3.2.

Chapter 3

Application in Python

This chapter presents the numerical implementation of the inverse problem and motivates the use of the truncated pseudo-inverse and Tikhonov regularization to reconstruct $\mathbf{q}$ from noisy data $\mathbf{\Phi}^{\delta}$. Note that from here on out we shall refer to the number of internal discretization points, i.e. discretization points excluding the two boundary points, as $m = n - 1$ as is implemented in the python code. In Appendix B all the code used for the numerical exploration is presented.

3.1 Approach Han and Wang

In [3], Han and Wang express the inverse problem (2.11) in its variational form. Multiplying by a test function and integrating over domain Ω they go from

$$-\nabla \cdot (D\nabla\Phi) + \mu_a \Phi = q$$

to

$$\int_\Omega -\nabla \cdot (D\nabla\Phi)v + \mu_a \Phi v dx = \int_\Omega q v dx$$

Applying integration by parts we get

$$\int_\Omega D \cdot \nabla\Phi\nabla v dx - \int_\Gamma v D \cdot \nabla\Phi ds + \int_\Omega \mu_a \Phi v dx = \int_\Omega q v dx$$

Han and Wang's boundary condition is slightly different than our (2.12) in that it is non-zero. Instead it reads

$$\Phi + 2AD\nabla\Phi \cdot v = g$$

which they rewrite to get that

$$\begin{aligned} 2AD\nabla\Phi \cdot v &= g - \Phi \\ D\nabla\Phi \cdot v &= \frac{1}{2A}g - \frac{1}{2A}\Phi \end{aligned}$$

Thus, the variational formulation becomes

$$\int_\Omega D \cdot \nabla\Phi\nabla v dx - \int_\Gamma \frac{1}{2A}g - \frac{1}{2A}\Phi ds + \int_\Omega \mu_a \Phi v dx = \int_\Omega q v dx$$
$$\int_\Omega D \cdot \nabla\Phi\nabla v dx + \int_\Gamma \frac{1}{2A}\Phi ds + \int_\Omega \mu_a \Phi v dx = \int_\Omega q v dx + \int_\Gamma \frac{1}{2A} g ds$$

It can be shown, through the Lax-Milgram lemma, that the variational formulation of the inverse problem has a unique solution. In [3], Han and Wang follow the idea of Tikhonov regularization to obtain a minimization problem $\min J(q)$ where the functional $J(q)$ is given by

$$J(q) = \frac{1}{2}||u(q) - g_2||^2_{L^2(\partial\Omega)} + \frac{\epsilon}{2}||q||^2_Q$$

where $Q = L^2(\Omega_0)$ where $\Omega_0 \subset \Omega$ is a region that contains the light source support. For the full details of Han and Wang's method and analysis we refer the reader to [3]. Although in this thesis we treat the simplified case given in 2.13, we choose to implement Tikhonov regularization as our alternative method based on its applicability to the more complex problem as shown in [3]. We begin by using the Finite Difference Method to discretize the problem.

3.2 Discretization of the Problem

To obtain a numerical solution of equation (2.13) the finite difference method can be applied to rewrite the problem in matrix form. We divide the interval $[0, 1]$ into a grid $x_i = i \cdot h$ for $i = 0, \ldots, n$ with $h = \frac{1}{n}$ and discretize the problem. Hence, the finite-difference approximation at gridpoint i is given by:

$$-\frac{1}{h^2}\left(u_{i+1} - 2u_i + u_{i-1}\right) - u_i = q_i, \quad i = 0, \ldots, n.$$

where $u_i = \Phi(x_i)$. To eliminate the ghost points u_{-1} and u_{n+1} we use the boundary conditions. At the left boundary ($i = 0$) we use a central difference approximation:

$$\begin{aligned}
\frac{u_1 - u_{-1}}{2h} + 2u_0 &= 0 \\
u_1 - u_{-1} &= -2u_0 \cdot 2h \\
-u_{-1} &= -4hu_0 - u_1 \\
u_{-1} &= 4hu_0 + u_1 \\
\Rightarrow \frac{1}{h^2}(u_1 - 2u_0 + 4hu_0 + u_1) - u_1 &= q_1
\end{aligned}$$

Similarly, at the right boundary we have that

$$\begin{aligned}
\frac{u_{n+1} - u_{n-1}}{2h} - 2u_n &= 0 \\
u_{n+1} - u_{n-1} &= 2u_n \cdot 2h \\
u_{n+1} &= 4hu_n + u_{n-1} \\
\Rightarrow \frac{1}{h^2}(u_{n-1} - 2u_n + 4hu_n + u_{n-1}) - u_n &= q_n
\end{aligned}$$

This yields a system of $n + 1$ equations in $n + 1$ unknowns. We can thus rewrite our *inverse* problem in matrix form as

$$A\mathbf{\Phi} = \mathbf{q} \tag{3.1}$$

where $A \in \mathbb{R}^{(n+1)\times(n+1)}$ is given by

$$A = \frac{1}{h^2}\begin{pmatrix}
4h-2 & 2-h^2 & 0 & \cdots & \cdots & \cdots & 0 \\
1 & -(h^2+2) & 1 & 0 & \cdots & \cdots & 0 \\
0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\
\vdots & \ddots & & \ddots & \ddots & \ddots & 0 \\
0 & \cdots & \cdots & 0 & 1 & -(h^2+2) & 1 \\
0 & \cdots & \cdots & \cdots & \cdots & 2 & 4h-(h^2+2)
\end{pmatrix} \tag{3.2}$$

$\mathbf{\Phi} \in \mathbb{R}^{n+1} = (u_i)$ denotes the measurements at position x_i and $\mathbf{q} \in \mathbb{R}^{n+1} = (q_i)$ is the discretized source function with $q_i = q(x_i)$. Note that this is a direct expression of our inverse problem. In fact, the forward problem would read

$$A^{-1}\mathbf{q} = \mathbf{\Phi} \tag{3.3}$$

In numerical computations of solving the inverse problem, the inversion of A^{-1} is avoided by using A directly. The code used to implement matrix A for given number of internal discretization points m is given in Appendix B.1 (lines 18-49). For the discretization of source function q `discrete(function,m)` given in lines 51-61 was used. In Appendix A.1, a test for the correct implementation of matrix A is given.

3.3 Truncated Pseudo Inverse

Let our measurements be subject to noise. Since the reconstruction of $\mathbf{q}$ is highly sensitive to errors, particularly those associated with small singular values, instead of inverting A^{-1} in 3.3 to obtain the solution of the inverse problem as in (3.1), we use a truncated pseudo-inverse of A^{-1} to get

$$A_k\mathbf{\Phi}^\delta = \mathbf{q}^\delta$$

with $A_k = (A^{-1})_k^\dagger = V_k\Sigma_k^{-1}U_k^*$ where $V_k = (v_1, .., v_k), U_k = (u_1, .., u_k)$ and Σ_k contains the k largest singular values of A^{-1}. This is done to avoid the issue of noise components being blown up by small singular values, by removing them altogether. Note that this is the same as decomposing A into its singular system and removing the largest singular values.

In theory, the use of the truncated pseudo-inverse should allow for an improvement in the reconstruction of the true $q(x)$; however, if too many singular values are removed then a lot of data is lost in the reconstruction, thus it is important to choose an optimal truncation parameter. In other words, although the truncated pseudo inverse defines a unique solution, it may not be stable as $||A^{-1}||_2||A_k||_2 = \frac{\sigma_1}{\sigma_k}$ may still be large.

Note that our solution $\mathbf{q}^\delta$ is given by

$$\mathbf{q}^\delta = V_n\Sigma_n^{-1}U_n^*\mathbf{\Phi}^\delta = \sum_{i=1}^{k}\frac{\langle u_i, \mathbf{\Phi}^\delta\rangle}{\sigma_i}v_i$$

where $\{U_n, \Sigma_n, V_n^*\}$ denotes the singular system of A^{-1}, i.e. $A^{-1} = U_n\Sigma_nV_n^*$. Hence, we note the component in $\mathbf{\Phi}^\delta$ corresponding to v_i is amplified by σ_i^{-1}. Thus if $\mathbf{\Phi}$ has noise components that correlate with v_i's corresponding to very small singular values, these noise components get amplified. If the Picard condition is satisfied this should not be a problem. The discrete Picard condition for our problem $A^{-1}\mathbf{q} = \mathbf{\Phi}$ is satisfied if for the vector $\mathbf{\Phi} \in \mathbb{R}^m$ the Fourier coefficients $|\langle U_i, \mathbf{\Phi}\rangle|$ decay faster than the singular values σ_i of A^{-1}, where U_i denotes the left singular vectors of A^{-1}. Python was used to plot $|\langle U_i, \mathbf{\Phi}\rangle|$,$|\langle U_i, \mathbf{\Phi}^\delta\rangle|$ and the singular values of A^{-1} (Appendix B.2 lines 70-82, B.1 lines 108-116).

The truncated pseudo-inverse solution was implemented using the function `best_trunc(A,phi_del,q)` shown in Appendix B.1 lines 81-106. The code was constructed such that this optimal parameter was picked by keeping the reconstruction with the smallest scaled error. The error, truncated solution, truncation parameter and number of singular values truncated are returned, such that a comparison can be made with the suggested truncation by the Picard condition.

3.4 Reconstruction using Tikhonov regularization

Another regularization method that can be applied to solve the noisy system is Tikhonov regularization. In this case, the regularized solution to (2.18) is given by

$$\mathbf{q}^\delta = \sum_{i=1}^{k}\frac{\sigma_i\langle u_i, \mathbf{\Phi}^\delta\rangle}{\sigma_i^2 + \alpha}v_i$$

where $\{(u_i, v_i, \sigma_i)\}_{i=1}^k$ is the singular system of A^{-1}. Thus, note that for noiseless data and $\alpha = 0$ we expect to get the same results as for the regular inversion.

Tikhonov regularization has a corresponding variational problem, namely

$$\min_q ||A^{-1}\mathbf{q} - \mathbf{\Phi}||_2^2 + \alpha||\mathbf{q}||_2^2 \tag{3.4}$$

Here we see explicitly the trade-off between data fidelity (given by $||A^{-1}\mathbf{q} - \mathbf{\Phi}||_2^2$) and the regularization $||\mathbf{q}||_2^2$. Once again the choice of regularization parameter α is important. A similar approach to choosing the truncation parameter was used. The corresponding normal equations of (3.4) are

$$\mathbf{q}^\delta = V_k(\Sigma_k^2 + \alpha I)^{-1}\Sigma_k U_k^* \mathbf{\Phi}$$

This was implemented in Python in a function called `Tikh_reg` and then looped over increasing values of α, starting with $\alpha = 10^{-16}$ and increasing in increments of power of 10 until $\alpha = 1$, using the function `best_Tikh`. The code for Tikhonov regularization is shown in Appendix B.1 (lines 119 - 136).

Chapter 4

Results

The numerical experiments and their results are presented and discussed in this chapter.

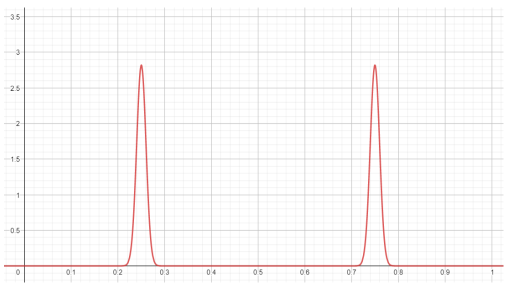
4.1 The Experiments

4.1.1 A Smooth Source Function

The numerical exploration begins with defining a simple source function q on interval $[0,1]$, namely $q(x) = x^2$ and computing the corresponding ideal measurements $\mathbf{\Phi}$. We let these in turn be subject to additive Gaussian noise,i.e. $\mathbf{\Phi}^\delta = \mathbf{\Phi} + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \delta^2)$, which was implemented in python by the function `noisy_add(phi,delta)`. This function takes as input the correct measurements $\mathbf{\Phi}$ and adds a vector of random numbers of the same length (Appendix B.1 lines 75-79); the random numbers are sampled from the standard normal distribution of mean 0 but with variance δ^2. Reconstruction was done 3 times: (1) computing $\mathbf{q}^\delta$ directly as $A\mathbf{\Phi}^\delta$, (2) implementing the truncated pseudo-inverse solution with $A_k = (A^{-1})^\dagger_k = V_k \Sigma_k^{-1} U_k^*$ and (3) using Tikhonov regularization given in section 3.4. For each method, the scaled reconstruction error $\frac{||\mathbf{q}-\mathbf{q}^\delta||}{||\mathbf{\Phi}^\delta||}$ was computed by taking the ratio of numpy's `linalg.norm` function with $\mathbf{q} - \mathbf{q}^\delta$ and $\mathbf{\Phi}^\delta$ as input. This was repeated 5 times (seed=0,1,2,3,4) using the code in Appendix B.3, for noise levels corresponding to $\delta = 0, 0.1, 0.01, 0.001$ and 0.0001 for each $m = 10, 20, 100,$ and 200 by running over python code given in Appendix B.2 with the corresponding inputs. The (average) scaled errors and reconstruction parameters were stored as a data frame using the pandas package.

4.1.2 2 Gaussian Peaks

In image reconstruction it is not only interesting to minimize the reconstruction error but also to distinguish between objects.


(a) $\mu = \{0.25, 0.75\}$

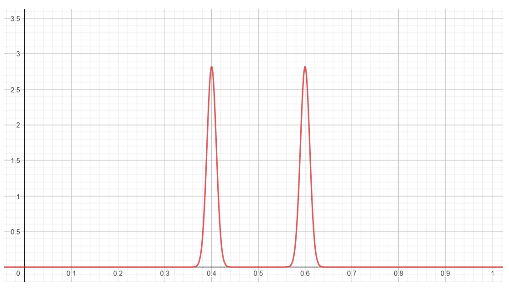

(b) $\mu = \{0.4, 0.6\}$

Figure 4.1: q with 2 Gaussian peaks of different μ.

In this case we let

$$q = \frac{1}{2\sqrt{\pi\sigma}} e^{\frac{1}{2}\frac{(x-\mu_1)^2}{\sigma^2}} + \frac{1}{2\sqrt{\pi\sigma}} e^{\frac{1}{2}\frac{(x-\mu_2)^2}{\sigma^2}}$$

such that we have two peaks. By adjusting μ_1 and μ_2 we can move the peaks closer together or further apart as shown in Figure 4.1. This allows us to test the ability of the methods to distinguish the peaks as they are brought closer together, as well as when they are subject to noise. Once again, given a source function q we compute the corresponding ideal Φ measurements and subsequently add random noise as before. The goal is to reconstruct the original q from the noisy measurements. The same reconstruction methods were implemented as before, fixing m at 200 points. The two peaks and reconstruction was implemented using the code in Appendix B.4. Note that the implementation is very similar to the code in B.2 used before, with the only significant difference being the implementation of `twopeaks(x)` and `gauss(x,sigma,mu)`.

Data was collected for (μ_i) pairs $(0.25, 0.75), (0.4, 0.6), (0.45, 0.55)$ with fixed $\sigma = 0.01$ for both peaks. Noise levels $\delta = 0.1, 0.01, 0.001, 0.0001$ and 0 were tested over 5 repetitions using the code shown in Appendix B.5. Once again we obtain the average scaled errors and regularization parameters as before.

4.1.3 2D reconstruction

Finally, the 3 methods were used to reconstruct a 2 dimensional image. $\mathbf{q}$ was taken to be the `shepp_logan_phantom()` shown in Figure 4.2 with $m = 158$. The corresponding measurements $\mathbf{\Phi}$ were computed as before and Gaussian noise was added once again. Reconstruction was repeated 5 times for noise levels $\delta = 0.1, 0.01, 0.001, 0.0001$ and 0. For the code of one trial see Appendix B.6. For the full experiment see Appendix B.7.

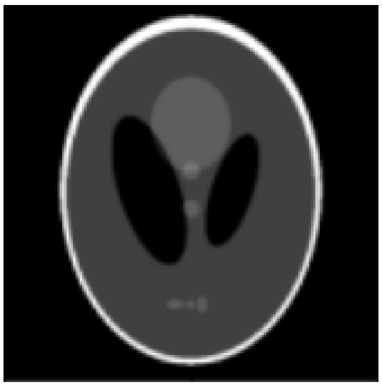


Figure 4.2: `shepp_logan_phantom()` used as true value of q.

4.2 The Results

4.2.1 A Smooth Source Function

Figure 4.3 shows the reconstruction for $m = 200$ and $\delta = 0.0001$ (seed=0). Clearly, although $m = 200$ is the finest discretization and 0.0001 is the smallest level of noise added, the regular inversion method is insufficient for the reconstruction of $q = x^2$. The poor reconstruction despite the little amount of noise is

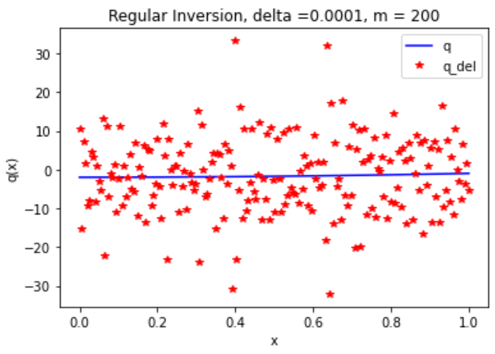


Figure 4.3: Reconstruction using regular inversion with $\delta = 0.001$, $m = 200$ and seed=0.

easily justified using the discrete Picard condition. The result of graphing the discrete Picard condition is given in figure 4.4.

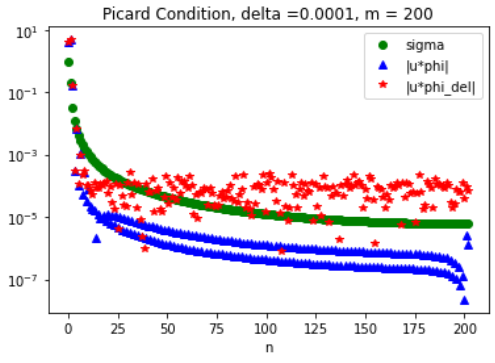
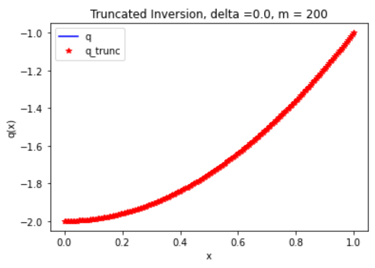


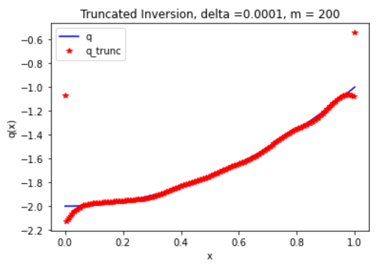
Figure 4.4: Picard condition computed for $m = 200$, $\delta = 0.0001$, seed=0.

As shown in figure 4.4, the Picard condition is not satisfied in the case of $\mathbf{\Phi}^\delta$ as the singular values of A^{-1} (in green) decay much quicker than the Fourier coefficients (in red). Nonetheless, in the case of noiseless data we have that the Picard condition does hold, which explains why using A works for that case. Since noise component corresponding to very small singular values get blown up, we expect to get an improved result in the reconstruction if we omit these values.

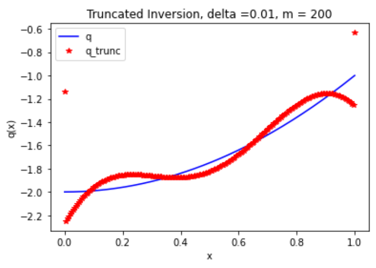
Figure 4.5 shows the result of reconstructing once again from noisy measurements $\mathbf{\Phi}^\delta$ for $m = 200$ (seed=0), this time using the truncated pseudo-inverse. We note from 4.5(b) that there is a notable improvement in the reconstruction of $\mathbf{q}$ from the noisy data Φ^δ compared to using just A as before. Note however that at the boundary (x=0 and x=1) the reconstruction significantly deviates from the true value compared to the rest of the reconstruction in both 4.5(b) and 4.5(c). Moreover, the internal points seem to suggest some form of undulation, which is most clear when there is greater noise. As in Figure 4.5(c), at $\delta = 0.01$ we clearly note the forced sinusoidal structure of the reconstruction using the truncated singular value decomposition (TSVD). In the case of no noise, since matrix A^{-1} is invertible, we should have no truncation and thus reconstruction should be the same as the regular inversion (i.e. exact solution) which is indeed the case as seen in Figure 4.5(a).



(a) $\delta = 0$



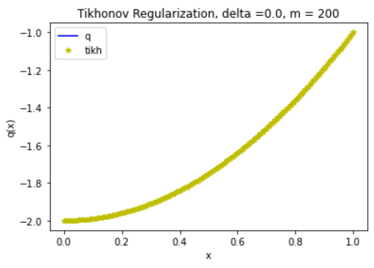
(b) $\delta = 0.0001$



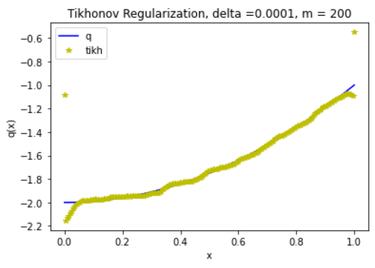
(c) $\delta = 0.01$

Figure 4.5: Truncated Singular Value Decomposition solution for $m = 200$, seed=0

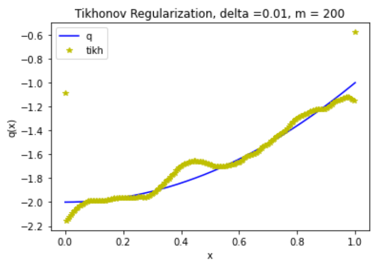
The same holds true for Tikhonov regularization; as shown in Figure 4.6(a), there is no regularization for $\delta = 0$. Figure 4.6(b) shows Tikhonov regularization for $\delta = 0.0001$ (seed=0) for comparison. Where the TSVD solution requires the sinusoidal shape, Tikhonov regularization does not enforce such regularity whilst still maintaining smoothness.



(a) $\delta = 0$



(b) $\delta = 0.0001$



(c) $\delta = 0.01$

Figure 4.6: Tikhonov regularization reconstruction for $m = 200$, seed=0

The average errors and parameters over 5 trials of each method are displayed in Table 4.1. The first column of values ("qd err") shows the error of reconstruction using the regular inverse, i.e. $\mathbf{q}^\delta = \mathbf{A}\mathbf{\Phi}^\delta$. As expected, this is 0 for no noise and largest for $\delta = 0.1$ in all cases of m. The absolute largest error occurs for $m = 200$ for $\delta = 0.1$ with a value of 2.13×10^4. Excluding $\delta = 0$, the smallest error occurs for $\delta = 0.0001$ for all m as expected. The absolute smallest occurs for $m = 10$ with a value of 6.18×10^{-2}. In general, decreasing the noise level by an order of magnitude reduces the error by an order of magnitude. This can be explained by the fact that at more discretization points the errors add up more whereas at $m = 10$ there are only few random errors in the first place and less chance of significant variations.

Nevertheless, most interesting is the comparison with the other methods. Both with the truncated solution as with Tikhonov regularization the largest average scaled error is of order -1 as shown in columns "qtrunc err" and "qtikh err". The scaled errors of both reconstruction methods is significantly smaller than of the regular inverse reconstruction. For both we have that the largest error is of order $\mathcal{O}(-1)$. The average scaled errors of both these methods have been graphed in Figure 4.7.

As shown in the Figure 4.7 we note that both the truncated inverse solution and Tikhonov regularization decreases in scaled error as δ decreases. Interesting to note is that for $\delta = 0.1$ Tikhonov performs worse at 20 discretization points compared to 10. Nevertheless, we note that this difference is merely 0.061. Moreover, we note that for $\delta = 0.1$ Tikhonov regularization results in a significantly larger error than the Truncated Singular Value Decomposition (TSVD) solution. In fact, overall, the TSVD solution seems to have a lower and otherwise similar, scaled error compared to Tikhonov regularization.

Although Figure 4.7 seems to present a clear picture of which reconstruction method is better, it is important to check the reconstruction parameters as well. In Table 4.1 these are given in columns "qtrunc param" and "qtikh param" for TSVD and Tikhonov regularization respectively. In the case of TSVD it is more insightful to look at the number of truncated singular values; the average number of truncated singular values for each δ given m internal discretization points is given in column "qtrunc num". Note that for

m	δ	qd err	qtikh err	qtikh param	qtrunc err	qtrunc num	qtrunc param
10	0.1000	69.863400	7.963300e-01	1.000000e-02	6.811760e-01	10.0	26.0
	0.0100	5.624440	5.232600e-01	1.000000e-03	5.291660e-01	8.8	90.8
	0.0010	0.662644	4.178320e-01	4.420000e-05	4.382700e-01	4.6	356.0
	0.0001	0.061848	6.176120e-02	2.200000e-08	6.184840e-02	0.0	490.0
	0.0000	0.000000	7.912800e-12	1.000000e-16	6.205300e-14	0.0	490.0
20	0.1000	251.002000	8.575580e-01	4.600000e-03	6.738740e-01	19.6	47.0
	0.0100	26.015400	5.128820e-01	1.000000e-03	4.400480e-01	18.8	84.8
	0.0010	2.131680	4.167800e-01	6.400000e-05	4.196520e-01	16.8	276.4
	0.0001	0.241670	2.147380e-01	8.200000e-08	2.416700e-01	0.0	1800.0
	0.0000	0.000000	9.035500e-11	1.000000e-16	5.645300e-13	0.0	1800.0
100	0.1000	5568.320000	6.849900e-01	2.800000e-03	5.240000e-01	99.2	64.0
	0.0100	536.812000	2.801260e-01	1.000000e-04	2.735540e-01	99.0	72.0
	0.0010	59.318200	2.501200e-01	1.000000e-05	2.337900e-01	96.6	278.0
	0.0001	5.588920	2.288460e-01	2.800000e-06	2.286280e-01	93.8	650.0
	0.0000	0.000000	2.544200e-08	1.000000e-16	2.298500e-11	0.0	41000.0
200	0.1000	21334.200000	6.115860e-01	1.000000e-03	3.424800e-01	199.0	79.0
	0.0100	2292.600000	2.400060e-01	1.000000e-04	2.436420e-01	199.0	79.0
	0.0010	221790000	1.875340e-01	1.000000e-05	1.771660e-01	195.8	374.0
	0.0001	21.540000	1.682960e-01	1.000000e-06	1.674360-01	191.4	1112.0
	0.0000	0.000000	2.881700e-07	1.000000e-16	8.802800e-11	0.0	160000.0

Table 4.1: Average scaled errors and reconstruction parameters for different values of δ and m over 5 trials.

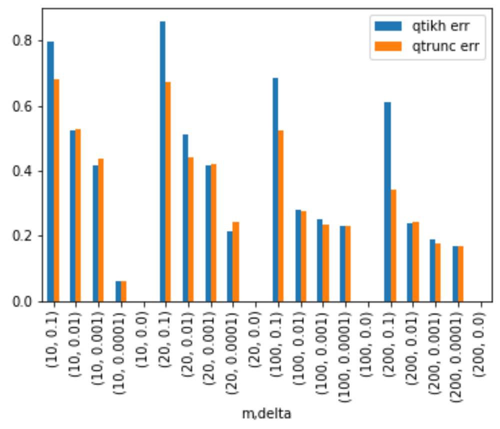


Figure 4.7: Bar chart showing average scaled errors and reconstruction parameters for the reconstruction of $q = x^2$ for different values of m and δ.

$m = 200$ and $\delta = 0.001$, TSVD supposedly performs better than Tikhonov regularization; however, we see that on average 196 out of 200 singular values were truncated which means that a lot of the original data is lost. Perhaps by using the Picard condition to choose the truncation parameter results in a more adequate reconstruction. As shown in figure 4.4, for the case of $m = 200$ and $\delta = 0.0001$ with seed=0, it seems as though truncating 175 singular values might suffice since then the majority of remaining singular values will then be larger than the Fourier coefficients of the noisy data. In this case we obtain the reconstruction shown in figure 4.8. In this case we note a significant improvement in the reconstruction of q compared to regular inversion, but certainly not less erroneous than the reconstruction with the smallest scaled error given in 4.5(b). Nonetheless, in the case that the true q were unknown and thus this 'best reconstruction' unavailable, using the Picard condition could allow for an initial choice of the truncation parameter. Nevertheless, we note that 175 truncated singular values (out of a total of 200) is still a lot.

Since $q = x^2$ is a smooth and regular function the fact that many singular values are truncated doesn't

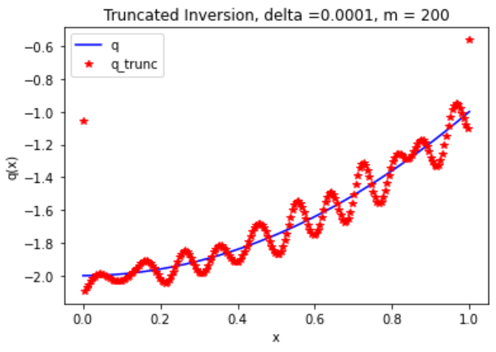


Figure 4.8: 175 truncated singular values reconstruction for $m = 200$ with $\delta = 0.0001$

cause too many problems in reconstruction. However, for different kinds of source functions and 2D imaging it may be more problematic. Furthermore, note that for smaller noise levels Tikhonov regularization has comparable error to TSVD as well as a small regularization parameter. The latter means that Tikhonov remains closer to the original data for small amounts of noise as desired.

We note that overall, the truncated inversion method forces the reconstruction to have a sinusoidal shape, which for a smooth curve with many discretization points may work well. Moreover, theoretically this allows the reconstruction error to remain small. However, the number of truncated parameters is large whereas Tikhonov may remain closer to the data fidelity term. Hence, for actual image reconstruction we cannot conclude that TSVD would perform better.

4.2.2 Further exploration: two peaks

μ	δ	qd err	qtikh err	qtikh param	qtrunc err	qtrunc num	qtrunc param
$[0.25, 0.75]$	0.1000	$9.703e+04$	$5.027e+00$	$4.60e+00$	$5.067e+00$	199.8	39.0
	0.0100	$3.521e+04$	$1.699e+01$	$8.20e-05$	$1.679e+01$	197.0	230.0
	0.0010	$3.590e+03$	$1.507e+01$	$1.00e-06$	$1.516e+01$	190.6	1276.0
	0.0001	$3.492e+02$	$9.876e+00$	$1.00e-08$	$9.070e+00$	171.4	8840.0
	0.0000	$3.244e-11$	$4.094e-07$	$1.00e-16$	$1.460e-10$	2.0	160000.0
$[0.4, 0.6]$	0.1000	$9.753e+04$	$4.951e+00$	$2.80e-03$	$4.959e+00$	199.0	79.0
	0.0100	$3.892e+04$	$1.814e+01$	$8.20e-05$	$1.839e+01$	199.0	79.0
	0.0010	$4.031e+03$	$1.633e+01$	$1.00e-06$	$1.681e+01$	189.0	1600.0
	0.0001	$3.921e+02$	$1.081e+01$	$1.00e-08$	$1.010e+01$	171.0	9080.0
	0.0000	$2.081e-11$	$4.597e-07$	$1.00e-16$	$1.392e-13$	3.0	160000.0
$[0.45, 0.55]$	0.1000	$9.754e+04$	$4.850e+00$	$1.00e-03$	$4.874e+00$	199.0	79.0
	0.0100	$3.929e+04$	$1.750e+01$	$1.00e-04$	$1.776e+01$	197.4	217.6
	0.0010	$4.079e+03$	$1.694e+01$	$2.80e-06$	$1.763e+01$	193.4	726.0
	0.0001	$3.967e+02$	$9.994e+00$	$1.00e-08$	$1.016e+01$	175.0	7100.0
	0.0000	$1.597e-11$	$4.651e-07$	$1.00e-16$	$1.387e-10$	2.0	160000.0

Table 4.2: Table of average scaled errors and reconstruction parameters for each μ_i pair with different δ.

The average scaled errors and parameters for the reconstruction of two Gaussian peaks are displayed in Figure 4.2 where $\mu_i = [0.25, 0.75]$ are the peaks that are furthest apart and $\mu_i = [0.45, 0.55]$ are closest together. First we note once again that a regular inverse does not suffice for the reconstruction. The largest average scaled error occurs for the peaks closest together and with most noise as expected. The value of this error is 9.75×10^4. We note that for no noise the average scaled error no longer obtains a value 0. For μ_i pair $(0.25, 0.75)$ this could be explained because the boundary conditions taken into account by matrix A do not hold. However, in the case of $(0.4, 0.6)$ and $(0.45, 0.55)$ the boundary conditions hold (since $\Phi'(0) = -2\Phi(0) = 0 = 2\Phi(1) = \Phi'(1)$), thus we conclude that the difficulty in reconstruction there has to do with the ill-posedness rather than the holding of the boundary conditions.

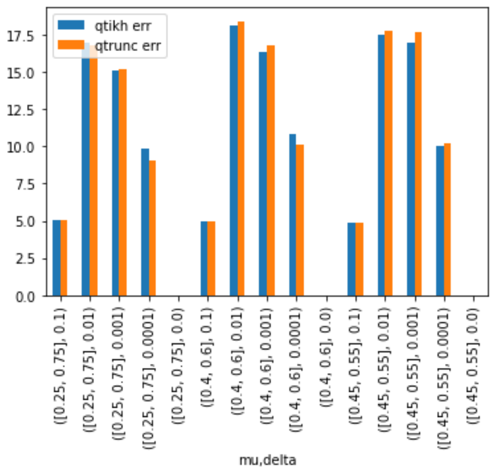
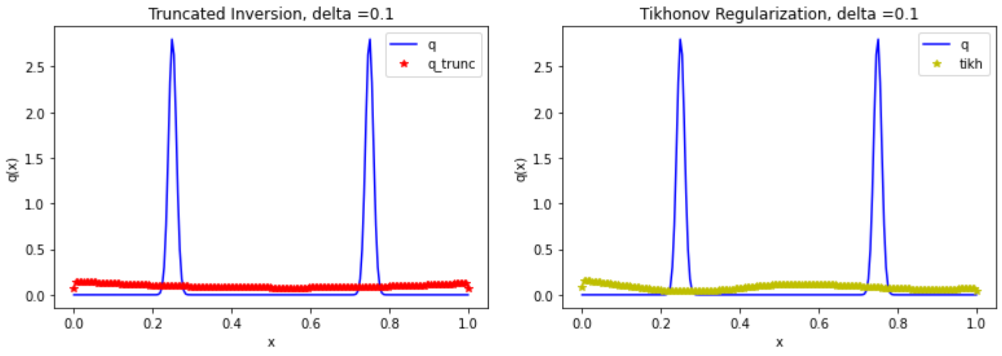


Figure 4.9: Bar chart showing average scaled errors and reconstruction parameters for the reconstruction of two gaussian peaks with different values of δ and μ_i pairs.

Moreover, the average scaled error increases as the peaks are brought closer together. Intuitively this makes sense because it becomes more difficult to distinguish the two peaks. The average scaled errors for the TSVD solution and Tikhonov regularization method are displayed in figure 4.9. We note that overall the errors behave similar to before in that for noise levels 0.01 to 0 the average scaled error decreases. Moreover, it is important to note that although the average scaled error for noise level $\delta = 0.1$ is the smallest, this is definitely not the best reconstruction. In fact, in this case both reconstruction methods no longer identify either of the peaks altogether as shown for $\mu = (0.25, 0.75)$ in figure 4.10.

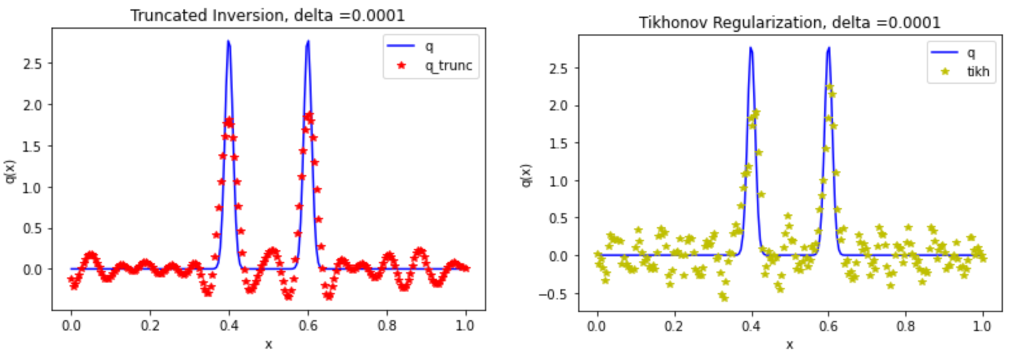


(a) TSVD

(b) Tikhonov regularization

Figure 4.10: Reconstruction with $m = 200$ for $\delta = 0.1$ (seed=0) using TSVD and Tikhonov regularization.

We consider the reconstruction of the peaks with $\mu_i = (0.4, 0.6)$ and noise level 0.0001. The results for seed=0 are shown in figure 4.11. In the case of both methods, the size of the peaks are underestimated. Moreover, both methods contain noise on the sides of the peaks. If there is a priori knowledge available on the shape of the source function this could be easily filtered using some sort of threshold. However, in general in applications the true value of the source function is unknown. In that case, it seems as though the truncated inversion solution is more easily misinterpreted to have more peaks, due to the sinusoidal shape and general regularity of the noisy reconstruction compared to Tikhonov regularization. Nevertheless, it must be noted that Tikhonov regularization can easily lead to oversmoothing.



(a) TSVD solution (b) Tikhonov regularization

Figure 4.11: Reconstruction of 2 peaks with $mu = (0.4, 0.6)$ and noise $\delta = 0.0001$, seed=0.

δ	qd err	qtikh err	qtikh param	qtrunc err	qtrunc num	qtrunc param
0.1000	6.682320e+01	2.239460e+00	1.000000e+00	4.731780e+00	2.0	26.0
0.0100	1.842580e+01	4.476820e+00	8.200000e-06	4.738980e+00	2.0	350.0
0.0010	1.902020e+00	1.209480e+00	1.000000e-09	1.358980e+00	2.0	47800.0
0.0001	1.902180e-01	1.852720e-01	1.000000e-11	1.859640e-01	2.0	100000.0
0.0000	5.149100e-11	3.969800e-07	1.000000e-16	7.669600e-11	2.0	100000.0

Table 4.3: Table of average scaled errors and reconstruction parameters for the reconstruction of `shepp_logan_phantom()` 2D image for $\delta = 0.1, 0.01, 0.001, 0.0001$ and 0.

4.2.3 2D reconstruction

The average scaled errors in the reconstruction of `shepp_logan_phantom()` over 5 trials is displayed in Table 4.3. We note once again that the errors in regular reconstruction increase as δ increases. Interesting

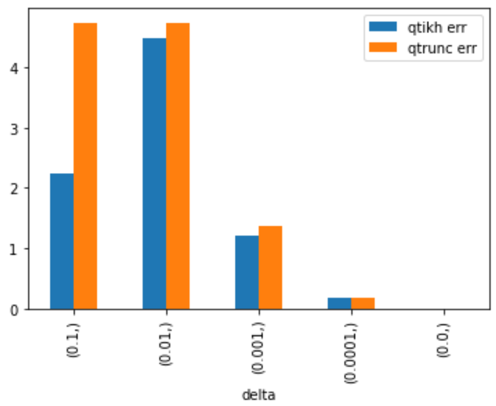
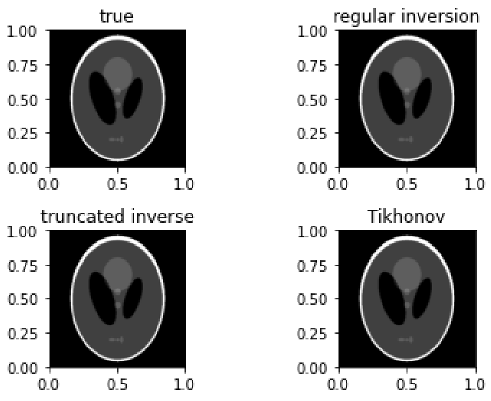
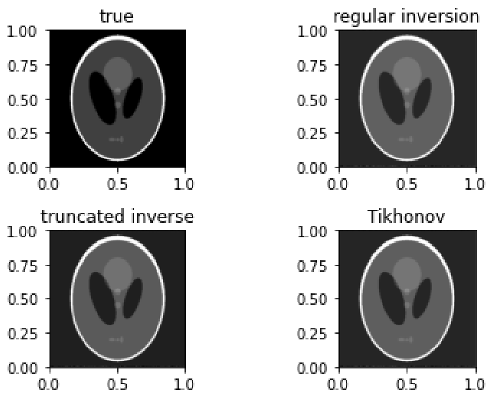


Figure 4.12: Average scaled errors over 5 trials of TSVD solution and Tikhonov regularization for the `shepp_logan_phanton` with noise $\delta = 0.1, 0.01, 0.001, 0.0001$ and 0

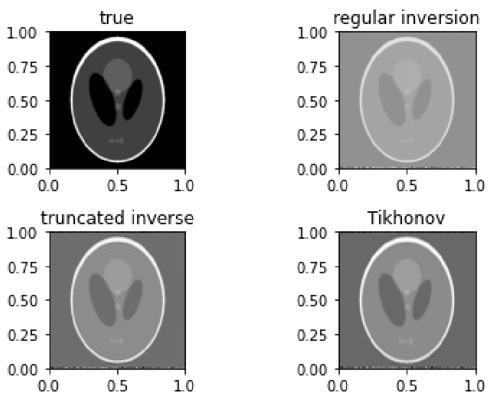
to note is that the largest average scaled error, which occurs for $\delta = 0.1$, is much smaller than in the previous 1 dimensional reconstructions at 66.8. The average scaled errors for the Truncated Singular Value Decomposition solution (TSVD) and Tikhonov regularization are graphed in figure 4.12. These are smaller than for the reconstruction of 2 peaks but larger than the 1 dimensional reconstruction of $q = x^2$. As expected, we note that Tikhonov regularization has a lower scaled error than TSVD. Interesting to note is that at the lowest amount of noise $\delta = 0.0001$ the average scaled error is comparable (4.74 and 4.48). For seed=0 the reconstruction looks as shown in figure 4.13(b).



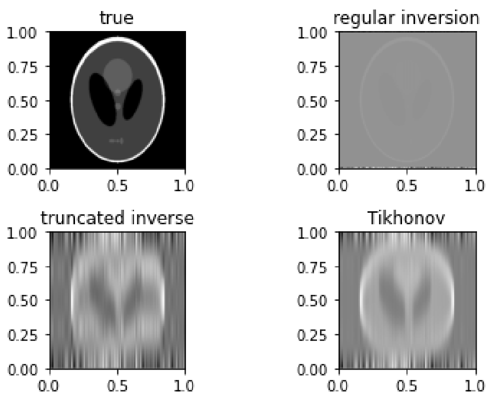
(a) $\delta = 0$



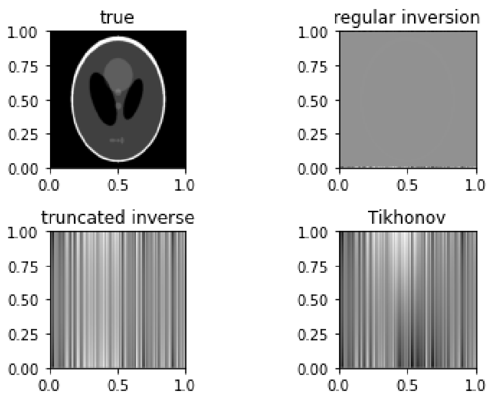
(b) $\delta = 0.0001$



(c) $\delta = 0.001$



(d) $\delta = 0.01$



(e) $\delta = 0.1$

Figure 4.13: Reconstruction using regular inversion (top right), TSVD (bottom left) and Tikhonov (bottom right) of `shepp_logan_phantom()` for different δ values seed=0.

We note that indeed all three reconstructions are similar. The biggest statistical difference is found at $\delta = 0.1$, figure 4.13(e), TSVD has a slightly larger scaled error than Tikhonov regularization; however, in terms of image reconstruction, all three methods are insufficient. For $\delta = 0.01$ we see that even the regular inverse actually performs alright in that the outer border and two main dark shapes are distinguished. Nonetheless, the other shapes are lost; resolution is lower due to the loss of contrast. This is recovered best with Tikhonov regularization, where the top circle is also visible. Although the image appears blurry, using Tikhonov regularization now three main shapes are still notable. Yet the smaller details are still lost. Compared to the regular inverse the shape is blurrier however the contrast of the border is clearer in Tikhonov. The image reconstructed using TSVD is so blurry that even the overall shape is distorted. $\delta = 0.001$ is an interesting case. Reconstruction with the regular inverse is good, nonetheless both Tikhonov regularization and TSVD solution has a greater contrast and thus can distinguish the smaller details better. We note that Tikhonov is slightly better than the truncated inverse. For noise level $\delta = 0.0001$ and no noise the reconstructions are optically similar (in the case of no noise identical).

Chapter 5

Concluding Remarks

In this chapter a conclusion is drawn from the numerical experiments and possible extensions to this project are suggested.

5.1 Conclusion

Bioluminescence Tomography is governed by the Radiative Transfer Equation, a six-variable integro-differential equation. By applying the diffusion approximation, we obtain a simpler 2nd order differential equation in section 2. This equation is, in the case of the steady-state assumption, dependent only on space. In terms of computational complexity, this is a much simpler problem to tackle. However, obtaining the source function from discrete (and incomplete) data is an ill-posed inverse problem. Han and Wang in [3] thus proceed to apply the idea of Tikhonov regularization to the variational formulation of the problem, which by Lax-Milgram does have a unique solution. Following this idea, we discretize the problem and implement a reconstruction of various source functions using a) a regular inversion, b) a truncated singular value decomposition and c) Tikhonov regularization in section 3 for comparison.

The results presented in section 4 of these numerical experiments emphasize the ill-posedness of the problem. We note that in one dimension the reconstruction from noisy data points without any form of regularization was inadequate, even in the case of a smooth one-dimensional source function $q = x^2$. Both alternative methods, namely Truncated Singular Value Decomposition (TSVD) and Tikhonov regularization, performed significantly better. Nevertheless, it must be pointed out that in our case we had complete knowledge of the source function we were attempting to reconstruct, which in applications is often not the case. This causes reconstruction to be even more difficult. For instance, it was noted that choosing the truncation parameter using the Picard condition resulted in a slightly less accurate reconstruction than when using the smallest scaled norm. Having said that, using the Picard condition for choosing a truncation parameter is a valid option if no a priori information is known of what the source function should look like, despite that it is not the optimal solution.

TSVD and Tikhonov regularization performed similarly well in reconstruction of $q = x^2$ and also $q = \frac{1}{2\sqrt{\pi\sigma}} e^{\frac{1}{2}\frac{(x-\mu_1)^2}{\sigma^2}} + \frac{1}{2\sqrt{\pi\sigma}} e^{\frac{1}{2}\frac{(x-\mu_2)^2}{\sigma^2}}$. Nevertheless, in the latter case, TSVD had the disadvantage that the reconstruction could be more readily misinterpreted due to the forced sinusoidal-like shape and thus extra 'peaks'. The advantage of Tikhonov regularization became most apparent in the two-dimensional reconstruction as for larger noise TSVD became significantly more blurry than Tikhonov regularization. Nevertheless, we still see that for larger noise it still performs slightly better than the regular inversion. Surprisingly, the regular inversion worked well for noise levels under $\delta = 0.001$.

Despite its advantages, Tikhonov regularization can result in the oversmoothing of an image. This means that edges could be blurred out. Hence, it would be an interesting extension to this project to test a method such as Total Variance regularization which allows for edges. This is also interesting for applications of BLT since it would allow for clearer identification of for instance organs in medical imaging. Moreover, in this project it was assumed that certain parameters were constants, which in general is not the case. The incorporation of their true value would be a valuable addition to this investigation. Lastly, note that the diffusion approximation is already an approximation, [1] suggests the development of a new approach which combines the Monte Carlo method (applied directly to the Radiative Transfer Equation) and the diffusion equation. This indeed seems like an interesting option which would allow for more accuracy at a slightly higher computational cost, but not too involved as the Monte Carlo Method is on its own.

In conclusion, these results give a glimpse on the ill-posedness of the inverse problem related to Bioluminescence Tomography of reconstructing the source function from (noisy) data, but leave plenty of room for further analysis. It will be interesting to see the developments within this field in a couple of years.

Bibliography

[1] Lihong V Wang and Hsin-i Wu. *Biomedical optics: principles and imaging.* John Wiley & Sons, 2012.

[2] Jingjing Yu et al. “Bioluminescence Tomography Based on One-Dimensional Convolutional Neural Networks”. In: *Frontiers in Oncology* 11 (2021). ISSN: 2234-943X. DOI: `10.3389/fonc.2021.760689`. URL: `https://www.frontiersin.org/article/10.3389/fonc.2021.760689`.

[3] Weimin Han and Ge Wang. “Bioluminescence tomography: biomedical background, mathematical theory, and numerical approximation”. In: *Journal of computational mathematics: an international journal on numerical methods, analysis and applications/edited by Editorial Committee of Journal of computational mathematics* 26.3 (2008), p. 324.

[4] Claire L Ryder. “Summary of Phase Function from scatter90 and for RFMDISORT”. In: *Reading, March* 18 (2011).

[5] Jie Tian et al. “Bioluminescence tomography”. In: *Molecular Imaging.* Springer, 2013, pp. 217–240.

[6] Richard C Haskell et al. “Boundary conditions for the diffusion equation in radiative transfer”. In: *JOSA A* 11.10 (1994), pp. 2727–2741.

Appendix A

Extras

A.1 Ensuring the Correct Implementation of Matrix A

Before conducting the numerical experiments it was crucial that matrix A given in [ref] be implemented correctly in python. To do this we choose a function to describe our measurements and compute the corresponding source function by hand. We compare this with obtaining q from matrix A.

A.1.1 Chooising a simple source function

We take an arbitrary function to describe our hypothetical measurements $\Phi = ax^3 + bx^2 + cx + d$ on $[0, 1]$ and require that Φ satisfies the boundary conditions (2.15)-(2.16), i.e.

$$\begin{aligned}
\Phi'(x) &= 3ax^2 + 2bx + c \\
3a(0)^2 + 2b(0) + c &= -2(a(0)^3 + b(0)^2 + c(0) + d) \\
c &= -2d \\
&\text{and} \\
3a(1)^2 + 2b(1) + c &= 2(a(1)^3 + b(1)^2 + c(1) + d) \\
3a + 2b + c &= 2a + 2b + 2c + 2d \\
a - c &= d
\end{aligned}$$

Hence, the simplest scenario to test would $a = c = d = 0$, namely $\Phi = x^2$.

A.1.2 Implementing matrix A

Given Φ we hence expect q to be

$$q(x) = -\Phi'' + \Phi = -2 + x^2$$

In other words, we want $A\mathbf{\Phi} = -2 + x^2$ for large n where $\mathbf{\Phi} = (\Phi_i)$ with $\Phi_i = \Phi(x_i)$. Φ was implemented with function `func(x)` and discretized using `discrete(function)` given in Appendix B.1 (lines 4-5 and 52-61 respectively). The corresponding **q** was computed (Appendix B.2 line 23) as $\mathbf{q} = A\mathbf{\Phi}$ where matrix A was implemented using the function `getA(m)` presented in Appendix B.1 (lines 18 - 61). As shown in Figure [ref], we note that indeed the two graphs (blue=true value and orange=discretized reconstruction) coincide. We also note that the reconstruction improves when a larger number of grid points are used. Note that for $m = 10$ in (a) there is a slight discrepancy around $x = 0$ whereas this is negligible in the graph for $m = 200$ (c). We conclude that the implementation of A is correct and move on to the reconstruction of noisy data.

Appendix B

Python Code

The following chapter includes all of the code I wrote for the numerical exploration of this thesis. B.1 contains all the predefined functions used to compute the solutions from the (noisy) data. B.2 contains 1 trial of the numerical experiment conducted in 4.

B.1 untitled0.py

```python
import numpy as np

#function
def func(x):
    return  x**2

#derivative of function
def funcprime(x):
    return 2*x

#construct matrix A and corresponding x-axis
def getK(m):
   K = np.linalg.inv(getA(m))
   x = np.linspace(0,1,m+2)
   return K,x

#construct matrix A of Finite Discretization Method
def getA(m):
    h = float(1/(m+1))
    H = -(1/(h**2))
    A = np.zeros((m+2,m+2))

    #left boundary
    A[0,0] = H*(4*h-2)
    A[0,1] = H*(2-h**2)

    #u_i-1
    i=1
    while i<=m:
        A[i,i-1]=H
        i=i+1

    #u_i
    i=1
    value=-H*(2+h**2)
    while i<=m:
        A[i,i]=value
        i=i+1

    #u_i+1
    i=1
    while i<=m:
        A[i,i+1]=H
        i=i+1

    #right boundary
    A[m+1,m]=H*2
    A[m+1,m+1]=H*(4*h-(2+h**2))
    return A

#discretization of function
def discrete(function,m):
    h = float(1/(m+1))
    j=0
    phi=[]
    while j<=m+1:
        x=j*h
        a=function(x)
        phi.append(a)
        j=j+1
    return phi

#check boundary conditions
def checkBC(func,funcprime):
    nulcheck = funcprime(0) + 2*func(0)
    onecheck = funcprime(1) - 2*func(1)
    if nulcheck ==0 and onecheck ==0:
        ans = print("boundary conditions hold")
        return ans
    else:
        print("0 boundary condition: ", nulcheck)
        print("1 boundary condition: ", onecheck)
        return False

#additive noise
def noisy_add(phi,delta):
    noise = delta*np.random.randn(len(phi))
    phi_del = phi + noise
    return phi_del

#truncated singular value decomposition solution with smallest scaled error
def best_trunc(A,phi_del,q):
    U, s, v = np.linalg.svd(A) #singular value decomposition
    trunc = s[0]
    Norm = []
    alt =0
    this = 400000
    while trunc>0:
        copy = s #initialize a copy of the singular values to truncated
        i=0
        while i < len(s):
            if copy[i] > trunc:
                copy[i]=0
            i=i+1
        A1 = np.matmul(np.matmul(U,np.diag(copy)),v) #truncated A
        q1 = np.matmul(A1,phi_del)
        norm = np.linalg.norm(q-q1)/np.linalg.norm(phi_del)
        Norm.append(norm)
        if norm<this: #remember smallest scaled error and corresponding truncation parameter
            this=norm
            alt=q1
            hi=trunc
            j=0
            for r in copy:
                if r==0:
                    j+=1
        trunc=trunc-10
    print("upper truncation:", hi)
    print("q vs q trunc error:", this)
    return this, alt, hi, j
    """"returns scaled error, TSVD solution, truncation parameter
    and number of truncated singular values"""

#to check Picard condition
def Picard_check(U,phi,phi_del):
    Picard=[]
    Picard_del=[]
    i=0
    while i<len(phi):
        Picard.append(abs(np.dot(U[:,i],phi)))
        Picard_del.append(abs(np.dot(U[:,i],phi_del)))
        i=i+1
    return Picard, Picard_del

#Tikhonov regularization solution
def Tikh_reg(U,s,v,phi,a):
    ans=v.T@np.linalg.inv((np.diag(s**2)+(a*np.identity(len(s)))))@np.diag(s)@U.T@phi
    return ans

#Tikhonov regularization with smallest scaled error
def best_Tikh(U,s,v,phi,q):
    ANS=0
    this=400
    a=1e-16
    while a<=100: #iterate over regularization parameter
        q1=Tikh_reg(U,s,v,phi,a)
        norm =np.linalg.norm(q-q1)/np.linalg.norm(phi)
        if norm<this:
            this=norm
            ANS=q1
            alph=a
        a=a*10
    return ANS,alph,this #returns regularized solution, parameter and scaled error

def bias_var(U, s, Vh,f,f_delta,u):
    # error, bias and variance for TSVD
    n=len(s)
    error = np.zeros(n)
    bias = np.zeros(n)
    variance = np.zeros(n)
    for k in range(0,n):
        uk = Vh[:k,:].T@np.diag(1/s[:k])@U[:,:k].T@f
        uk_delta = Vh[:k,:].T@np.diag(1/s[:k])@U[:,:k].T@f_delta
        error[k] = np.linalg.norm(u - uk_delta)
        bias[k] = np.linalg.norm(u - uk)
        variance[k] = np.linalg.norm(uk - uk_delta)

    return error, bias, variance
```

B.2 CORRECTTHESIS.py

The option of user input for values of m and δ (instead of argument line input) is possible and given in lines 11 and 39 respectively.

```
1  import importlib as imp
2  import numpy as np
3  import matplotlib.pyplot as mp
4  import untitled0 as u
5  imp.reload(u)
6  from sys import argv
7  #PART 1: Constructing and testing matrix A in A*phi=q
8
9  m=int(argv[1])
10 #how many internal grid points
11 #m = int(input("How many internal grid points should be used? (Insert positive integer): "))
12
13 # construct matrix A
14 A = u.getA(m)
15
16 #discretize phi
17 phi = u.discrete(u.func,m)
18
19 #check boundary conditions
20 u.checkBC(u.func,u.funcprime)
21
22 #compute q = A*phi
23 q = np.matmul(A,phi)
24
25 #graph q vs -ddphi + phi
26 xaxis2 = np.linspace(0.0, 1.0, 1000)
27 xaxis = np.linspace(0,1,m+2)
28
29 mp.figure(0)
30 mp.title("Test of A")
31 mp.xlabel("x")
32 mp.ylabel("q(x)")
33 mp.plot(xaxis,q,label='q=A*phi')
34 mp.plot(xaxis2,u.func(xaxis2)-2,label='q=-ddphi+phi')
35 mp.legend()
36
37
38 #PART 2: adding noise
39 delta = float(argv[2])
40 #delta = float(input("Noise level to be added: "))
41 phi_del = u.noisy_add(phi,delta)
```

```

#compute q from noisy measurements
q_del = np.matmul(A,phi_del)

#plotting
mp.figure(1)
mp.title("Regular Inversion, delta ="+str(delta)+", m = "+str(m))
mp.xlabel("x")
mp.ylabel("q(x)")
mp.plot(xaxis,q,'b',label='q')
mp.plot(xaxis,q_del,'r*',label='q_del')
mp.legend()

err = np.linalg.norm(q-q_del)/np.linalg.norm(phi_del)
print("q vs q del error: ", err)

#PART 3
#truncated solution
#choosing best truncation parameter
trunc_err, alt, trunc_param, numb = u.best_trunc(A,phi_del,q)
print("number of truncated s: ",numb)

#plot
mp.figure(2)
mp.title("Truncated Inversion, delta ="+str(delta)+", m = "+str(m))
mp.xlabel("x")
mp.ylabel("q(x)")
mp.plot(xaxis,q,'b',label='q')
mp.plot(xaxis,alt,'r*',label='q_trunc')
mp.legend()
mp.show()

#PART 4
#Calculating picard condition
U, s, v = np.linalg.svd(np.linalg.inv(A))
Picard, Picard_del = u.Picard_check(U,phi,phi_del)

mp.figure(3)
mp.title("Picard Condition, delta ="+str(delta)+", m = "+str(m))
mp.xlabel("n")
mp.semilogy(xaxis*(m+2),s,'go',label='sigma')
mp.semilogy(xaxis*(m+2),Picard,'b^',label='|u*phi|')
mp.semilogy(xaxis*(m+2),Picard_del,'r*',label='|u*phi_del|')
mp.legend()
mp.show()
```

```

#PART 5
#Tikhonov regularization
ans,a,NORM = u.best_Tikh(U,s,v,phi_del,q)

print("parameter alpha: ",a)
print("error in tikhonov: ", NORM)

#plot
mp.figure(4)
mp.title("Tikhonov Regularization, delta ="+str(delta)+", m = "+str(m))
mp.xlabel("x")
mp.ylabel("q(x)")
mp.plot(xaxis,q,'b',label='q')
mp.plot(xaxis, ans, 'y*', label='tikh')
mp.legend()
mp.show()

#PART 6
#Visualize bias variance tradeoff for TSVD
error, bias, variance = u.bias_var(U, s, v,phi,phi_del,q)

k = np.linspace(0,m+1,m+2)
mp.figure(5)
mp.title("Bias Variance Trade-Off, delta ="+str(delta)+", m = "+str(m))
mp.plot(k,bias,label='bias')
mp.plot(k,variance,label='variance')

mp.xlabel('number of singular values cut off')
mp.ylabel('error')
#note that limits are best adjusted manually depending on value of delta
#mp.ylim([0,10])
#mp.xlim([0,10])

mp.legend()
mp.show()

print("condition number of A: ",np.linalg.cond(A))
```

B.3 DATACOLLECTIONtrue.py

```
from numpy import random
import pandas as pd

```

```
#m and delta values tested
list_m = [10,20,100,200]
list_delta = [0.1,0.01,0.001,0.0001,0]

#one trial of numerical experiment
def data_collection(r=0,list_m=list_m,list_delta=list_delta):
    df = pd.DataFrame()
    first_row=[]
    qd_error_list=[]
    qtrunc_error_list=[]
    qtrunc_param_list=[]
    qtrunc_num_list=[]
    qtikh_error_list=[]
    qtikh_param_list=[]
    for i in list_m:
        random.seed(r) #set seed
        for d in list_delta:
            first_row.append(tuple([i,d]))
            this = str(i)+' '+str(d)
            runfile('C:/Users/jocel/Thesis/CORRECTTHESIS.py', args=this,\
            wdir='C:/Users/jocel/Thesis')
            qd_error_list. append(float('%.5g'%err))
            qtrunc_error_list.append(float('%.5g'%trunc_err))
            qtrunc_param_list.append(float('%.2g'%trunc_param))
            qtrunc_num_list.append(numb)
            qtikh_error_list.append(float('%.5g'%NORM))
            qtikh_param_list.append(float('%.2g'%a))
            d+=1
    df = pd.DataFrame(
            {"qd err" : qd_error_list,
             "qtrunc err" : qtrunc_error_list,
             "qtrunc param": qtrunc_param_list,
             "qtrunc num": qtrunc_num_list,
             "qtikh err": qtikh_error_list,
             "qtikh param": qtikh_param_list},
            index = pd.MultiIndex.from_tuples(first_row
                , names=["m","delta"]))
    return df


trials = 5

i=1
data=data_collection(0)
result=data
```

```
while i < trials:
    one_trial = data_collection(i)
    data = pd.merge(data,one_trial,on=["m","delta"])
    result = pd.concat([result,one_trial],axis=1)
    i+=1
pd.set_option('display.max_columns', None)
print(data)

#display means over 5 trials
print(result.groupby(level=0,axis=1).mean())
result.loc[:, ~result.columns.isin(['qd err','qtikh param', 'qtrunc param', 'qtrunc num'])]\
.groupby(level=0,axis=1).mean().plot.bar();
```

B.4 TwoPeaks.py

```
import importlib as imp
import untitled0 as u
import numpy as np
import matplotlib.pyplot as mp
imp.reload(u)
from sys import argv

#gaussian
def gauss(x,sigma,mu):
    frac = 1/(2*np.sqrt(np.pi*sigma))
    powe = -0.5*((x-mu)**2/sigma**2)
    expo = np.e**(powe)
    return frac*expo

#two peaks
def twopeaks(x):
    peakone = gauss(x,0.01,float(argv[1]))
    peaktwo = gauss(x,0.01,float(argv[2]))
    return peakone+peaktwo

m=200

q = u.discrete(twopeaks,m)
A = u.getA(m)
phi = np.linalg.inv(A)@q

xaxis2 = np.linspace(0.0, 1.0, 1000)
xaxis = np.linspace(0,1,m+2)

```

```
#PART 2: adding noise
delta = float(argv[3])
#delta = float(input("Noise level to be added: "))
phi_del = u.noisy_add(phi,delta)

#compute q from noisy measurements
q_del = np.matmul(A,phi_del)

mp.figure(1)
mp.title("Regular Inversion, delta ="+str(delta))
mp.xlabel("x")
mp.ylabel("q(x)")
mp.plot(xaxis,q,'b',label='q')
mp.plot(xaxis,q_del,'r*',label='q_del')
mp.legend()

err = np.linalg.norm(q-q_del)/np.linalg.norm(phi_del)
print("q vs q del error: ", err)

#PART 3
#truncated solution
#choosing best truncation parameter
trunc_err, alt, trunc_param, numb = u.best_trunc(A,phi_del,q)
print("number of truncated s: ",numb)

mp.figure(2)
mp.title("Truncated Inversion, delta ="+str(delta))
mp.xlabel("x")
mp.ylabel("q(x)")
mp.plot(xaxis,q,'b',label='q')
mp.plot(xaxis,alt,'r*',label='q_trunc')
mp.legend()
mp.show()

#PART 4
#Calculating picard condition
U, s, v = np.linalg.svd(np.linalg.inv(A))
Picard, Picard_del = u.Picard_check(U,phi,phi_del)

mp.figure(3)
mp.title("Picard Condition, delta ="+str(delta))
mp.xlabel("n")
mp.semilogy(xaxis*(m+2),s,'go',label='sigma')
mp.semilogy(xaxis*(m+2),Picard,'b^',label='|u*phi|')
mp.semilogy(xaxis*(m+2),Picard_del,'r*',label='|u*phi_del|')
mp.legend()
mp.show()

#PART 5
#Tikhonov regularization
ans,a,NORM = u.best_Tikh(U,s,v,phi_del,q)

print("parameter alpha: ",a)
print("error in tikhonov: ", NORM)
mp.figure(4)
mp.title("Tikhonov Regularization, delta ="+str(delta))
mp.xlabel("x")
mp.ylabel("q(x)")
mp.plot(xaxis,q,'b',label='q')
mp.plot(xaxis, ans, 'y*', label='tikh')
mp.legend()
mp.show()

#PART 6
#Visualize bias variance tradeoff for TSVD
error, bias, variance = u.bias_var(U, s, v,phi,phi_del,q)

k = np.linspace(0,m+1,m+2)
mp.figure(5)
mp.title("Bias Variance Trade-Off, delta ="+str(delta))
mp.plot(k,bias,label='bias')
mp.plot(k,variance,label='variance')

mp.xlabel('number of singular values cut off')
mp.ylabel('error')
#note that limits are best adjusted manually depending on value of delta
#mp.ylim([0,10])
#mp.xlim([0,10])

mp.legend()
mp.show()

print("condition number of A: ",np.linalg.cond(A))
```

B.5 TPdatacollection.py

```
from numpy import random
import pandas as pd

```

```
#mu paris of the gaussian peaks tested
list_mu = [[0.25,0.75],[0.4,0.6],[0.45,0.55]]
list_delta = [0.1,0.01,0.001,0.0001,0]

def data_collection(r=0,list_mu=list_mu,list_delta=list_delta):
    df = pd.DataFrame()
    first_row=[]
    qd_error_list=[]
    qtrunc_error_list=[]
    qtrunc_param_list=[]
    qtrunc_num_list=[]
    qtikh_error_list=[]
    qtikh_param_list=[]
    for i in list_mu:
        random.seed(r)
        for d in list_delta:
            first_row.append(tuple([str(i),d]))
            this = str(i[0])+' '+str(i[1])+' '+str(d)
            runfile('C:/Users/jocel/Thesis/TwoPeaks.py', args=this,\
            wdir='C:/Users/jocel/Thesis')
            qd_error_list.append(float('%.5g'%err))
            qtrunc_error_list.append(float('%.5g'%trunc_err))
            qtrunc_param_list.append(float('%.2g'%trunc_param))
            qtrunc_num_list.append(numb)
            qtikh_error_list.append(float('%.5g'%NORM))
            qtikh_param_list.append(float('%.2g'%a))
            d+=1
    df = pd.DataFrame(
            {"qd err" : qd_error_list,
             "qtrunc err" : qtrunc_error_list,
             "qtrunc param": qtrunc_param_list,
             "qtrunc num": qtrunc_num_list,
             "qtikh err": qtikh_error_list,
             "qtikh param": qtikh_param_list},
            index = pd.MultiIndex.from_tuples(first_row
                , names=["mu","delta"]))
    return df

trials = 5

i=1
data=data_collection(0)
result=data
while i < trials:
    one_trial = data_collection(i)
    data = pd.merge(data,one_trial,on=["mu","delta"])
    result = pd.concat([result,one_trial],axis=1)
    i+=1
print(data)

pd.set_option('display.max_columns', None)
print(result.groupby(level=0,axis=1).mean())

result.loc[:, ~result.columns.isin(['qd err','qtikh param', 'qtrunc param','qtrunc num'])]\
.groupby(level=0,axis=1).mean().plot.bar();
```

B.6 2dimages.py

```
import importlib as imp
import numpy as np
import matplotlib.pyplot as mp
import untitled0 as u
imp.reload(u)
from sys import argv

from skimage.data import shepp_logan_phantom,camera
from skimage.transform import radon, rescale
#PART 1: Constructing and testing matrix A in A*phi=q

#how many internal grid points
m = 158
#m=203 #alternative m for reconstructing camera() instead of phantom

# construct matrix A
A = u.getA(m)

#discretize phi
q = rescale(shepp_logan_phantom(), scale=0.4, mode='reflect', multichannel=False)
#q = rescale(camera(), scale=0.4, mode='reflect', multichannel=False)

#compute q = A*phi
phi = np.matmul(np.linalg.inv(A),q)


fig, ax = mp.subplots(2, 2, figsize=(8, 4.5), sharey=False)
ax[0,0].imshow(q, cmap=mp.cm.Greys_r, extent=(0, 1, 0, 1))
ax[0,0].set_title("true")

#PART 2: adding noise
```

```python
delta = float(argv[1])
#delta = float(input("Noise level to be added: "))
phi_del = u.noisy_add(phi,delta)

#compute q from noisy measurements
q_del = np.matmul(A,phi_del)
ax[0,1].imshow(q_del, cmap=mp.cm.Greys_r, extent=(0, 1, 0, 1))
ax[0,1].set_title("regular inversion")
err = np.linalg.norm(q-q_del)/np.linalg.norm(phi_del)
print("q vs q del error: ", err)

#PART 3
#truncated solution
#choosing best truncation parameter
trunc_err, alt, trunc_param, numb= u.best_trunc(A,phi_del,q)
ax[1,0].imshow(alt,cmap=mp.cm.Greys_r,extent=(0,1,0,1))
ax[1,0].set_title("truncated inverse")

#PART 4
#Calculating picard condition
U, s, v = np.linalg.svd(np.linalg.inv(A))
Picard, Picard_del = u.Picard_check(U,phi,phi_del)

#PART 5
#Tikhonov regularization
ans,a,NORM = u.best_Tikh(U,s,v,phi_del,q)

ax[1,1].imshow(ans,cmap=mp.cm.Greys_r,extent=(0,1,0,1))
ax[1,1].set_title("Tikhonov")
print("parameter alpha: ",a)
print("error in tikhonov: ", NORM)

#PART 6
#Visualize bias variance tradeoff for TSVD
error, bias, variance = u.bias_var(U, s, v,phi,phi_del,q)

k = np.linspace(0,m+1,m+2)
mp.figure(5)
mp.title("Bias Variance Trade-Off, delta ="+str(delta)+", m = "+str(m))
mp.plot(k,bias,label='bias')
mp.plot(k,variance,label='variance')

mp.xlabel('number of singular values cut off')
mp.ylabel('error')
#note that limits are best adjusted manually depending on value of delta
```

```python
#mp.ylim([0,10])
#mp.xlim([0,10])

mp.legend()
mp.show()

print("condition number of A: ",np.linalg.cond(A))
```

B.7 untitled1.py

```python
from numpy import random
import pandas as pd

list_delta = [0.1,0.01,0.001,0.0001,0]

def data_collection(r=0,list_delta=list_delta):
    df = pd.DataFrame()
    first_row=[]
    qd_error_list=[]
    qtrunc_error_list=[]
    qtrunc_param_list=[]
    qtrunc_num_list=[]
    qtikh_error_list=[]
    qtikh_param_list=[]
    for i in list_delta:
        random.seed(r)
        first_row.append(tuple([i]))
        this = str(i)
        runfile('C:/Users/jocel/Thesis/2dimages.py', args=this,\
        wdir='C:/Users/jocel/Thesis')
        qd_error_list. append(float('%.5g'%err))
        qtrunc_error_list.append(float('%.5g'%trunc_err))
        qtrunc_param_list.append(float('%.2g'%trunc_param))
        qtrunc_num_list.append(numb)
        qtikh_error_list.append(float('%.5g'%NORM))
        qtikh_param_list.append(float('%.2g'%a))
    df = pd.DataFrame(
            {"qd err" : qd_error_list,
             "qtrunc err" : qtrunc_error_list,
             "qtrunc param": qtrunc_param_list,
             "qtrunc num": qtrunc_num_list,
             "qtikh err": qtikh_error_list,
             "qtikh param": qtikh_param_list},
            index = pd.MultiIndex.from_tuples(first_row
                 , names=["delta"]))
    return df

trials = 5

i=1
data=data_collection(0)
result=data
while i < trials:
    one_trial = data_collection(i)
    data = pd.merge(data,one_trial,on=["delta"])
    result = pd.concat([result,one_trial],axis=1)
    i+=1
pd.set_option('display.max_columns', None)
print(data)
print(result.groupby(level=0,axis=1).mean())
result.loc[:, ~result.columns.isin(['qd err','qtikh param', 'qtrunc param', 'qtrunc num'])]\
.groupby(level=0,axis=1).mean().plot.bar();
```