Inverse Problems in Proton Beam Imaging at ESS

Analysis and Numerical Methods

Joel Henriksson

Master's thesis
2026:E12

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Numerical Analysis

Centrum Scientiarum Mathematicarum

Inverse Problems in Proton Beam Imaging at ESS

Analysis and Numerical Methods

Joel Henriksson

Master's Thesis in Engineering Mathematics

Lund University, Faculty of Engineering (LTH)

European Spallation Source (ESS)

January 28, 2026

Abstract

The imaging system at the European Spallation Source (ESS) records beam-on-target images that contain information about the proton beam's shape and motion. In this thesis, that information is extracted using a mathematical model of the rastered beam and two different parameter estimation methods. Concretely, we are trying to solve an *inverse problem*: given an observed image, what were the underlying parameters that produced it?

In the first part of the thesis, the structure of the inverse problem itself is analyzed. In particular, four distinct cases of non-identifiability are established. One of these cases is due to long pulse times, for which a limiting distribution is derived using Birkhoff's Theorem from ergodic theory. The identifiability issues impose fundamental limitations on what any of the numerical methods can achieve, which is demonstrated in practice throughout the rest of the thesis.

In the second part, the problem is treated as a deterministic optimization problem. The parameters are estimated by minimizing the least-squares difference between the simulated and observed images, using a gradient-based numerical solver applied from multiple initial points. This multistart quasi-Newton method reliably finds accurate solutions from a subset of the initial points.

In the third part, the problem is formulated in a Bayesian framework. Since the image formation process is affected by Poisson noise, the parameters are treated as random variables and the goal is to approximate their posterior distribution. To do this, a parallel tempering Markov Chain Monte Carlo (MCMC) method is implemented, which in addition to point estimates provides uncertainty quantification.

The two methods are compared using a set of simulated test images, from which we conclude that both methods are effective in producing accurate parameter estimates. The next natural step is to continue testing on real images, after which the methods can be deployed in practice at ESS for beam tuning and potentially machine protection.

Acknowledgements

This master's thesis was carried out at the European Spallation Source in Lund during the autumn of 2025.

I would first like to thank my main supervisor at ESS, Cyrille Thomas, for giving me the opportunity to work on this exciting project and for his constant willingness to explain and discuss physics- and mathematics-related topics. I would also like to thank my co-supervisor at ESS, Karin Rathsman, for giving me such a warm welcome to ESS and for contributing her extensive knowledge of master's thesis writing and common pitfalls. Last but not least among my supervisors, I would like to thank Eskil Hansen at LTH for providing excellent feedback on all mathematical aspects of the thesis.

In addition to my supervisors, I have also received valuable feedback and support from Johan Lindström at Mathematical Statistics, Monika Eisenmann at Numerical Analysis, and Jörg Schmeling at Dynamical Systems. Thank you.

Finally, I would like to thank my family and friends for their support throughout this work. I could not have done it without you. A special thanks to my brother Oskar for being a constant inspiration in mathematics and for reminding me to stay curious and enjoy learning.

Lund, December 2025
Joel Henriksson

Contents

Chapter 1

Introduction

1.1 Background

The European Spallation Source (ESS) is a research facility currently in commissioning in Lund, Sweden. Once operational, it will enable detailed studies of materials using neutron beams. These beams are generated by accelerating protons and directing them onto a rotating tungsten target. When the high-energy protons collide with the tungsten nuclei, they induce a nuclear reaction known as *spallation*, in which multiple neutrons are ejected from the nuclei. These free neutrons are then collected and moderated (slowed down) to energies suitable for interacting with the atoms in a material sample. At the final step, the effects of these interactions are studied, from which conclusions can be drawn about the structure of materials at the atomic scale. This allows for new fundamental discoveries in chemistry and biology, with applications in, for example, medicine, energy and infrastructure.

One of the central sections of the ESS facility works with proton beam diagnostics. This means designing tools for measuring key properties of the beam to make sure it behaves as expected. Three such instruments are called the Aperture Monitor, the GRID and the IMG system, which complement and cross-validate each other. This thesis is dedicated to the analysis of the IMG system. The basic principle behind this system is the following: During operation, the beam is rastered back and forth over the target to avoid concentrating too much energy (and thus heat) on individual spots. While this rastering takes place, the IMG system captures an image of the time-integrated intensity trace that the beam leaves on the target. Intuitively, this trace contains a lot of information about the shape of the beam and how it has moved over time. What is missing is a proper mathematical analysis and development of numerical methods to extract this information, which is what this thesis aims at providing.

Finding effective methods to estimate beam parameters related to beam shape and raster motion would be particularly valuable during beam tuning, where magnet settings are adjusted iteratively based on diagnostic measurements. Further, accurate parameter estimates would help calculate heat loads on the target, thereby contributing to machine protection.

1.2 Objectives

The main research questions we want to answer in this thesis are:

(a) Which parameters can be reliably inferred from proton beam images?

(b) How can different numerical methods (e.g. optimization, statistical inference, or machine learning) be used to estimate the parameters?

(c) How can the strengths, limitations, and behavior of these methods be analyzed and compared?

1.3 Code availability

All code used to produce the simulations and inference results is publicly available at `https://github.com/juhelh/ess-beam-imaging-inverse-problems`.

1.4 Notational conventions and definitions

- $\mathbf{k} \in \mathbb{R}^d$ denotes the beam parameter vector. Unless explicitly stated, the vector $\mathbf{k}$ refers to the full $d = 10$ parameter vector. Specifically, these parameters are horizontal and vertical raster amplitudes (A_x, A_y), beam widths (σ_x, σ_y), beam center offsets (c_x, c_y), raster frequencies in kHz (f_x, f_y), and raster phases (ϕ_x, ϕ_y). In some numerical experiments we use $d = 8$ or $d = 6$ by fixing frequencies and phases.
- We use $\mathbf{k}^*$ to denote the true underlying parameters for an observed (synthetic) image $\tilde{I}_{\text{obs}}$.
- The time t in the integration of the beam is measured in ms, and the coordinates in the image plane are measured in mm.
- Images are denoted $I(x, y)$ and represent the number of photons per unit area. Note, however, that figures will typically be normalized for practical reasons (and the fact that scale does not matter in an optimization setting). To differentiate between the observed noisy image and the one the model produces given a parameter vector $\mathbf{k}$ we write $\tilde{I}_{\text{obs}}$ and I_{model}, respectively.
- The nominal parameters used throughout this thesis are

$$\mathbf{k}_{\text{nom}} = [60.0,\ 20.0,\ 13.5,\ 5.05,\ 0.0,\ 0.0,\ 39.55,\ 29.05, 0, 0].$$

  These correspond to a "typical" image in the numerical experiments.

Typographic convention. To clearly distinguish general theoretical background from results that are specific to this thesis, the latter will in general be presented in boxed form.

1.5 Contributions

The main contributions of this thesis are the following:

- An analysis of identifiability for beam parameter estimation from rastered images, including four limitations to uniqueness: periodic (or near-periodic) rastering, finite image resolution, short pulse lengths and long pulse lengths (Section 3.3–3.4).
- A derivation of the long-time limiting distribution of the rastered beam image for aperiodic rastering using ergodic theory, showing independence from raster frequencies and phases, together with a closed-form expression (Section 3.4).
- An implementation of an optimization pipeline in JAX for beam parameter estimation (Section 1.3).
- A derivation of the expected minimum attainable loss in the presence of Poisson noise, providing a reference level for optimization-based estimation. (Section 3.6).
- An implementation of a parallel tempering Markov Chain Monte Carlo method in JAX for Bayesian inference of beam parameters (Section 1.3).
- An evaluation of the proposed methods on synthetic images (Section 4.5 and Section 5.6).
- An initial evaluation on real beam images, together with suggestions for further improvements (Section 6.1 and 7.3).

1.6 Thesis structure

The remainder of the thesis is organized as follows:

- **Chapter 2** describes the underlying model of how an image is generated by sweeping the proton beam over time, and formulates the inverse problem of estimating parameters from the observed image.
- **Chapter 3** introduces the theoretical foundations of inverse problems and applies them to the beam imaging context. It addresses fundamental questions about identifiability and limitations in parameter recovery.
- **Chapter 4** presents the optimization-based estimation approach, focusing on the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, convergence behavior, and numerical experiments that evaluate the method's performance.

- **Chapter 5** develops the Bayesian inference framework using Markov Chain Monte Carlo methods, and more specifically Metropolis–Hastings and Parallel Tempering. Similar experiments to those in the optimization chapter are used for comparison.
- **Chapter 7** summarizes and compares the results from the different estimation methods, discusses practical implications for ESS, and outlines directions for future research.

Chapter 2

Model and problem formulation

In this chapter, we present the mathematical model for how the beam deposits its intensity on the screen over time. This leads up to the formulation of a parameter estimation problem in Section 2.2. Finally, we describe how the model is implemented on a computer, and how methods for solving the estimation problem are evaluated.

2.1 The beam model

The core of the beam model is described in Adli et al. (2022). The main assumption of this model is that the beam can be approximated as a two-dimensional Gaussian distribution centered at a time-dependent position $\mathbf{r}(t; \mathbf{k})$ on the screen. This position varies over time according to a known so-called *rastering* pattern, where rastering is the technical term for moving the beam back and forth using electromagnets. By doing this during a time interval $[0, T]$, an image representing the final energy distribution is generated.

To characterize the state of the beam we define a full parameter vector $\mathbf{k} \in \mathbb{R}^{10}$ that includes both the shape of the beam and the parameters defining its time-dependent trajectory. Specifically, we let

$$\mathbf{k} = [A_x, A_y, \sigma_x, \sigma_y, c_x, c_y, f_x, f_y, \phi_x, \phi_y]$$

where

- A_x, A_y are the amplitudes of the raster motion in the x and y directions, expressed in mm.
- σ_x, σ_y are the Gaussian beam widths in x and y, expressed in mm.
- c_x, c_y are the center offsets of the beam, expressed in mm.
- f_x, f_y are the raster frequencies, expressed in kHz to have them in the same order of magnitude as the rest of the parameters.
- ϕ_x, ϕ_y are the phase parameters, expressed in radians.

2.1.1 Beam intensity

At any moment in time $t \in [0, T]$, the beam intensity at screen location $(x, y) \in \mathbb{R}^2$ is given by a two-dimensional, scaled Gaussian distribution centered at the beam position $\mathbf{r}(t; \mathbf{k})$:

$$\mathcal{B}(x, y, t; \mathbf{k}) = \frac{C}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{(x - r_x(t; \mathbf{k}))^2}{2\sigma_x^2} - \frac{(y - r_y(t; \mathbf{k}))^2}{2\sigma_y^2}\right).$$

Here, $r_x(t; \mathbf{k})$ and $r_y(t; \mathbf{k})$ denote the x and y components of the time-dependent beam center trajectory, which is described below. The constant C is assumed known from measurements and represents the expected total number of photons per unit time (over the entire screen). For all the simulations made in this thesis this will be assumed to be $C = 20\,000$ for simplicity, which matches typical brightness levels of the real proton beam images.

2.1.2 Raster trajectory

The beam follows a deterministic time-varying path across the screen. This trajectory is modeled as a triangle wave in both directions:

$$\mathbf{r}(t; \mathbf{k}) = \begin{bmatrix} A_x \cdot \mathrm{tri}(2\pi f_x t + \phi_x) + c_x \\ A_y \cdot \mathrm{tri}(2\pi f_y t + \phi_y) + c_y \end{bmatrix}$$

where $\mathrm{tri}(\theta)$ denotes a continuous triangle-wave function given by

$$\mathrm{tri}(\theta) = \frac{2}{\pi} \arcsin(\sin\theta).$$

This produces a periodic waveform ranging from -1 to $+1$, with fundamental frequency f. An example plot of what the raster trajectory may look like is shown in Figure 2.1. Note that there are several other ways of expressing the triangle function: another common way is to define it as $\mathrm{tri}(\theta) = 4\left|(\theta \bmod 1) - \frac{1}{2}\right| - 1$.

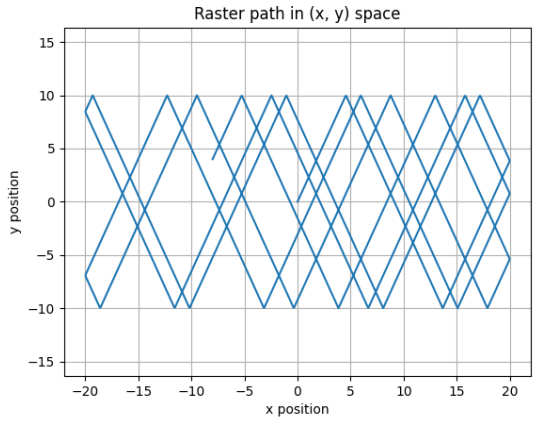

Figure 2.1: Raster scan trajectory in the (x, y) plane generated by triangle-wave motion in each direction.

2.1.3 Forward model (image formation)

The final observed image is formed by integrating the beam intensity $\mathcal{B}$ over the full exposure time T:

$$I_{\text{model}}(x, y; \mathbf{k}) = \int_0^T \mathcal{B}(x, y, t; \mathbf{k})\, dt \tag{2.1}$$

This defines the forward model $\mathcal{F} : \mathbf{k} \mapsto I_{\text{model}}$, which we aim to invert. Note that the unit of I_{model} is photons per unit area, which means that when we later introduce pixels and integrate over pixel areas, we indeed get number of photons (i.e. a count).

2.1.4 Assumptions in the model

The model is simplified in relation to reality in several ways to make it practical to work with. The main assumptions are:

- The beam has a two-dimensional Gaussian profile at all times.
- The raster trajectory $\mathbf{r}(t; \mathbf{k})$ is known exactly and parameterized by $\mathbf{k}$.
- The beam's intensity is constant over time and independent of (x, y).
- Each parameter is defined on a closed and bounded interval in $\mathbb{R}$. We let the full parameter vector be

  $$\mathbf{k} = (A_x, A_y, \sigma_x, \sigma_y, c_x, c_y, f_x, f_y, \phi_x, \phi_y),$$

  and define the parameter domain as the Cartesian product

  $$\Omega = \prod_{i=1}^{10} I_i \subset \mathbb{R}^{10},$$

  where the intervals I_i are given by

  $$\begin{aligned} I_1 &= [0, 65], & I_2 &= [0, 25], \\ I_3 &= [2, 20], & I_4 &= [2, 20], \\ I_5 &= [-20, 20], & I_6 &= [-20, 20], \\ I_7 &= [20, 50], & I_8 &= [20, 50], \\ I_9 &= [-\pi, \pi], & I_{10} &= [-\pi, \pi]. \end{aligned}$$

These assumptions give us a simple forward model that is suitable for testing optimization and other inference methods. The figures 2.2 and 2.3 below show the beam profile and the rastered image, respectively, to illustrate what some of the assumptions lead to in practice.

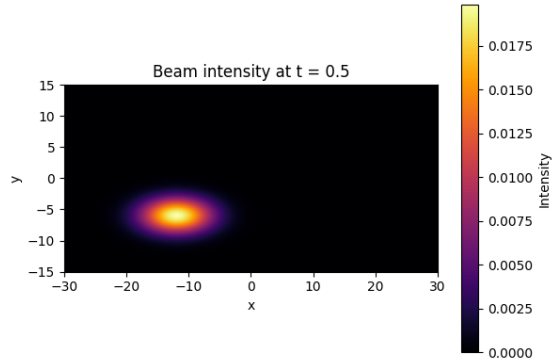

Figure 2.2: Beam intensity profile at sampling time $t_i = 0.5$ ms, evaluated on a grid $x \in [-30, 30]$ and $y \in [-15, 15]$.

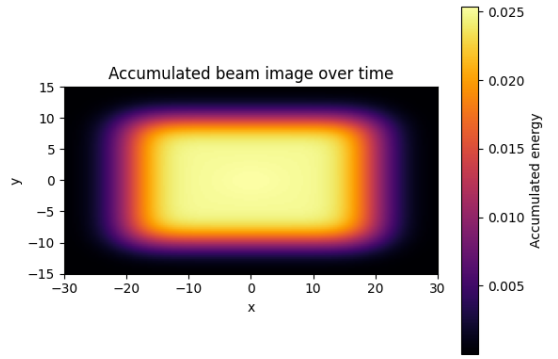

Figure 2.3: Simulated (normalized) beam image formed by integrating the Gaussian beam intensity over time, using the same beam parameters as in Figure 2.2.

2.2 A first problem formulation

2.2.1 Assuming noise-free images

We formulate the problem of inferring beam parameters as an inverse problem (see Section 3.1). This essentially means that we have a model of how images are generated that depends on the parameters and an observed image that follows the model reasonably well, and want to answer the question "Which underlying parameters $\mathbf{k}^*$ caused the image?". One of the main ways such a problem can be formulated is as a minimization problem, where the quantity to minimize is the distance between the model and the data. For our particular problem, we want to find a parameter vector $\mathbf{k} \in \Omega$ that minimizes the integrated squared distance between the observed image $I_{\text{obs}} = I_{\text{model}}(\mathbf{k}^*)$ and the model image $I_{\text{model}}(\mathbf{k})$:

$$\arg\min_{\mathbf{k}\in\Omega} \mathcal{L}(\mathbf{k}) = \arg\min_{\mathbf{k}\in\Omega} \int_{\mathbb{R}^2} \left[I_{\text{obs}}(x,y) - \int_0^T \mathcal{B}(x,y,t;\mathbf{k})\, dt \right]^2 dx\, dy \tag{P}$$

The mapping $\mathcal{L} : \mathbf{k} \to \mathbb{R}$ will be referred to as the *loss function*. Note that the integral over $\mathbb{R}^2$ corresponds to the L^2 norm, which means that the loss function may also be written as

$$\mathcal{L}(\mathbf{k}) = \left\| I_{\text{obs}} - I_{\text{model}}(\mathbf{k}) \right\|_2^2 = \int_{\mathbb{R}^2} \left[I_{\text{obs}}(x,y) - \int_0^T \mathcal{B}(x,y,t;\mathbf{k})\, dt \right]^2 dx\, dy. \tag{2.2}$$

The L^2 loss is not the only reasonable choice of loss, though it is standard in applications like these due to nice properties such as preserving differentiability, which is important in optimization. Sometimes in this thesis we will also reference the possibility of adding a regularization term $R(\mathbf{k})$ to incorporate prior knowledge in the optimization. An example of this would be

$$R(\mathbf{k}) = \gamma(k_i - k_i^{\text{prior}})^2 \tag{2.3}$$

for selected components of $\mathbf{k}$, with regularization weight $\gamma > 0$. This particular regularization term would help drive the solution towards the expected parameter.

2.2.2 Assuming Poisson noise in images

The real image will have noise due to the fact that photons arrive stochastically to the camera. This uncertainty is commonly modeled (Mandel and Wolf 1995) by assuming that the number of photons recorded in each pixel (i,j) follows a Poisson distribution:

$$N_{ij} \sim \text{Poisson}(\lambda_{ij}), \qquad \lambda_{ij}(\mathbf{k}) = \int_{\text{pixel}_{ij}} I_{\text{model}}(x,y;\mathbf{k})\, dx\, dy, \tag{2.4}$$

where $I_{\text{model}}(x,y;\mathbf{k})$ is the expected photon density (photons per unit area). We then define the observed image as a piecewise constant density

$$\tilde{I}_{\text{obs}}(x,y) = \frac{N_{ij}}{\Delta x\, \Delta y}, \qquad (x,y) \in \text{pixel}_{ij}.$$

This makes it simple to formulate the corresponding noisy minimization problem to (P). To do this, we insert the definition of the noisy image instead of the noiseless one:

$$\arg\min_{\mathbf{k}\in\Omega} \left\| \tilde{I}_{\text{obs}} - I_{\text{model}}(\mathbf{k}) \right\|_2^2. \tag{$\tilde{P}$}$$

The problem $(\tilde{P})$ will be the main problem we reference throughout the optimization part of the thesis. Note, however, that the fact that we introduce randomness at this point opens up the possibility of a statistical formulation of the inverse problem. Indeed such an alternative

formulation will be introduced later in Chapter 5.

Remark. Minimization with the L^2 norm under Poisson noise technically gives a *biased* estimator of the parameters. However, this bias is expected to be negligible in comparison to model mismatch and other sources of error.

2.3 Discretization; general notes on implementation

2.3.1 Discretization

The time integral in (2.1) is in practice approximated by discrete sums using a very small timestep $\Delta t = 10^{-3}$ ms. In particular,

$$\int_0^T \mathcal{B}(x, y, t; \mathbf{k})\, dt \;\approx\; \sum_{i=0}^{N} \mathcal{B}\big(x, y, i\,\Delta t; \mathbf{k}\big)\, \Delta t, \tag{2.5}$$

where $N = T/\Delta t$. Choosing $\Delta t = 10^{-3}$ ms ensures both sufficient numerical accuracy and that the high-frequency components of the raster motion (which lie in the kHz range) are captured without aliasing. We also discretize the spatial domain, so that the positions (x, y) are sampled on a rectangular grid corresponding exactly to the pixel layout of the images. In other words, the continuous coordinates are replaced by

$$x = x_i, \qquad y = y_j,$$

where x_i and y_j denote the coordinates of the pixel at position (i, j) in the spatial grid. With this spatial discretization, the beam intensity $\mathcal{B}$ becomes a matrix $B \in \mathbb{R}^{N_x \times N_y}$ defined entry-wise by

$$B(i, j) = \mathcal{B}\big(x_i,\, y_j,\, t; \mathbf{k}\big), \qquad 1 \le i \le N_x,\;\; 1 \le j \le N_y.$$

Further, the discretized version of the loss function in $(\tilde{P})$ becomes

$$\mathcal{L}_{\text{disc}}(\mathbf{k}) = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \Big(\tilde{I}_{\text{obs}}(i, j) - I_{\text{model}}(i, j; \mathbf{k})\Big)^2\, \Delta x\, \Delta y, \tag{2.6}$$

where Δx and Δy are the pixel widths and heights, respectively. In this thesis we will however work with the *relative squared error* (RSE), defined as

$$\mathcal{L}_{\text{rel}}(\mathbf{k}) = \frac{\sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \Big(\tilde{I}_{\text{obs}}(i, j) - I_{\text{model}}(i, j; \mathbf{k})\Big)^2}{\sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \Big(\tilde{I}_{\text{obs}}(i, j)\Big)^2}, \tag{2.7}$$

where the factor $\Delta x \Delta y$ cancels. This loss offers a more intuitive interpretation than (2.6); when multiplied by 100, it measures the percentage discrepancy between the estimated image and the observed one. It also keeps the magnitude of the loss values numerically well scaled across images of different intensity.

2.3.2 Computational speed and parallelism with JAX

Performing computations on high resolution images with integration over fine time steps is inherently computationally heavy. Parallel computing and GPU use is crucial for reasonable computing times for both estimation methods that are used in this thesis. To achieve this the JAX library in Python (Bradbury et al. 2018) has been used. For similar applications where this library has been used successfully, see for example Wen et al. (2025) for optimization and Baudart, Mandel, and Tekin (2022) for Bayesian inference.

2.4 Evaluation of estimation methods

For numerical experiments and evaluation of the different estimation methods, we primarily use simulated images. In this section we describe how the sets of such images are constructed. A few examples of real images (generated by a laser in the ESS optics lab) will also be used for initial validation in Chapter 6; these more closely resemble what the actual proton beam images look like.

2.4.1 Synthetic data for numerical experiments

For numerical experiments, we use the nominal parameter vector $\mathbf{k}$ defined in Section 1.4 and three different images generated from these using pulse lengths $T = 0.05, 0.5, 3$ ms, i.e. short, medium and long pulse length. These images are used for preliminary testing of the estimation methods and numerically investigating properties like noise robustness, convexity and convergence.

2.4.2 Synthetic data for evaluation

To quantify how well our methods are able to estimate the true parameters for an observed image, we define the following test sets below. All of the generated images have Poisson noise as described in Section 2.2.2. For further details, such as the choices of standard deviations in the Gaussian distributions, see Appendix A).

- **Synthetic–uniform**: 10 different parameter vectors $\mathbf{k}$ with corresponding images, where the parameters are drawn uniformly from their allowed intervals. This test set is used to

determine how well the estimation methods perform in a scenario where we have no (or very little) prior knowledge about what the true parameter values are.

- **Synthetic–prior**: 10 different parameter vectors $\mathbf{k}$ with corresponding images, where the parameters are drawn from Gaussian distributions around "reasonable" values, i.e. values that the machine will typically be set to in practice. For this test we may assume that we have this prior knowledge of what the true values should be, and can thus make use of that in the estimation methods themselves (which both numerical optimization and Markov Chain Monte Carlo allows for).

Chapter 3

Analysis of the inverse problem

In Section 2.2 we mentioned that the problem of estimating parameters from our images may be classified as an *inverse problem*. This chapter aims at clarifying what that means, and how the framework of inverse problems can be used to answer some fundamental questions. For example, is it even possible to solve $(\tilde{P})$ in general? Which limitations exist? Are there some parameters that cannot be estimated?

3.1 Theory of inverse problems

3.1.1 What is an inverse problem?

An inverse problem refers to the task of estimating unknown model parameters $\mathbf{k}$ from observations $\mathbf{d}$, based on a known forward model $\mathcal{F}$ that maps parameters to data according to $\mathbf{d} = \mathcal{F}(\mathbf{k})$. The data is typically noisy in practice, so a more accurate description of the relationship between data and parameters is that

$$\mathbf{d} = \mathcal{F}(\mathbf{k}^*) + \boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon}$ denotes the noise, and $\mathbf{k}^*$ is the unknown true parameter vector. This definition is used in (Aster, Borchers, and Thurber 2018). Some common examples of inverse problems include

- Medical imaging: Reconstructing a three-dimensional structure from tomography (National Research Council 1996).
- Seismology: Locating the source of an earthquake or estimating its strength from measured wave signals (Apostol 2018).
- Heat transfer: Determining how temperature was distributed in a material at earlier times based on measurements taken later (Borukhov and Zayats 2024).

Inverse problems are a broad category of mathematical problems, and it is therefore useful to try to differentiate between subsets of them, as done in Aster, Borchers, and Thurber (2018). Some of the features one can base this categorization on are the following:

- Linear vs. Nonlinear. Depending on whether the forward model $\mathcal{F}$ is a linear operator or not.
- Continuous vs. Discrete. Whether the data and model/parameters are functions or finite-dimensional vectors.
- Few vs. Many parameters. Problems with few parameters are often called *parameter estimation problems*, while high-dimensional or function-valued problems are more generally referred to as inverse problems.

Based on these features, our beam problem can be categorized as a nonlinear, discrete inverse problem. Due to the fact that the number of parameters is rather small to moderate, we might as well simply call it a parameter estimation problem, but since a lot of the general framework of inverse problems is still widely used (both in literature and in this thesis), we will use the two different terms more or less interchangeably.

3.1.2 Characteristics and main perspectives

One of the fundamental challenges of inverse problems are that they are often ill-posed. This means that they tend to violate one or more of the three following conditions, stated in Aster, Borchers, and Thurber (2018):

1. A solution exists.
2. The solution is unique.
3. The solution depends continuously on the data.

In other words a solution may not exist at all, there may be multiple solutions, or a small change in the observation may correspond to a huge change in the underlying parameters. To deal with ill-posedness one typically has to use some kind of *regularization* as part of the solution method, which essentially means incorporating some kind of extra information that constrains the problem, so that for example multiple solutions are no longer possible. The concept of regularization is something that we will come back to many times throughout the thesis.

In most literature there are two main approaches when it comes to formulating an inverse problem and coming up with a solution. One of those approaches is the *deterministic* one, in which the objective is to find the parameters that best fit the data, for example by minimizing the squared distance between data and model. Here, we see the parameters as something that

is exact but unknown. The other main approach is a *Bayesian* one, where we in contrast see parameters as being more or less likely, and are thus interested in finding the probability distribution of parameters, given the data that is observed. Both the deterministic and the Bayesian perspective are valid and will be used in this thesis.

3.2 Fundamental properties: existence, uniqueness and stability

We begin the investigation of our beam problem by looking at the three fundamental properties of the inverse problem, described in Section 3.1. The particular problem formulation we are studying here is $(\tilde{P})$, where the observed image $\tilde{I}_{\text{obs}}$ has Poisson noise.

3.2.1 Definitions

We first formalize the concepts of existence and uniqueness in the context of the minimization problem $(\tilde{P})$. Here, $\mathcal{L}$ denotes the continuous loss function defined in (2.2), whereas $\Omega \subset \mathbb{R}^d$ is the parameter domain.

Definition 3.1 (Existence)**.** We say that the problem $(\tilde{P})$ has a solution for a given observation $\tilde{I}_{\text{obs}}$ if there exists at least one parameter vector $\mathbf{k}^* \in \Omega$ such that

$$\mathcal{L}(\mathbf{k}^*) \leq \mathcal{L}(\mathbf{k}) \quad \text{for all } \mathbf{k} \in \Omega.$$

Note that there can be many different parameter vectors that minimize the problem if we only have existence. That is why we are also interested in the notion of uniqueness.

Definition 3.2 (Uniqueness)**.** We say that a solution $\mathbf{k}^*$ to the problem $(\tilde{P})$ is *unique* if it is the only parameter vector that minimizes the loss function, that is, for any $\mathbf{k} \in \Omega$,

$$\mathcal{L}(\mathbf{k}) = \mathcal{L}(\mathbf{k}^*) \quad \Rightarrow \quad \mathbf{k} = \mathbf{k}^*.$$

Finally, there is the even more demanding notion of stability of solution. This property will not be used in this thesis, due to the fact that it requires the inverse map $\mathcal{F}^{-1}\colon I \mapsto \mathbf{k}$ to exist, which we will show is not the case. For completeness, however, we state the definition below.

Definition 3.3 (Stability)**.** We say that the inverse problem is *stable* at a point $\mathbf{k}^* \in \Omega$ if for every $\varepsilon > 0$, there exists $\delta > 0$ such that

$$\|\mathcal{F}(\mathbf{k}) - \mathcal{F}(\mathbf{k}^*)\| < \delta \quad \Rightarrow \quad \|\mathbf{k} - \mathbf{k}^*\| < \varepsilon.$$

Here, $\|\cdot\|$ on the left-hand side denotes the L^2-norm in the image domain, and on the right-hand side the ℓ^2 norm in parameter space.

Remark. In literature, for example in (Preston et al. 2025), it is also common to use the word *identifiability* instead of uniqueness, which has the same essential meaning but puts more emphasis on the parameters themselves. If like in this case we do not have uniqueness, one can alternatively say that "the parameters are not identifiable", or, if the solution is in theory unique but in practice different parameters can produce very similar results, "the parameters are weakly identifiable".

3.2.2 Proofs: existence and non-uniqueness

Now we are ready to prove which properties we have for our particular problem $(\tilde{P})$. For simplicity, let us assume that the regularization term is zero. We start by showing that there always exists a solution.

Proposition 3.1 (Existence of minimizer)**.** For any observed image $\tilde{I}_{\text{obs}}$, there exists a parameter vector $\mathbf{k}^* \in \Omega$ such that $\mathcal{L}(\mathbf{k}^*) \leq \mathcal{L}(\mathbf{k})$ for all $\mathbf{k} \in \Omega$.

Proof. The loss function

$$\mathcal{L}(\mathbf{k}) = \left\| \tilde{I}_{\text{obs}} - I_{\text{model}}(\mathbf{k}) \right\|_2^2$$

is continuous in $\mathbf{k}$, since $I_{\text{model}}(\mathbf{k})$ depends continuously on the parameters. Moreover, the parameter domain Ω is compact, since each parameter is defined on a closed and bounded interval (see Section 2.1.4). By the Weierstrass extreme value theorem, any continuous function on a compact set attains its minimum (and maximum). Therefore, there exists at least one $\mathbf{k}^*$ such that $\mathcal{L}(\mathbf{k}^*)$ is minimal, and the problem $(\tilde{P})$ has a solution. □

Next, we are going to prove that in general we do not have uniqueness. The strategy to prove this will be to provide concrete examples of non-uniqueness for the noise-free problem (P), which via the lemma below is sufficient for proving non-uniqueness of $(\tilde{P})$.

Lemma 3.1 (Non-uniqueness in (P) sufficient)**.** *If the noise-free inverse problem* (P) *does not have a unique solution, then the noisy inverse problem* $(\tilde{P})$ *does not have a unique solution either.*

Proof. Non-uniqueness of solution for (P) means that there exist $\mathbf{k}_1 \neq \mathbf{k}_2$ such that $I_{\text{model}}(\mathbf{k}_1) = I_{\text{model}}(\mathbf{k}_2) = I_{\text{obs}}$. Consider $(\tilde{P})$ under the particular noise realization $\tilde{I}_{\text{obs}} = I_{\text{obs}}$, i.e. the rare case where each pixel value becomes equal to the expected value of its Poisson distribution. Then the loss

$$\mathcal{L}(\mathbf{k}) = \left\| \tilde{I}_{\text{obs}} - I_{\text{model}}(\mathbf{k}) \right\|_2^2$$

satisfies $\mathcal{L}(\mathbf{k}_1) = \mathcal{L}(\mathbf{k}_2) = 0$, so both $\mathbf{k}_1$ and $\mathbf{k}_2$ are minimizers of $(\tilde{P})$. Hence the solution to $(\tilde{P})$ is not unique in general. □

We are now ready to prove that the solution of $(\tilde{P})$ is in general non-unique. We first state the result as a proposition.

Proposition 3.2. The parameter estimation problem $(\tilde{P})$ does not in general have a unique solution.

Proof. By Lemma 3.1, it is enough to show that there exists a problem instance of (P) for which the minimizer is not unique. To do this, we only need to find two distinct parameter vectors $\mathbf{k}_1 \neq \mathbf{k}_2$ such that $\mathcal{F}(\mathbf{k}_1) = \mathcal{F}(\mathbf{k}_2)$. This is sufficient because if we define a problem with the observed noise-free image by $I_{\text{obs}} := \mathcal{F}(\mathbf{k}_1) = \mathcal{F}(\mathbf{k}_2)$, then it follows trivially by the definition of the loss function that $\mathcal{L}(\mathbf{k}_1; I_{\text{obs}}) = \mathcal{L}(\mathbf{k}_2; I_{\text{obs}}) = 0$, i.e. that both $\mathbf{k}_1$ and $\mathbf{k}_2$ are minimizers of (P). Explicit examples of such pairs $\mathbf{k}_1, \mathbf{k}_2$ are presented in detail in Section 3.3. Therefore, by Lemma 3.1, the problem $(\tilde{P})$ is also non-unique. □

3.3 Limitations to uniqueness

3.3.1 Limitation 1: Periodic rastering

We now present a concrete example of non-uniqueness based on the possibility of generating the same raster pattern by scaling both frequencies f_x and f_y by the same amount. The conditions we need are the following:

- The original parameter vector is

$$\mathbf{k}_1 = [A_x, A_y, \sigma_x, \sigma_y, c_x, c_y, f_x, f_y].$$

- The ratio $f_y/f_x \in \mathbb{Q}$. This implies that the raster trajectory eventually returns to its initial position, because the ratio can be written in lowest terms as

$$\frac{f_y}{f_x} = \frac{N_y}{N_x}, \qquad N_x, N_y \in \mathbb{N},$$

  and after

$$T = \operatorname{lcm}(N_x, N_y)$$

  both directions have completed an integer number of full periods.

- Define a second parameter vector

$$\mathbf{k}_2 = [A_x, A_y, \sigma_x, \sigma_y, c_x, c_y, \alpha f_x, \alpha f_y],$$

  where $\alpha \in \mathbb{N}$ is a positive integer scaling factor.

- The total integration time T is such that rastering with f_x, f_y completes an integer number of cycles (this ensures that we have full cycles). Then, rastering with $\alpha f_x, \alpha f_y$ will complete exactly α times as many cycles over the same time interval.

Intuitively, the raster will spend the same amount of time in each point of the image plane in both cases. Although the beam with parameters $\mathbf{k}_2$ moves faster (by a factor α), it completes correspondingly more raster cycles within the same integration time T. Hence, every point on the target accumulates the same total energy in both cases, resulting in identical images.

To show this formally, consider the simulated image for $\mathbf{k}_2$,

$$I_{\text{model}}(x, y; \mathbf{k}_2) = \int_0^T \exp\left(-\frac{(x - A_x \cdot \text{tri}(\alpha f_x t) - c_x)^2}{2\sigma_x^2} - \frac{(y - A_y \cdot \text{tri}(\alpha f_y t) - c_y)^2}{2\sigma_y^2}\right) dt.$$

Now do a change of variables $s = \alpha t$, so that $t = s/\alpha$, $dt = ds/\alpha$, and the limits become $s \in [0, \alpha T]$. Then

$$I_{\text{model}}(x, y; \mathbf{k}_2) = \frac{1}{\alpha} \int_0^{\alpha T} \exp\left(-\frac{(x - A_x \cdot \text{tri}(f_x s) - c_x)^2}{2\sigma_x^2} - \frac{(y - A_y \cdot \text{tri}(f_y s) - c_y)^2}{2\sigma_y^2}\right) ds.$$

Here we note that the integrand is identical to that of $I_{\text{model}}(x, y; \mathbf{k}_1)$, so we may write

$$I_{\text{model}}(x, y; \mathbf{k}_2) = \frac{1}{\alpha} \int_0^{\alpha T} \mathcal{B}(x, y, s; \mathbf{k}_1)\, ds.$$

Finally, we can use the fact that the raster for parameters $\mathbf{k}_1$ is periodic with T, which means that we can split the full integral into α subintervals. We then obtain the desired result by using that the integral evaluates to the same thing on each subinterval:

$$\begin{aligned} I_{\text{model}}(x, y; \mathbf{k}_2) &= \frac{1}{\alpha} \sum_{m=0}^{\alpha-1} \int_{mT}^{(m+1)T} \mathcal{B}(x, y, s; \mathbf{k}_1)\, ds \\ &= \frac{1}{\alpha} \cdot \alpha \int_0^T \mathcal{B}(x, y, s; \mathbf{k}_1)\, ds \\ &= \int_0^T \mathcal{B}(x, y, s; \mathbf{k}_1)\, ds \\ &= I_{\text{model}}(x, y; \mathbf{k}_1). \end{aligned}$$

Below we make a numerical demonstration of the phenomenon.

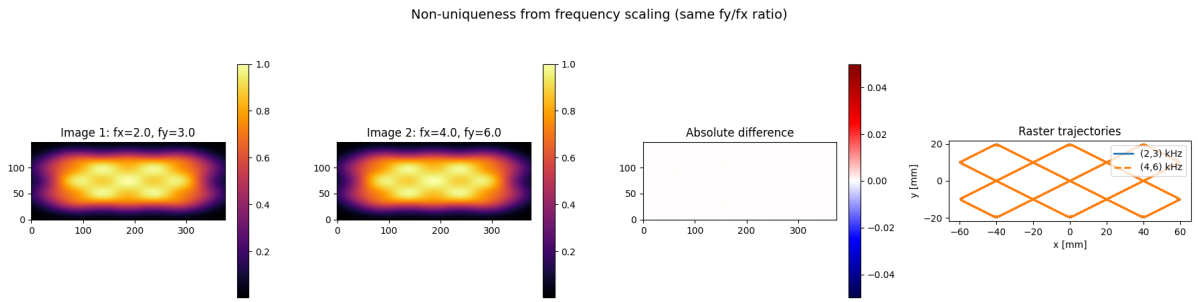

Figure 3.1: Demonstration of non-uniqueness due to frequency scaling. Two different parameter sets $(f_x, f_y) = (2, 3)$ and $(4, 6)$ produce identical images (left and middle), despite the raster motion being twice as fast in the second case.

Though the periodic rastering example requires quite specific assumptions to work out exactly, it points to something more general:

> **Limitation 1.** If $f_y/f_x \in \mathbb{Q}$, the raster frequencies may not be identifiable. By continuity of the forward model, frequencies with a ratio *close* to such a ratio will only be weakly identifiable.

The question of how a small perturbation ε of a rational ratio $f_y/f_x = p/q$ to an irrational value $\rho = f_y/f_x + \varepsilon$ affects how long the corresponding raster trajectories remain close is related to results from Diophantine approximation; see Church (2021). This theory studies how closely irrational numbers can be approximated by rational ones.

3.3.2 Limitation 2: Short pulse time

We now construct a second counterexample to uniqueness, based on an ambiguity for phase and offset in the raster trajectory. The idea is that, over a short time interval, a phase-shifted raster pattern and an offset-shifted raster pattern can lead to identical beam paths, and therefore images. We assume:

- The parameter vector $\mathbf{k}_1$ includes nonzero phases $0 < \phi_{x,1}, \phi_{y,1} < \pi/2$, and zero offsets:

$$c_{x,1} = 0, \qquad c_{y,1} = 0.$$

  This places the raster slightly into its trajectory at time $t = 0$, but before its first peak.

- The parameter vector $\mathbf{k}_2$ has zero phases, $\phi_{x,2} = \phi_{y,2} = 0$, but nonzero offsets chosen to match the beam position at time $t = 0$ under $\mathbf{k}_1$:

$$c_{x,2} = A_{x,1} \cdot \mathrm{tri}(\phi_{x,1}), \qquad c_{y,2} = A_{y,1} \cdot \mathrm{tri}(\phi_{y,1}).$$

- All other parameters than the phase and offset are shared between $\mathbf{k}_1$ and $\mathbf{k}_2$.

- The integration is performed over a short time interval $[0, \Delta t]$, with

$$\Delta t \ll \frac{1}{4f_x}, \qquad \Delta t \ll \frac{1}{4f_y}.$$

  This means we are only observing the raster trajectories before the triangle wave signals $r_{x,1}(t)$ and $r_{y,1}(t)$ reach their first peaks. At that point the rastering with $\mathbf{k}_1$ will diverge from the rastering with $\mathbf{k}_2$, since the latter is ahead in time because of the phase.

To prove that the images under $\mathbf{k}_1$ and $\mathbf{k}_2$ are equal, it is sufficient to show that their raster trajectories coincide during the interval $[0, \Delta t]$. We first analyze the x-component.

The horizontal raster trajectory under $\mathbf{k}_1$ is given by

$$r_{x,1}(t) = A_{x,1} \cdot \text{tri}(2\pi f_{x,1} t + \phi_{x,1}) + c_{x,1} = A_{x,1} \cdot \text{tri}(2\pi f_{x,1} t + \phi_{x,1}).$$

Now, using the assumption about the short time interval, we note that the triangle function behaves locally as the identity function, $\text{tri}(s) = s$. Hence, within this interval

$$r_{x,1}(t) = A_{x,1} \cdot (2\pi f_{x,1} t + \phi_{x,1}) = A_{x,1} 2\pi f_{x,1} t + A_{x,1}\phi_{x,1} = A_{x,1} 2\pi f_{x,1} t + c_{x,2},$$

where at the end we also used the assumed relationship between offset and phase. Finally, all the parameters that are not related to phase or offset are assumed to be equal for $\mathbf{k}_1$ and $\mathbf{k}_2$. Using that, and again the identity property of the triangle function, we finally get

$$r_{x,1}(t) = A_{x,1} 2\pi f_{x,1} t + c_{x,2} = A_{x,1} \cdot \text{tri}(2\pi f_{x,1} t) + c_{x,2} = A_{x,2} \cdot \text{tri}(f_{x,2} t) + c_{x,2} = r_{x,2}(t).$$

This proves that the x-components are equal. To prove that the y-components are also equal, the exact same reasoning can be used. In total we get the desired result: that $I_{\text{model}}(x, y; \mathbf{k}_1) = I_{\text{model}}(x, y; \mathbf{k}_2)$ for the assumed $\mathbf{k}_1 \neq \mathbf{k}_2$. Framed a bit differently, this is quite fundamental:

Limitation 2. There exists a lower pulse duration $T_{\text{lower}} > 0$ such that the forward model $\mathcal{F}$ is not invertible for pulse durations $T < T_{\text{lower}}$.

An interesting direction for further study is how the lower pulse duration T_{lower} depends on the parameter vector $\mathbf{k}$, for example whether it can be expressed as a function $T_{\text{lower}} = T_{\text{lower}}(\mathbf{k})$. A related and more general question is how the amount of information about the parameters contained in the image changes as the pulse duration T approaches T_{lower}. Questions like these may be possible to answer with tools from *information theory*, for example via the Fisher information matrix as demonstrated in works like Chao, Ward, and Ober (2016).

3.3.3 Limitation 3: Spatial resolution

As a third example, we demonstrate non-uniqueness due to coarse spatial resolution. When the number of pixels is small, different beam trajectories can result in the same accumulated energy per pixel.

Consider a domain with only two pixels, split vertically at $x = 0$, with pixel centers at $x = \pm x_0$, where $0 < x_0 \leq A/\sqrt{2}$. We now make the following assumptions:

- $\mathbf{k}_1$ has diagonal rastering:

  $$r_{x,1}(t) = A \cdot \mathrm{tri}(ft), \qquad r_{y,1}(t) = A \cdot \mathrm{tri}(ft),$$

  so the beam moves between bottom-left and top-right. In particular, $\phi_{x,1} = \phi_{y,1} = 0$.

- $\mathbf{k}_2$ has a mirrored trajectory:

  $$r_{x,2}(t) = A \cdot \mathrm{tri}(ft), \qquad r_{y,2}(t) = A \cdot \mathrm{tri}(ft + \pi) = -A \cdot \mathrm{tri}(ft),$$

  corresponding to motion between top-left and bottom-right. Here $\phi_{y,2} = \pi \neq \phi_{y,1}$.

- The beam shape parameters σ_x and σ_y are equal for the two beams.

The idea is that, although the beams follow different paths, they are always at the same distances from the two pixel centres $(x_0, 0)$ and $(-x_0, 0)$, and should therefore deposit the same energy. To show this, we only need to verify that the image values are equal at those two points:

$$I_{\text{model}}(x, y; \mathbf{k}_1) = I_{\text{model}}(x, y; \mathbf{k}_2),$$

evaluated at $(x, y) = (\pm x_0, 0)$. Beginning with the first point we get

$$\begin{aligned} I_{\text{model}}\left(x_0, 0; \mathbf{k}_1\right) &= \int_0^T \exp\left(-\frac{\left(x_0 - A \cdot \mathrm{tri}(ft)\right)^2}{2\sigma_x^2} - \frac{\left(0 - A \cdot \mathrm{tri}(ft)\right)^2}{2\sigma_y^2} \right) dt \\ &= \int_0^T \exp\left(-\frac{\left(x_0 - A \cdot \mathrm{tri}(ft)\right)^2}{2\sigma_x^2} - \frac{\left(0 + A \cdot \mathrm{tri}(ft)\right)^2}{2\sigma_y^2} \right) dt \\ &= I_{\text{model}}\left(x_0, 0; \mathbf{k}_2\right). \end{aligned}$$

Exactly the same kind of calculation can be made for $(x, y) = (-x_0, 0)$. Hence we conclude that the two parameter vectors $\mathbf{k}_1$ and $\mathbf{k}_2$ give rise to the same two-pixel image $I_{\text{model}}(x, y)$.

The trajectories through the two pixels we consider could also be considered as part of a much larger image of many pixels. Because of that, it is worth summarizing the result as another fundamental limitation:

Limitation 3. The discretization in space can make (parts of) different trajectories map to the same pixel values; this can lead to non-identifiability of parameters.

Further information-theoretical analysis could reveal quantitatively how identifiability depends on number of pixels N and parameters $\mathbf{k}$. Intuitively, it is for example clear that small amplitudes require more pixels to resolve differences between images.

3.4 Derivation of long pulse formula

Next we want to show that if the rastering is aperiodic, then after sufficiently long time all frequency and phase information in the image is lost; the final image does not depend on f_x, f_y and phases ϕ_x, ϕ_y, and the parameters cannot be estimated. Note how this is essentially another case of non-uniqueness, but we let this case have its own section due to its practical importance, and the fact that it requires some extra theory. Further: in this section we use the time discretized model described in 2.3, because the theoretical framework we will use is most easily stated in terms of sums. This is, however, neither a limitation in terms of the physical interpretation of the result nor for the simulations, since it holds for any Δt.

To be able to make a statement about a limiting distribution (what the image looks like as time goes to infinity), we need to consider the long-time *average* image, where we divide by the number of time points. Otherwise the infinite sum will have no chance of converging. Using the discretized model in Section 2.3, we therefore define the average image

$$I_N(x,y) = \frac{1}{N}\sum_{i=0}^{N-1} \mathcal{B}\big(x,y,t_i;\,\mathbf{k}\big)\Delta t.$$

For notational simplicity we may assume that $\Delta t = 1$, since this would only act as a constant factor in all calculations below, and hence not change the fundamental result. Further, we want to make the dependence on the raster trajectory explicit, as this mapping in time is central to the argument in this section. Making these two changes in notation we will write the average image as

$$I_N(x,y) = \frac{1}{N}\sum_{i=0}^{N-1} \mathcal{B}\big(x - r_x(t_i),\, y - r_y(t_i)\big)$$

The limiting distribution, if it exists, is then defined as

$$I_\infty(x,y) = \lim_{N\to\infty} I_N(x,y).$$

The main goal in this section is to prove the following statement. Here X denotes the state space of the raster dynamics.

Proposition 3.3 (Loss of frequency and phase information)**.** For aperiodic rastering (i.e. when $f_x/f_y \in \mathbb{R} \setminus \mathbb{Q}$.), the long-time averaged image $I_N(x, y)$ becomes independent of the raster frequencies f_x, f_y and phases ϕ_x, ϕ_y for almost every initial point $(r_x(t_0), r_y(t_0)) \in X$ as $N \to \infty$.

From this proposition, an explicit formula for the limiting distribution can be derived easily; this is stated as a corollary below. The formula can be of great practical use: it means that the final image after long rastering can be computed without any integration in time, which allows for significantly faster simulations and inference algorithms. It also reveals that the geometry of the image is very simple compared to a shorter pulse image.

Corollary 3.1 (Closed-form limiting distribution)**.** The average limiting distribution for aperiodic rastering has a closed form given by

$$I_\infty(x,y) = \frac{1}{16A_xA_y}\left[\operatorname{erf}\left(\frac{x-c_x+A_x}{\sqrt{2}\sigma_x}\right) - \operatorname{erf}\left(\frac{x-c_x-A_x}{\sqrt{2}\sigma_x}\right)\right] \cdot \left[\operatorname{erf}\left(\frac{y-c_y+A_y}{\sqrt{2}\sigma_y}\right) - \operatorname{erf}\left(\frac{y-c_y-A_y}{\sqrt{2}\sigma_y}\right)\right].$$

Proposition 3.3 will be proved using tools from *ergodic theory*, in particular the well-known Birkhoff ergodic theorem, which we state below. For definitions of all terminology related to ergodicity that is used throughout this section, see Appendix B.

Theorem 3.1 (Birkhoff's ergodic theorem)**.** *If $T : X \to X$ is measure-preserving and ergodic on a probability space (X, Σ, μ), and $f \in L^1(\mu)$, then*

$$\lim_{n\to\infty} \frac{1}{n}\sum_{k=0}^{n-1} f\big(T^k x\big) = \int_X f\, d\mu \quad \textit{for } \mu\textit{-almost every } x.$$

We can immediately note the connection between this theorem and our problem at hand: the left hand side of the equation looks similar to our average image I_N and the right hand side is completely independent of time, indicating that those parameters that describe the particular evolution of the rastering over time (frequencies and phases) do not affect the result. In fact, we will see that proving 3.3 is almost entirely a matter of showing that Birkhoff's theorem applies to our situation and that all conditions are fulfilled. To do that, let us first make the following identifications:

- The state space X is the rectangle $[c_x - A_x,\, c_x + A_x] \times [c_y - A_y,\, c_y + A_y] \subset \mathbb{R}^2$.
- The map $T : X \to X$ corresponds to a discretization of the rastering equations with some arbitrary Δt. Writing $t_k = k\Delta t$, we have

$$T(r_x(t_k), r_y(t_k)) = (r_x(t_{k+1}),\, r_y(t_{k+1})).$$

- The observable $f : X \to \mathbb{R}$ is the beam intensity at a fixed point (x, y),

$$f(r_x, r_y) = \mathcal{B}\big(x - r_x, y - r_y\big).$$

  Note that x and y are always considered fixed in this section; we are considering what the average image looks like *pointwise*. This means that there are no "extra" variables compared to the identity in Birkhoff's theorem.

- The time average in Birkhoff's theorem becomes exactly our averaged image, for a fixed (x, y):

$$\frac{1}{N}\sum_{k=0}^{N-1} f\big(T^k(r_x^{(0)}, r_y^{(0)})\big) = I_N(x, y).$$

3.4.1 Proof of frequency and phase independence

After the identifications above, we are ready to begin the proof of Proposition 3.3. To make the proof structured, we split it into showing the following three statements:

(i) The dynamical system (X, Σ, μ, T) in our rastering problem fits the setting of the theorem: X is equipped with the Borel σ-algebra and normalized Lebesgue measure μ, and $f \in L^1(\mu)$.

(ii) The rastering map T is measure-preserving with respect to μ.

(iii) The map T is ergodic when $f_y/f_x \in \mathbb{R} \setminus \mathbb{Q}$.

Together, (i)-(iii) mean that Birkhoff's theorem applies.

Proof of Proposition 3.3.

(i) Setting.

We take $X = [-A_x + c_x, A_x + c_x] \times [-A_y + c_y, A_y + c_y] \subset \mathbb{R}^2$ with its Borel σ-algebra and normalized Lebesgue measure μ, so $\mu(X) = 1$. For each fixed point (x, y) we define $f(r_x, r_y) = \mathcal{B}(x - r_x, y - r_y)$. Since $\mathcal{B}$ is continuous on the compact set X, the observable f is bounded, and hence $f \in L^1(\mu)$. Thus the setting matches the one for Birkhoff's theorem.

(ii) Measure preservation.

Since μ is the normalized Lebesgue measure on X, we have

$$\mu(A) = \frac{1}{|X|}\int_A dr_x\, dr_y.$$

On each step, the rastering map T acts on (r_x, r_y) either as a translation or as a reflection at the boundary. Intuitively, Euclidean area should be preserved. This can formally be shown by a change of variables. By setting $(r_x, r_y) = T^{-1}(u, v)$, we obtain

$$\int_{T^{-1}(A)} dr_x\, dr_y = \int_A \left|\det J_{T^{-1}}(u, v)\right| du\, dv = \int_A du\, dv,$$

where $|\det J_{T^{-1}}| = 1$ for translations and reflections. Dividing by $|X|$ yields $\mu(T^{-1}(A)) = \mu(A)$ for all measurable $A \subset X$.

(iii) Ergodicity.

Here, the notion of ergodicity corresponds to there being no invariant sets for T other than ones of trivial measure, which physically means that the raster cannot get stuck moving across just a subarea of the rectangle. The core idea to prove this is to relate the triangle wave in one dimension to a rotation on the circle $[0, 1)$ with glued endpoints, and then the full raster motion in the rectangle to a rotation on the torus $[0, 1)^2$. These one- and two-dimensional maps are classical examples of ergodic dynamical systems, described in for example Sarig (2009).

Specifically, it can be shown that the raster map is a so-called *factor* of an irrational rotation on the circle. By standard results in ergodic theory (see Dajani and Kraaikamp (2009)), the raster map then inherits ergodicity from the rotation. The relation between the raster map T, the rotation R_α with $\alpha = f\Delta t$, and the factor map h can be written as

$$T \circ h = h \circ R_\alpha,$$

where h can be defined so that it maps each point on the circle (torus) to a corresponding point on the raster interval (rectangle). What remains is to define h explicitly and verify that the properties of a factor map are satisfied; however, for this thesis we do not go into that level of detail.

Now we are free to apply Birkhoff's theorem to our particular problem. This yields

$$I_\infty(x, y) = \lim_{N\to\infty} I_N(x, y) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \mathcal{B}\big(x - r_x(t_k), y - r_y(t_k)\big) = \int_X \mathcal{B}(x - r_x, y - r_y)\, d\mu(r_x, r_y).$$

Using the definition of the normalized Lebesgue measure and the corners of the rectangle X we obtain the final integral expression

$$I_\infty(x, y) = \frac{1}{4A_xA_y} \int_{c_x - A_x}^{c_x + A_x} \int_{c_y - A_y}^{c_y + A_y} \mathcal{B}\big(x - r_x, y - r_y\big)\, dr_y\, dr_x,$$

or with the explicit expression for $B\big(x - r_x, y - r_y\big)$:

$$I_\infty(x,y) = \frac{1}{4A_xA_y}\int_{c_x-A_x}^{c_x+A_x}\int_{c_y-A_y}^{c_y+A_y}\frac{1}{2\pi\sigma_x\sigma_y} \cdot \exp\left(-\frac{(x-r_x)^2}{2\sigma_x^2}\right)\exp\left(-\frac{(y-r_y)^2}{2\sigma_y^2}\right)\,dr_y\,dr_x. \tag{3.1}$$

It is now clear that the limiting distribution is independent of f_x, f_y as well as ϕ_x, ϕ_y, which is what we wanted to prove. □

3.4.2 Derivation of closed form

We now want to compute the integral in (3.1) to obtain the closed form stated in Proposition 3.1.

Proof of Corollary 3.1. The double integral in (3.1) can be separated into an x-dependent and a y-dependent factor, since the Gaussian kernel is multiplicative in the two coordinates. Hence,

$$I_\infty(x,y) = \frac{1}{4A_xA_y}\cdot\frac{1}{2\pi\sigma_x\sigma_y}\left(\int_{c_x-A_x}^{c_x+A_x}\exp\left[-\frac{(x-r_x)^2}{2\sigma_x^2}\right]\,dr_x\right)\left(\int_{c_y-A_y}^{c_y+A_y}\exp\left[-\frac{(y-r_y)^2}{2\sigma_y^2}\right]\,dr_y\right).$$

Each of the one-dimensional integrals can be expressed in terms of the error function:

$$\int_{c_x-A_x}^{c_x+A_x}\exp\left[-\frac{(x-r_x)^2}{2\sigma_x^2}\right]\,dr_x = \sqrt{\frac{\pi}{2}}\,\sigma_x\left[\operatorname{erf}\left(\frac{x-c_x+A_x}{\sqrt{2}\sigma_x}\right)-\operatorname{erf}\left(\frac{x-c_x-A_x}{\sqrt{2}\sigma_x}\right)\right],$$

and an identical expression holds in the y-direction. Substituting these back into the separated expression and simplifying the constants yields

$$\boxed{I_\infty(x,y) = \frac{1}{16A_xA_y}\left[\operatorname{erf}\left(\frac{x-c_x+A_x}{\sqrt{2}\sigma_x}\right)-\operatorname{erf}\left(\frac{x-c_x-A_x}{\sqrt{2}\sigma_x}\right)\right] \cdot\left[\operatorname{erf}\left(\frac{y-c_y+A_y}{\sqrt{2}\sigma_y}\right)-\operatorname{erf}\left(\frac{y-c_y-A_y}{\sqrt{2}\sigma_y}\right)\right].}$$

This completes the proof. □

We end this section by making an important practical note: the fact that the argument builds on using the time average does not limit the usefulness of the final formula for real long-pulse images that are not averaged. The factor N merely changes the absolute scale, which means that for a sufficiently long pulse we have

$$I_N \approx I_\infty \iff \frac{1}{N}\tilde{I}_{\text{obs}} \approx I_\infty \iff \tilde{I}_{\text{obs}} \approx N\,I_\infty.$$

3.4.3 Numerical verification and fourth limitation

Based on Figure 3.2 we draw the conclusion that a simple numerical experiment validates the result of Corollary 3.1.

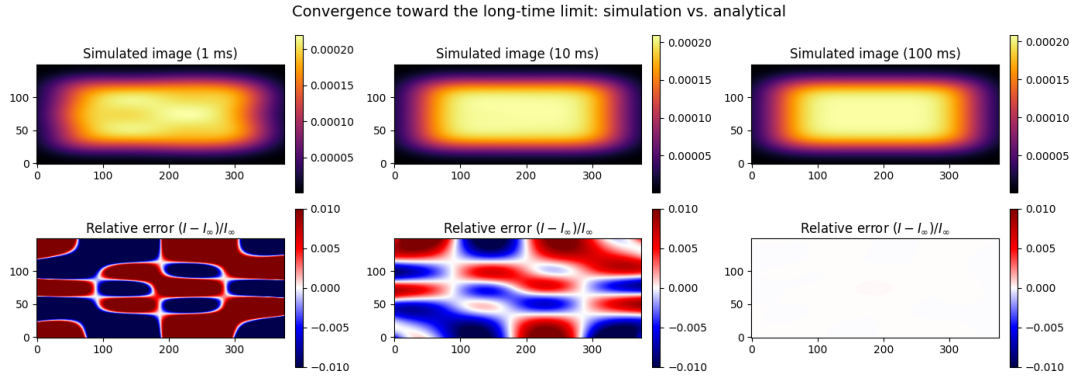

Figure 3.2: Convergence of the simulated rastered-beam image toward the analytical long-time limit $I_\infty(x, y)$. The upper row shows simulated images for pulse durations of 1, 10, and 100 ms. The lower row shows the relative error $(I - I_\infty)/I_\infty$, illustrating how it decreases as integration time increases.

Finally, we may put the result of Proposition 3.3 and the numerical verification as another fundamental limitation we have discovered:

> **Limitation 4.** There exists an upper pulse duration $T_{\text{upper}} > 0$ such that the forward model $\mathcal{F}$ is not invertible (in practice) for pulse durations $T > T_{\text{upper}}$.

In combination with Limitation 2, this means that we have established a necessary condition for invertibility, namely that

$$T \in [T_{\text{lower}}, T_{\text{upper}}].$$

3.5 Identifiability in practice

It is worth clarifying that there is a difference between coming up with highly specific examples of points $\mathbf{k}$ in parameter space where there are issues with identifiability, and whether this actually causes problems in practice when, for example, running a numerical minimization algorithm. It could be that "almost all" parameter vectors $\mathbf{k}$ (in a measure-theoretic sense) that generate an image $I_{\text{model}}(\mathbf{k})$ are identifiable. However, there does seem to be some cause for concern in practice: below we present an example of two images generated by vastly different $\mathbf{k}$ that are still very close in image norm. The wrong $\mathbf{k}$ was estimated using the minimization algorithm presented in detail in Chapter 4, which terminated after having come close to a zero gradient, as can be seen in the table.

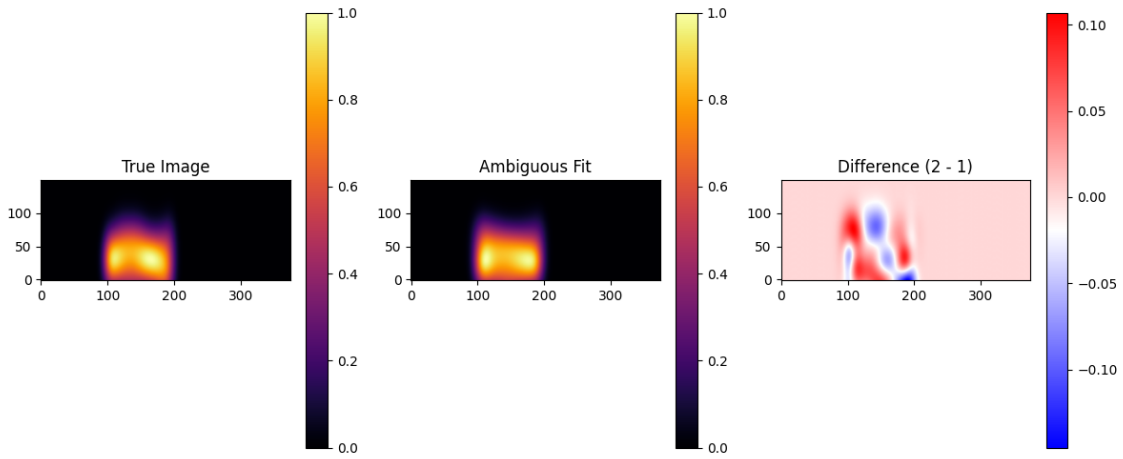

Figure 3.3: Comparison of observed image, simulated image from estimated parameters, and the pixel-wise differences between the images.

Minimization quantity	**Value**
Final loss (relative MSE)	8.04×10^{-3}
Gradient norm at solution	0.00×10^{0}
Parameter error norm	1.40×10^{1}

Table 3.1: Final results from BFGS optimization. The loss is small and the gradient norm is zero, indicating convergence to a stationary point. However the estimated parameters are far away from the true parameters.

Parameter	**True**	**Estimated**	**Error**	**Comment (error size)**
A_x	1.97×10^{1}	1.91×10^{1}	5.90×10^{-1}	Small
A_y	1.07×10^{1}	0.00×10^{0}	1.07×10^{1}	Large
σ_x	2.68×10^{0}	3.05×10^{0}	3.70×10^{-1}	Small
σ_y	1.07×10^{1}	1.22×10^{1}	1.54×10^{0}	Moderate
c_x	-1.61×10^{1}	-1.59×10^{1}	2.10×10^{-1}	Small
c_y	-1.76×10^{1}	-1.69×10^{1}	7.00×10^{-1}	Small
f_x	2.62×10^{1}	2.73×10^{1}	1.07×10^{0}	Moderate
f_y	4.15×10^{1}	3.28×10^{1}	8.73×10^{0}	Large
Total error norm			1.40×10^{1}	

Table 3.2: Comparison of true vs. estimated beam parameters for the stationary point that was found.

The example does not necessarily show that it is impossible to create a good algorithm that estimates the parameters accurately, but it does show that one has to be a bit careful. Concretely, when working with a minimization algorithm one has to be aware of how different termination criteria will affect the result; in this specific case, the point that was "almost" a minimizer could probably have been avoided with a stricter criterion on the loss function.

We end this section by stating the main conclusion clearly. Relating to section 3.3 it is worth noting that this is essentially a case of Limitation 3.

Conclusion. Practical issues with identifiability exist. In particular, there can be local minima $\hat{\mathbf{k}}$ such that $\mathcal{F}(\hat{\mathbf{k}}) \approx \mathcal{F}(\mathbf{k}^*)$, $\|\hat{\mathbf{k}} - \mathbf{k}^*\| \gg 0$..

Remark. The stationary point found above was verified to be a local minimum by evaluating the Hessian matrix numerically. All eigenvalues were positive, confirming local convexity of the loss function in this region. The smallest and largest eigenvalues were $\lambda_{\min} = 5.03 \times 10^{-4}$ and $\lambda_{\max} = 2.12 \times 10^{-2}$, respectively.

3.6 A formula for expected noise floor

3.6.1 Derivation of the formula

A central question for any optimization algorithm is when the process should stop, see for example Böiers (2010). Here it is useful to know something about how small values of the loss function we can expect given a certain $\tilde{I}_{\text{obs}}$. In this section we derive a formula for what we may call the *expected minimum loss*, that is both applicable to our particular beam problem and in a more general setting.

We start by introducing the vectorized forms of the images by defining

$$\mathbf{Y} := \text{vec}(\tilde{I}_{\text{obs}}) \in \mathbb{R}^N, \quad \mathbf{x} := \text{vec}(I_{\text{model}}(\mathbf{k})) \in \mathbb{R}^N, \quad N = m \cdot n,$$

where $\mathbf{x}$ is a known image for every $\mathbf{k}$ whereas $\mathbf{Y}$ is a random variable defined by (**??**). The relative squared loss in (2.7) can then be written as

$$\mathcal{L}(\mathbf{x}; \mathbf{Y}) := \frac{\|\mathbf{x} - \mathbf{Y}\|_2^2}{\|\mathbf{Y}\|_2^2} = \frac{\sum_{i=1}^N (x_i - Y_i)^2}{\sum_{i=1}^N Y_i^2}.$$

The exact minimum value of this loss is not known, due to the fact that $\mathbf{Y}$ is random; note that we are studying a *general* $\mathbf{Y}$ at this point, as opposed to a particular realization $\mathbf{y}$ with known values. However, we can estimate the *expected* minimum loss if we assume that this is the loss that you get from finding exactly the true underlying parameters $\mathbf{k}^*$ and comparing the resulting image $\mathbf{x} = \mathbf{x}(\mathbf{k}^*)$ to the noisy image $\mathbf{Y}$. For this we define

$$\mathcal{L}_{\min}(\mathbf{Y}) := \mathcal{L}(\mathbf{x}(\mathbf{k}^*); \mathbf{Y}).$$

To estimate the expected value $\mathbb{E}[\mathcal{L}_{\min}(\mathbf{Y})]$, we may use the second-order Taylor expansion for functions of random vectors (see for example Rego et al. (2021) for the Taylor approximation). This gives

$$\mathbb{E}[\mathcal{L}_{\min}(\mathbf{Y})] \approx \mathcal{L}_{\min}(\mathbb{E}[\mathbf{Y}]) + \frac{1}{2} \text{Tr}\left(H_{\mathcal{L}_{\min}}(\mathbb{E}[\mathbf{Y}]) \cdot \Sigma\right), \tag{3.2}$$

where the matrices that appear are ones that can be calculated analytically here. The first

matrix $H_{\mathcal{L}_{\min}} \in \mathbb{R}^{N\times N}$ is the Hessian of $\mathcal{L}_{\min}$ with respect to the components of $\mathbf{Y}$, defined by

$$[H_{\mathcal{L}_{\min}}]_{i,j} = \frac{\partial^2 \mathcal{L}_{\min}}{\partial Y_i \partial Y_j}.$$

Performing this differentiation for a general Hessian component and evaluating at $\mathbf{Y} = \mathbb{E}[\mathbf{Y}]$ yields, after simplification,

$$[H_{\mathcal{L}_{\min}}]_{i,j} = \frac{2}{\|\mathbb{E}[\mathbf{Y}]\|_2^2}\,\delta_{ij} \quad \Longrightarrow \quad H_{\mathcal{L}_{\min}} = \frac{2}{\|\mathbb{E}[\mathbf{Y}]\|_2^2}\,\mathrm{Id},$$

where δ_{ij} denotes the Kronecker delta and Id is the identity matrix. The second matrix $\Sigma \in \mathbb{R}^{N\times N}$ in the second-order term in (3.2) is the covariance matrix of $\mathbf{Y}$, given by

$$[\Sigma]_{i,j} = \mathrm{Cov}(Y_i, Y_j).$$

Under the independent Poisson noise model, all off-diagonal terms vanish and the diagonal equals the mean:

$$\Sigma = \mathrm{diag}(\mathbb{E}[Y_1], \ldots, \mathbb{E}[Y_N]) = \mathrm{diag}(\mathbb{E}[\mathbf{Y}]).$$

Using this simplification, together with $\mathcal{L}_{\min}(\mathbb{E}[\mathbf{Y}]) = 0$, and the explicit form of $H_{\mathcal{L}_{\min}}$, the approximation becomes

$$\begin{aligned}
\mathbb{E}[\mathcal{L}_{\min}] &\approx \frac{1}{2}\,\mathrm{Tr}\left(\frac{2}{\|\mathbb{E}[\mathbf{Y}]\|_2^2}\,\mathrm{Id}\cdot\mathrm{diag}(\mathbb{E}[\mathbf{Y}])\right) \\
&= \frac{1}{\|\mathbb{E}[\mathbf{Y}]\|_2^2}\,\mathrm{Tr}(\mathrm{diag}(\mathbb{E}[\mathbf{Y}])) \\
&= \frac{\sum_{i=1}^N \mathbb{E}[Y_i]}{\sum_{i=1}^N \mathbb{E}[Y_i]^2}.
\end{aligned}$$

Since the mean $\mathbb{E}[\mathbf{Y}] = \mathbf{x}(\mathbf{k}^*)$ is unknown during inference, we replace it by its observed realization $\mathbf{y}$ as a plug-in estimator for the mean intensity. In other words, we approximate $\mathbb{E}[\mathbf{Y}] \approx \mathbf{y}$. This yields the final noise–floor estimator

$$\boxed{\widehat{\mathbb{E}}[\mathcal{L}_{\min}] \;\approx\; \frac{\sum_{i=1}^N y_i}{\sum_{i=1}^N y_i^2}.} \tag{3.3}$$

3.6.2 Numerical verification for nominal parameters

To evaluate the accuracy of the approximation, we consider the long-pulse image generated from the nominal parameter vector $\mathbf{k}_{\mathrm{nom}}$. The corresponding noiseless model image is denoted $\mathbf{x}_{\mathrm{nom}}$, and an observed image $\mathbf{y}_{\mathrm{nom}}$ is obtained by applying Poisson noise to it. We then compare the

empirical loss $\mathcal{L}(\mathbf{x}_{\text{nom}}; \mathbf{y}_{\text{nom}})$ with the noise–floor estimator

$$\widehat{\mathbb{E}}[\mathcal{L}_{\min}] \;\approx\; \frac{\sum_{i=1}^{N} y_{\text{nom},i}}{\sum_{i=1}^{N} y_{\text{nom},i}^2}$$

For the Poisson realization used in this test, the values were very close: the empirical loss was 0.0286, while the approximation gave 0.0297, corresponding to a relative error of about 4%. This indicates that the expected minimum loss approximation can be used to get a rough lower bound on the achievable loss, and hence help decide when the optimization or estimation process should stop.

3.7 Conclusions

In this chapter we have investigated some key properties of the inverse problem of finding the underlying parameters for the rastered beam images. We have found that a solution always exists, though it is not necessarily unique. The problem with non-uniqueness (or weak identifiability) is present both in the idealized theoretical world and in practice in the simulations. In Section 3.4 we stated and proved a particularly important result related to non-uniqueness; the fact that after sufficiently long time neither frequency nor phase parameters can be distinguished, and that it is possible to give an explicit formula for the limit distribution. Finally, we derived an approximation of expected noise levels that may guide decisions related to when a solution is "sufficiently close" to the true one.

Chapter 4

Estimation with numerical optimization

Here we study how optimization can be used to solve ($\tilde{P}$) numerically. First, the theory behind the optimization method is described in detail; in particular we present a convergence theorem with proof in Section 4.1.3. In Section 4.2.1 and 4.2.2, we also derive the gradient of the loss function and investigate its convexity properties. Finally we test the method on simulated noisy images.

Notation. Throughout this chapter we use H to denote the Hessian of the loss function $\mathcal{L}(\mathbf{k})$. For the approximation of the Hessian we will use $\mathbf{B}_n$ and for the approximation of the inverse Hessian we will use $\mathbf{B}_n^{-1}$. Note that this is different from the reference Nocedal and Wright (2006) where $\mathbf{H}_n$ is used for the inverse Hessian approximation; here we avoid that due to the close resemblance between $\mathbf{H}_n$ and H.

4.1 Theory of optimization

4.1.1 Gradient-based optimization and Newton methods

When solving minimization problems such as the one encountered in this thesis, it is common to use numerical gradient-based methods. The general idea behind these methods is to begin from an initial guess $\mathbf{k}_0$ that is believed to be close to the solution, and then iteratively move towards a local minimum of the loss function. At each step, we compute the *gradient* of the loss, which indicates the direction of steepest increase. The *negative gradient* therefore points in the direction of steepest descent and forms the basis of the update direction. In more advanced methods, the *curvature* of the loss surface is also taken into account, either exactly (via the Hessian) or approximately, to adaptively refine the direction and step size. This process is repeated at each new point until a stopping criterion is satisfied, typically when the gradient norm becomes small, or when changes in the parameter vector or function value fall below a predefined threshold.

A natural refinement of basic gradient descent is *Newton's method*, which uses the Hessian explicitly. This is for example described in Böiers (2010). The method is derived from a second-order Taylor expansion of the loss function around the current point, which means that it is going to be a good approximation and have fast, quadratic convergence when the current point is sufficiently close to a local minimum. The updating rule can be written as

$$\mathbf{k}_{n+1} = \mathbf{k}_n - H(\mathbf{k}_n)^{-1} \nabla \mathcal{L}(\mathbf{k}_n).$$

An alternative to trying to calculate the Hessian explicitly is to make approximations of it (or its inverse), which is usually a lot less computationally expensive. This is, for example, done in *quasi-Newton methods*. Here, gradient information is used to estimate the Hessian with a matrix $\mathbf{B}_n \approx H(\mathbf{k}_n)$ that is symmetric and, if possible, positive definite. One of the most well-known quasi-Newton methods is *BFGS* (Broyden–Fletcher–Goldfarb–Shanno), where the Hessian approximation is updated using differences between successive points and gradients:

$$\mathbf{s}_n = \mathbf{k}_{n+1} - \mathbf{k}_n, \qquad \mathbf{y}_n = \nabla \mathcal{L}(\mathbf{k}_{n+1}) - \nabla \mathcal{L}(\mathbf{k}_n). \tag{4.1}$$

Based on these differences, the next Hessian approximation $\mathbf{B}_{n+1}$ is chosen such that the so-called secant condition $\mathbf{B}_{n+1}\mathbf{s}_n = \mathbf{y}_n$ is satisfied (Nocedal and Wright 2006).

Before finishing this general section on optimization it is worth defining the concept of *convexity* of a function, which plays a central role in the theory of convergence that is discussed throughout the chapter. We use the standard definition from Böiers (2010).

Definition 4.1 (Convex function)**.** A function $\mathcal{L} : \Omega \to \mathbb{R}$ is called *convex* if for any $\mathbf{k}_0, \mathbf{k}_1 \in \Omega$ and any $\lambda \in [0, 1]$,

$$\mathcal{L}((1-\lambda)\,\mathbf{k}_0 + \lambda\,\mathbf{k}_1) \;\leq\; (1-\lambda)\,\mathcal{L}(\mathbf{k}_0) + \lambda\,\mathcal{L}(\mathbf{k}_1). \tag{4.2}$$

We also want to introduce the notion of a *strongly convex* function. Here we use a definition from Fawzi (2023) that is based on the assumption that the function is twice continuously differentiable, since this will be the case for all the theory in Section 4.1.3.

Definition 4.2 (Strong convexity)**.** Let $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable on a convex set Ω. We say that $\mathcal{L}$ is *strongly convex* on Ω if its Hessian satisfies

$$\mathbf{h}^\top \nabla^2 \mathcal{L}(\mathbf{k})\,\mathbf{h} > 0 \qquad \text{for all } \mathbf{k} \in \Omega \text{ and all nonzero directions } \mathbf{h} \in \mathbb{R}^n.$$

4.1.2 A note on the choice of minimization algorithm

It is worth clarifying that there are many possible choices of optimization algorithm. Using BFGS can be motivated by the following reasons:

1. BFGS is well-studied in both theory and applications.

2. It avoids computation of the exact Hessian, which in our setting would be very costly due to the many time integrals in the derivatives.

3. The method already has a reliable and efficient implementation in `scipy`.

Apart from these reasons, quasi-Newton methods also have the benefit of being extremely general; they do not require any specific structure of the loss function in order to be applicable. However, it should be mentioned that for nonlinear least-squares problems there exist more specialized methods, such as Gauss–Newton or Levenberg–Marquardt. These can be considered in future work for this problem if the loss function $\mathcal{L}(\mathbf{k})$ in (2.2) continues to be used and if the regularization terms in $R(\mathbf{k})$ are quadratic.

4.1.3 The BFGS algorithm; convergence

Since the BFGS algorithm is going to be used throughout this chapter as it is applied to the beam problem, we will now present it in full detail based on Nocedal and Wright (2006). The central point that separates the algorithm from other gradient or Hessian based methods is the update formula for the inverse Hessian:

$$\mathbf{B}_{n+1}^{-1} = (I - \rho_n \mathbf{s}_n \mathbf{y}_n^\top)\, \mathbf{B}_n^{-1}\, (I - \rho_n \mathbf{y}_n \mathbf{s}_n^\top) + \rho_n \mathbf{s}_n \mathbf{s}_n^\top, \qquad \rho_n = \frac{1}{\mathbf{y}_n^\top \mathbf{s}_n}, \tag{4.3}$$

where $\mathbf{s}_n, \mathbf{y}_n$ are given by (4.1). This formula comes from requiring three things from the approximation $\mathbf{B}_{n+1}^{-1}$ of the true inverse Hessian:

1. it should satisfy the secant condition $\mathbf{B}_{n+1}^{-1}\mathbf{y}_n = \mathbf{s}_n$, that we saw in Section 4.1.1;

2. it should remain symmetric;

3. it should change as little as possible from the previous iterate.

These conditions combined define a constrained optimization problem of its own for $\mathbf{B}_{n+1}$:

$$\begin{aligned} \min_{\mathbf{C}} \quad & \|\mathbf{C} - \mathbf{B}_n^{-1}\|^2 \\ \text{s.t.} \quad & \mathbf{C}\,\mathbf{y}_n = \mathbf{s}_n, \\ & \mathbf{C} = \mathbf{C}^\top, \end{aligned}$$

where the unique solution is exactly the update rule (4.3). However, there also exists an update rule for just the Hessian approximation $\mathbf{B}_n$, which can be combined with an efficient method (Cholesky factorization) to get the inverse. This update rule is

$$\mathbf{B}_{n+1} = \mathbf{B}_n - \frac{\mathbf{B}_n \mathbf{s}_n \mathbf{s}_n^\top \mathbf{B}_n}{\mathbf{s}_n^\top \mathbf{B}_n \mathbf{s}_n} + \frac{\mathbf{y}_n \mathbf{y}_n^\top}{\mathbf{y}_n^\top \mathbf{s}_n}, \tag{4.4}$$

and will be the one that the convergence proof for the BFGS algorithm below will be based on. The equations (4.3) and (4.4) are in fact equivalent, which can be shown by the so-called Sherman–Morrison–Woodbury formula. Hence, using (4.3) in Algorithm 1 below and in the `scipy` implementation, but (4.4) in the proof, is valid.

Now we are ready to present the full BFGS algorithm. Note that $\mathbf{B}_n^{-1}$ is only notation, and does not correspond to the operation of "taking the inverse".

Algorithm 1 BFGS Method

Require: Starting point $\mathbf{k}_0$, tolerance $\varepsilon > 0$, initial inverse Hessian approximation $\mathbf{B}_0^{-1}$ (often I or a scaled I)

1: **while** $\|\nabla \mathcal{L}_n\| > \varepsilon$ **do**
2: **Search direction:** $\mathbf{p}_n = -\mathbf{B}_n^{-1} \nabla \mathcal{L}_n$
3: **Line search:** Choose step length α_n satisfying the Wolfe conditions
4: **Update point:** $\mathbf{k}_{n+1} = \mathbf{k}_n + \alpha_n \mathbf{p}_n$
5: **Compute quantities for inverse Hessian:**

$$\mathbf{s}_n = \mathbf{k}_{n+1} - \mathbf{k}_n, \qquad \mathbf{y}_n = \nabla \mathcal{L}_{n+1} - \nabla \mathcal{L}_n$$

6: **Update inverse Hessian:**

$$\mathbf{B}_{n+1}^{-1} = (I - \rho_n \mathbf{s}_n \mathbf{y}_n^\top)\, \mathbf{B}_n^{-1}\, (I - \rho_n \mathbf{y}_n \mathbf{s}_n^\top) + \rho_n \mathbf{s}_n \mathbf{s}_n^\top, \quad \text{where } \rho_n = \frac{1}{\mathbf{y}_n^\top \mathbf{s}_n}$$

7: **Update iteration index:** $n = n + 1$
8: **end while**

The so-called Wolfe conditions referred to in the algorithm can be stated as

$$\mathcal{L}(\mathbf{k}_n + \alpha_n \mathbf{p}_n) \leq \mathcal{L}(\mathbf{k}_n) + c_1 \alpha_n \nabla \mathcal{L}_n^\top \mathbf{p}_n, \tag{4.5}$$

$$\nabla \mathcal{L}(\mathbf{k}_n + \alpha_n \mathbf{p}_n)^\top \mathbf{p}_n \geq c_2 \nabla \mathcal{L}_n^\top \mathbf{p}_n, \tag{4.6}$$

where typical choices of the constants are $c_1 = 10^{-4}$ and $c_2 = 0.9$, with $0 < c_1 < c_2 < 1$. The first condition ensures that the step length α_n is not too large (it should not overshoot the minimum too much), while the second condition ensures that α_n is not too small (it should make the slope significantly less negative). The second condition is in fact what ensures that the BFGS formula (4.3) is well defined and preserves positive definiteness at each iteration, by keeping $1/\mathbf{y}_n^\top \mathbf{s}_n > 0$.

Now we want to prepare to state the theorem that says under which conditions the BFGS algorithm converges. Two of the fundamental conditions that are used in the formulation are presented below, given some initial point $\mathbf{k}_0$.

Assumption 4.1.

(i) The objective function $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable.

(ii) The level set

$$S = \{\, \mathbf{k} \in \Omega \mid \mathcal{L}(\mathbf{k}) \leq \mathcal{L}(\mathbf{k}_0) \,\}$$

is convex, and there exist positive constants m and M such that, for all $\mathbf{k} \in S$ and $\mathbf{h} \in \mathbb{R}^n$,

$$m \, \|\mathbf{h}\|^2 \;\leq\; \mathbf{h}^\top H(\mathbf{k}) \mathbf{h} \;\leq\; M \, \|\mathbf{h}\|^2,$$

where $H(\mathbf{k}) = \nabla^2 \mathcal{L}(\mathbf{k})$ denotes the Hessian of $\mathcal{L}$.

Note that these conditions at least imply that the objective function is strongly convex on the level set. Next we will present three lemmas that together make up the key ingredients for the proof of the convergence theorem. The first one simply states that the function $\mathcal{L}$ defined on S has a unique minimizer.

Lemma 4.1 (Existence and uniqueness of the minimizer on S)**.** *Given Assumption 4.1, the function $\mathcal{L}$ has a unique minimizer $\mathbf{k}^*$ on S.*

Proof of Lemma 4.1. Assumption 4.1(ii) states that for all $\mathbf{k} \in S$ and $\mathbf{h} \in \mathbb{R}^n$,

$$m\|\mathbf{h}\|^2 \;\leq\; \mathbf{h}^\top H(\mathbf{k})\, \mathbf{h} \;\leq\; M\|\mathbf{h}\|^2,$$

with $m > 0$. This implies that $H(\mathbf{k})$ is positive definite on S. Then it follows from Nocedal and Wright (2006, Thm. 2.4–2.5) that $\mathcal{L}$ has a unique minimizer $\mathbf{k}^*$ on S. □

For the second lemma we need to introduce the notion of an "average Hessian".

Definition 4.3 (Average Hessian)**.** For each iteration n, the *average Hessian* is defined as

$$\bar{\mathbf{G}}_n = \int_0^1 \nabla^2 \mathcal{L}\big(\mathbf{k}_n + \tau \alpha_n \mathbf{p}_n\big)\, d\tau,$$

i.e. the mean value of the Hessian along the step from $\mathbf{k}_n$ to $\mathbf{k}_{n+1} = \mathbf{k}_n + \alpha_n \mathbf{p}_n$.

The average Hessian matrix shows up in the bounds that are established in the lemma below. These bounds will in turn be needed in the convergence proof to show that the inverse Hessian stays "well-behaved" for all iterations $n \geq 0$ in the algorithm.

Lemma 4.2 (Bounds in Hessian update)**.** *Let $\mathbf{s}_n = \mathbf{k}_{n+1} - \mathbf{k}_n$ and $\mathbf{y}_n = \nabla\mathcal{L}(\mathbf{k}_{n+1}) - \nabla\mathcal{L}(\mathbf{k}_n)$, as in Algorithm 1. Then the quantities satisfy the secant condition exactly:*

$$\bar{\mathbf{G}}_n \mathbf{s}_n = \mathbf{y}_n,$$

where $\bar{\mathbf{G}}_n$ is the average Hessian from Definition 4.3.

Further, under Assumption 4.1, the following bounds hold:

$$\frac{\mathbf{y}_n^\top \mathbf{s}_n}{\mathbf{s}_n^\top \mathbf{s}_n} = \frac{\mathbf{s}_n^\top \bar{\mathbf{G}}_n \mathbf{s}_n}{\mathbf{s}_n^\top \mathbf{s}_n} \;\geq\; m, \tag{4.7}$$

$$\frac{\mathbf{y}_n^\top \mathbf{y}_n}{\mathbf{y}_n^\top \mathbf{s}_n} = \frac{\mathbf{s}_n^\top \bar{\mathbf{G}}_n^2 \mathbf{s}_n}{\mathbf{s}_n^\top \bar{\mathbf{G}}_n \mathbf{s}_n} \;=\; \frac{\mathbf{h}_n^\top \bar{\mathbf{G}}_n \mathbf{h}_n}{\mathbf{h}_n^\top \mathbf{h}_n} \;\leq\; M, \qquad \text{where } \mathbf{h}_n := \bar{\mathbf{G}}_n^{1/2} \mathbf{s}_n. \tag{4.8}$$

Proof of Lemma 4.2. This is shown in Nocedal and Wright (2006, Inequalities 6.40–6.41). Note that the square root of a matrix $\bar{\mathbf{G}}_n^{1/2}$ can be defined via diagonalization. □

Finally, we state the following lemma, which provides the last ingredient needed for the convergence proof. It relates the limiting behavior of the quantity

$$\cos\theta_n = \frac{\nabla \mathcal{L}_n^\top \mathbf{B}_n^{-1} \nabla \mathcal{L}_n}{\|\nabla \mathcal{L}_n\| \, \|\mathbf{B}_n^{-1} \nabla \mathcal{L}_n\|}, \tag{4.9}$$

to the limiting behavior of the gradient norm $\|\nabla \mathcal{L}_n\|$. Here, θ_n may be interpreted as the "angle" between the negative gradient direction and the BFGS search direction. This interpretation follows directly from the Cauchy–Schwarz inequality, see Appendix C.1.

Lemma 4.3 (Zoutendijk condition for BFGS)**.** *Under Assumption 4.1, let* $\{\mathbf{k}_n\}$ *be the iterates generated by Algorithm 1, i.e.*

$$\mathbf{k}_{n+1} = \mathbf{k}_n - \alpha_n \mathbf{B}_n^{-1} \nabla \mathcal{L}_n,$$

where each step length α_n *satisfies the Wolfe conditions. Then the sequence* $\{\nabla \mathcal{L}_n\}$ *satisfies*

$$\sum_{n=0}^{\infty} (\cos\theta_n)^2 \, \|\nabla \mathcal{L}_n\|^2 < \infty.$$

Proof of Lemma 4.3. For the general result, see Nocedal and Wright (2006, Thm. 3.2) and the proof that follows. Our particular version for BFGS follows from setting the descent direction $\mathbf{p}_n = -\alpha_n \mathbf{B}_n^{-1} \nabla \mathcal{L}_n$ in their notation. □

An immediate consequence of the lemma, which is used in the proof of the convergence theorem for BFGS below, is that if θ_n is bounded away from 90°, then

$$\lim_{n \to \infty} \|\nabla \mathcal{L}_n\| = 0.$$

We are now ready to establish the theorem. We also describe the main steps of the proof below.

Theorem 4.1 (Convergence of the BFGS method)**.** *Let* $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ *and* $\mathbf{k}_0 \in \mathbb{R}^n$ *be such that Assumptions 4.1 hold. Initialize the BFGS algorithm with the starting point* $\mathbf{k}_0$ *and any symmetric positive definite matrix* $\mathbf{B}_0^{-1}$*. Then the sequence* $\{\mathbf{k}_n\}$ *generated by Algorithm 1 converges to the unique minimizer* $\mathbf{k}^*$ *of* $\mathcal{L}$ *on the level set* S*.*

Proof sketch.

Goal. We want to show that

$$\liminf_{n\to\infty} \|\nabla \mathcal{L}_n\| = 0.$$

Under strong convexity (see Definition 4.2) on the level set S this implies $\mathbf{k}_n \to \mathbf{k}^*$. For a proof of this auxiliary result, which is *not* included in Nocedal and Wright, see Appendix C.2.

Idea. Show that the angle between the steepest–descent direction $-\nabla \mathcal{L}_n$ and the BFGS search direction $-\mathbf{B}_n^{-1}\nabla \mathcal{L}_n$ stays bounded away from 90°. Intuitively, that means the algorithm produces search directions "close enough" to steepest descent. By Zoutendijk's lemma this forces $\liminf_{n\to\infty} \|\nabla \mathcal{L}_n\| = 0$, and in turn $\mathbf{k}_n \to \mathbf{k}^*$ by strong convexity.

Notation. For the matrix analysis we use the B-form of BFGS defined by the update rule (4.4). Further we define the quantities

$$\cos\phi_n := \frac{\mathbf{s}_n^\top \mathbf{B}_n \mathbf{s}_n}{\|\mathbf{s}_n\|\,\|\mathbf{B}_n \mathbf{s}_n\|}, \qquad q_n := \frac{\mathbf{s}_n^\top \mathbf{B}_n \mathbf{s}_n}{\mathbf{s}_n^\top \mathbf{s}_n}. \tag{4.10}$$

Recall that ϕ_n can be interpreted as the angle between $\mathbf{s}_n$ and $\mathbf{B}_n\mathbf{s}_n$. Note: from substituting $\mathbf{s}_n = \mathbf{k}_{n+1} - \mathbf{k}_n = -\alpha_n \mathbf{B}_n^{-1}\nabla \mathcal{L}_n$. and simplifying, it follows that $\cos\phi_n = \cos\theta_n$ in (4.9), which means that controlling the angle between $\mathbf{s}_n$ and $\mathbf{B}_n\mathbf{s}_n$ is equivalent to controlling the angle between $-\nabla \mathcal{L}_n$ and $-\mathbf{B}_n^{-1}\nabla \mathcal{L}_n$. It also means that Lemma 4.3 can be applied, which is crucial in step 5 below.

Steps.

1) *Trace and determinant bounds.* Taking trace and determinant of the BFGS update yields

$$\operatorname{trace}(\mathbf{B}_{n+1}) = \operatorname{trace}(\mathbf{B}_n) - \frac{\|\mathbf{B}_n\mathbf{s}_n\|^2}{\mathbf{s}_n^\top \mathbf{B}_n \mathbf{s}_n} + \frac{\|\mathbf{y}_n\|^2}{\mathbf{y}_n^\top \mathbf{s}_n}, \qquad \det(\mathbf{B}_{n+1}) = \det(\mathbf{B}_n)\,\frac{\mathbf{y}_n^\top \mathbf{s}_n}{\mathbf{s}_n^\top \mathbf{B}_n \mathbf{s}_n}.$$

2) *Angle interpretation and simplifications.* Using the definitions in (4.10) yields

$$\frac{\|\mathbf{B}_n\mathbf{s}_n\|^2}{\mathbf{s}_n^\top \mathbf{B}_n \mathbf{s}_n} = \frac{q_n}{\cos^2\phi_n}, \qquad \det(\mathbf{B}_{n+1}) = \det(\mathbf{B}_n)\,\frac{m_n}{q_n},$$

with

$$m_n := \frac{\mathbf{y}_n^\top \mathbf{s}_n}{\mathbf{s}_n^\top \mathbf{s}_n}, \qquad M_n := \frac{\mathbf{y}_n^\top \mathbf{y}_n}{\mathbf{y}_n^\top \mathbf{s}_n}.$$

Further, by using that $\mathbf{y}_n = \bar{\mathbf{G}}_n \mathbf{s}_n$, it follows from Lemma 4.2 that for each iteration n we have $m_n \geq m,\ M_n \leq M$.

3) *Potential function.* Introduce

$$\psi(\mathbf{B}) := \operatorname{trace}(\mathbf{B}) - \ln\det(\mathbf{B}),$$

defined for positive definite matrices $\mathbf{B}$. This is a positive scalar function that penalizes eigenvalues that are either very large or very small (in the sense that for such eigenvalues, the value of ψ becomes large).

4) *One-step bound for ψ.* Substituting expressions from steps 1–2 and simplifying gives

$$\psi(\mathbf{B}_{n+1}) = \psi(\mathbf{B}_n) + (M_n - \ln m_n - 1) + \left(1 - \tfrac{q_n}{\cos^2\phi_n} + \ln \tfrac{q_n}{\cos^2\phi_n}\right) + \ln\cos^2\phi_n.$$

Since $h(t) := 1 - t + \ln t \leq 0$ for $t > 0$ and $M_n - \ln m_n - 1 \leq c := M - \ln m - 1$, we can bound the potential function by

$$0 < \psi(\mathbf{B}_{n+1}) \leq \psi(\mathbf{B}_n) + c + \ln\cos^2\phi_n.$$

Applying the same bound at each iteration $j = 0, \ldots, n$ yields

$$0 < \psi(\mathbf{B}_{n+1}) \leq \psi(\mathbf{B}_0) + c(n+1) + \sum_{j=0}^{n} \ln\cos^2\phi_j. \tag{4.11}$$

5) *Angle bounded away from 90°; Zoutendijk ⇒ small gradients.* Now consider the sequence $\{\cos\phi_n\}_{n=0}^{\infty}$, as defined in (4.10). Assume, for contradiction, that $\cos\phi_n \to 0$ as $n \to \infty$. One can then show (see Nocedal and Wright for details) that the sum in (4.11) dominates the linear term for sufficiently large n, making the right-hand side negative. This contradicts the fact that $\psi(\mathbf{B}_{n+1}) > 0$.

The fact that $\cos\phi_n \nrightarrow 0$ means that there exists a constant $\delta > 0$ and a subsequence of indices $\{n_\ell\}_{\ell=1}^{\infty}$ such that $\cos\phi_{n_\ell} \geq \delta$ for all ℓ. By Zoutendijk's lemma, this in turn implies that $\|\nabla\mathcal{L}_{n_\ell}\| \to 0$, or equivalently that

$$\liminf_{n\to\infty} \|\nabla\mathcal{L}_n\| = 0.$$

Finally, by strong convexity it follows that $\mathbf{k}_n \to \mathbf{k}^*$. □

4.2 Analysis of the loss function

4.2.1 Deriving the gradient

The purpose of this subsection is to obtain the analytical expression for the gradient, ensuring that it can be safely used in the BFGS or any similar gradient-based algorithm.

As established earlier, the loss function can be written as

$$\mathcal{L}(\mathbf{k}) = \iint_{\Omega} \left[\tilde{I}_{\text{obs}}(x,y) - \int_0^T \mathcal{B}(x,y,t;\mathbf{k})\,dt\right]^2 dx\,dy.$$

The gradient takes the form

$$\nabla_{\mathbf{k}}\mathcal{L}(\mathbf{k}) = -2\iint_{\Omega}\left[\tilde{I}_{\text{obs}}(x,y) - \int_0^T \mathcal{B}(x,y,t;\mathbf{k})\,dt\right]\left(\int_0^T \nabla_{\mathbf{k}}\mathcal{B}(x,y,t;\mathbf{k})\,dt\right)dx\,dy,$$

where

$$\nabla_{\mathbf{k}}\mathcal{B}(x,y,t;\mathbf{k}) = \left[\frac{\partial\mathcal{B}}{\partial A_x}, \frac{\partial\mathcal{B}}{\partial A_y}, \frac{\partial\mathcal{B}}{\partial\sigma_x}, \frac{\partial\mathcal{B}}{\partial\sigma_y}, \frac{\partial\mathcal{B}}{\partial c_x}, \frac{\partial\mathcal{B}}{\partial c_y}, \frac{\partial\mathcal{B}}{\partial f_x}, \frac{\partial\mathcal{B}}{\partial f_y}, \frac{\partial\mathcal{B}}{\partial\phi_x}, \frac{\partial\mathcal{B}}{\partial\phi_y}\right]^{\top}.$$

For convenience we introduce the residual factor

$$\mathcal{L}_{\text{res}}(x,y) = -2\left[\tilde{I}_{\text{obs}}(x,y) - \int_0^T \mathcal{B}(x,y,t;\mathbf{k})\,dt\right].$$

Beam–shape parameters

Differentiating with respect to the Gaussian width parameters gives

$$\frac{\partial\mathcal{L}}{\partial\sigma_x} = \iint_{\Omega}\mathcal{L}_{\text{res}}(x,y)\left[\int_0^T \mathcal{B}(x,y,t;\mathbf{k})\cdot\frac{(x-r_x(t))^2-\sigma_x^2}{\sigma_x^3}\,dt\right]dx\,dy, \tag{4.12}$$

and similarly

$$\frac{\partial\mathcal{L}}{\partial\sigma_y} = \iint_{\Omega}\mathcal{L}_{\text{res}}(x,y)\left[\int_0^T \mathcal{B}(x,y,t;\mathbf{k})\cdot\frac{(y-r_y(t))^2-\sigma_y^2}{\sigma_y^3}\,dt\right]dx\,dy. \tag{4.13}$$

Raster–trajectory parameters

Let

$$\mathbf{k}_r = \{A_x, A_y, c_x, c_y, f_x, f_y, \phi_x, \phi_y\}$$

denote the set of parameters that are present in the raster equations

$$r_x(t) = A_x\text{tri}(2\pi f_x t + \phi_x) + c_x, \qquad r_y(t) = A_y\text{tri}(2\pi f_y t + \phi_y) + c_y.$$

Using the chain rule, the partial derivative of $\mathcal{B}$ with respect to any $k_r \in \mathbf{k}_r$ is

$$\frac{\partial\mathcal{B}}{\partial k_r} = \mathcal{B}(x,y,t;\mathbf{k})\left[\frac{x-r_x(t)}{\sigma_x^2}\frac{\partial r_x}{\partial k_r} + \frac{y-r_y(t)}{\sigma_y^2}\frac{\partial r_y}{\partial k_r}\right].$$

Further, we can compute each specific raster derivative $\frac{\partial r_x}{\partial k_r}$ and $\frac{\partial r_y}{\partial k_r}$:

$$\begin{aligned}
\frac{\partial r_x}{\partial A_x} &= \operatorname{tri}(2\pi f_x t + \phi_x), & \frac{\partial r_y}{\partial A_y} &= \operatorname{tri}(2\pi f_y t + \phi_y),\\
\frac{\partial r_x}{\partial c_x} &= 1, & \frac{\partial r_y}{\partial c_y} &= 1,\\
\frac{\partial r_x}{\partial f_x} &= 2\pi t\, A_x \operatorname{tri}'(2\pi f_x t + \phi_x), & \frac{\partial r_y}{\partial f_y} &= 2\pi t\, A_y \operatorname{tri}'(2\pi f_y t + \phi_y),\\
\frac{\partial r_x}{\partial \phi_x} &= A_x \operatorname{tri}'(2\pi f_x t + \phi_x), & \frac{\partial r_y}{\partial \phi_y} &= A_y \operatorname{tri}'(2\pi f_y t + \phi_y).
\end{aligned} \tag{4.14}$$

The general expression for the partial derivatives of the loss then becomes

$$\frac{\partial \mathcal{L}}{\partial k_r} = \iint_\Omega \mathcal{L}_{\text{res}}(x,y) \left[\int_0^T \mathcal{B}(x,y,t;\mathbf{k}) \left(\frac{x - r_x(t)}{\sigma_x^2} \frac{\partial r_x}{\partial k_r} + \frac{y - r_y(t)}{\sigma_y^2} \frac{\partial r_y}{\partial k_r} \right) dt \right] dx\, dy, \tag{4.15}$$

where the parameter–specific raster derivatives may be substituted from (4.14). Note that the derivative $\operatorname{tri}'(u)$ is not defined at the points $u = \frac{\pi}{2} + \pi n$, $n \in \mathbb{Z}$. However, it is defined *almost everywhere* (in the technical sense), which means that in practice there is no issue in computing it using automatic differentiation; see Section 4.3.2.

Final gradient and stationarity condition

Any minimizer $\mathbf{k}^\star$ must satisfy the first-order optimality condition

$$\nabla_{\mathbf{k}} \mathcal{L}(\mathbf{k}^\star) = 0.$$

Substituting the partial derivatives (4.12), (4.13) and (4.15) into that equation yields the final result:

$$\begin{aligned}
\iint_\Omega \mathcal{L}_{\text{res}}(x,y) \left[\int_0^T \mathcal{B}(x,y,t;\mathbf{k}^*) \frac{(x - r_x(t))^2 - \sigma_x^2}{\sigma_x^3}\, dt \right] dx\, dy = 0,\\
\iint_\Omega \mathcal{L}_{\text{res}}(x,y) \left[\int_0^T \mathcal{B}(x,y,t;\mathbf{k}^*) \frac{(y - r_y(t))^2 - \sigma_y^2}{\sigma_y^3}\, dt \right] dx\, dy = 0,\\
\iint_\Omega \mathcal{L}_{\text{res}}(x,y) \left[\int_0^T \mathcal{B}(x,y,t;\mathbf{k}^*) \left(\frac{x - r_x(t)}{\sigma_x^2} \frac{\partial r_x}{\partial k_r} + \frac{y - r_y(t)}{\sigma_y^2} \frac{\partial r_y}{\partial k_r} \right) dt \right] dx\, dy = 0.
\end{aligned} \tag{4.16}$$

Solving this system analytically is very difficult. The equations are highly nonlinear, have multiple factors and consist of time integrals inside a space integral. Furthermore, the system fully coupled. Hence, we draw the conclusion that we are going to have to rely on numerical methods to find candidate solutions for $(\tilde{P})$.

Remark. It is worth observing that with noisy data $\tilde{I}_{\text{obs}}$, the residual $\mathcal{L}_{\text{res}}$ is nonzero even at the true parameters, so no trivial stationary point exists. In other words, the stationary equations can only be fulfilled due to positive and negative parts canceling out over the space integral.

4.2.2 Convexity properties of the loss function

The next important property about the loss function that we want to investigate is convexity, since it plays a central role in convergence of the BFGS algorithm as we saw in 4.1.3. If we do *not* have global convexity, that means that we cannot guarantee that a numerical method converges in general, and even if it converges we cannot be sure that the stationary point that was found is a global minimum.

We will now show that the loss function defined by (2.2) is indeed not convex when the observed image is generated by the nominal parameters, $\tilde{I}_{\text{obs}} = I_{\text{model}}(\mathbf{k}_{\text{nom}})$ for a short pulse $T = 50\,\mu$s. (Note that we are limiting ourselves to this specific case for now. Also: for this section we use fixed $\phi_x = \phi_y = 0$. This does not change any of the conclusions.) By Definition 4.1, it suffices to find two parameter vectors $\mathbf{k}_1$ and $\mathbf{k}_2$ such that the inequality 4.2 is violated for some $\lambda \in (0, 1)$. Precisely such an example is shown in Figure 4.1. The loss is evaluated along the line between some points $\mathbf{k}_1$ and $\mathbf{k}_2$ in parameter space. In the figure, the blue curve shows the loss $\mathcal{L}(\mathbf{k}_\lambda)$ as a function of $\lambda \in [0, 1]$, where $\mathbf{k}_\lambda = (1 - \lambda)\mathbf{k}_0 + \lambda\mathbf{k}_1$. The red line that connects two points on the loss curve does not stay above the graph everywhere, violating convexity.

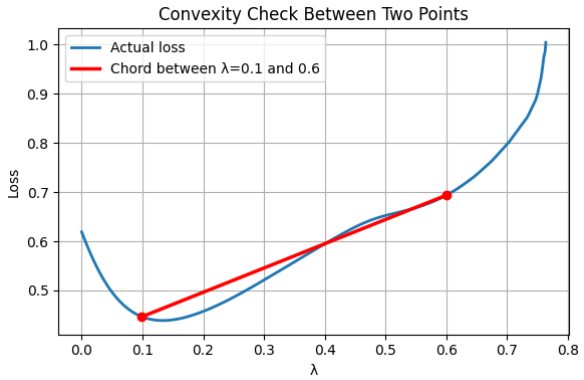

Figure 4.1: The loss function evaluated along a straight line in parameter space between $\mathbf{k}_1 = \mathbf{k}^* + \mathbf{k}_{\text{pert},1}$ and $\mathbf{k}_2 = \mathbf{k}^* + \mathbf{k}_{\text{pert},2}$. The reference parameters are $\mathbf{k}^* = [60.0, 20.0, 13.5, 5.05, 0.0, 0.0, 39.55, 29.05]$, and the perturbation vectors are $\mathbf{k}_{\text{pert},1} = [1.0, -1.0, 5.0, -3.0, 2.0, -1.5, 1.0, -3.0]$ and $\mathbf{k}_{\text{pert},2} = [-4.0, 5.0, -10.0, 7.0, -3.0, 4.0, -10.0, 10.0]$.

Remark. Another way to show non-convexity is to find a point in parameter space where the Hessian is not positive semi-definite. For twice continuously differentiable functions, this implies local non-convexity, which of course implies global non-convexity.

While we do not have global convexity of the loss function for $\tilde{I}_{\text{obs}} = I_{\text{model}}(\mathbf{k}_{\text{nom}})$, we do however have local convexity in a neighborhood of $\mathbf{k}_{\text{nom}}$. As we saw from Theorem 4.1, this is part of the sufficient conditions for convergence of the BFGS algorithm. The fact that the

function is locally convex is possible to verify numerically by computing the eigenvalues of the Hessian matrix. If all eigenvalues are positive, the Hessian is positive definite and in turn the loss function convex. Below we show the eight eigenvalues, and observe that all of them are larger than zero.

Index	Eigenvalue
8	0.46
7	0.15
6	0.018
5	0.012
4	0.0077
3	0.0013
2	0.0010
1	0.00056

Table 4.1: Eigenvalues of the Hessian at the nominal point $\mathbf{k}_{\text{nom}}$, rounded to two significant digits. All values are positive, indicating local convexity in this region.

To visualize the convexity we also show the loss function along the different eigenvectors of the Hessian. We see that the loss is convex along each of these directions (sufficiently close to the minimum). This is because along an eigenvector $\mathbf{q}_i$, the loss reduces to a one-dimensional function

$$\mathcal{L}(\mathbf{k}^* + \varepsilon \mathbf{q}_i) \;\approx\; \mathcal{L}(\mathbf{k}^*) + \tfrac{1}{2}(\varepsilon \mathbf{q}_i)^\top H(\mathbf{k}^*)(\varepsilon \mathbf{q}_i) \;=\; \tfrac{1}{2}\lambda_i \varepsilon^2,$$

which is a parabola in ε with positive second derivative for $\lambda_i > 0$.

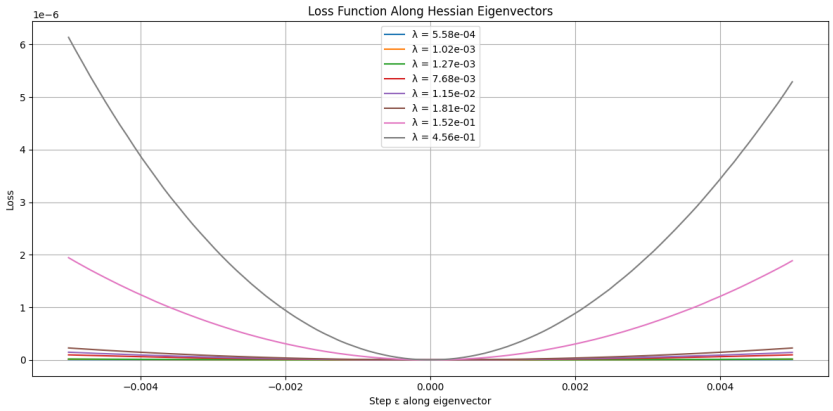

Figure 4.2: The loss function is evaluated along each of the 8 eigenvectors of the Hessian matrix at the minimum. The horizontal axis corresponds to a perturbation ε along each eigenvector direction, and the vertical axis shows the resulting loss. Each curve corresponds to a different eigenvector with corresponding eigenvalue λ.

4.2.3 Satisfying the BFGS assumptions

The convergence theorem technically requires a bit more than local strong convexity. From Assumption 4.1, the conditions can be listed as:

(a) $\mathcal{L} \in C^2$;
(b) the level set $S = \{\mathbf{k} \in \Omega : \mathcal{L}(\mathbf{k}) \leq \mathcal{L}(\mathbf{k}_0)\}$ is convex;
(c) the quadratic form is lower bounded, $\mathbf{h}^\top H(\mathbf{k})\mathbf{h} \geq m\|\mathbf{h}\|^2$ on S;
(d) an upper bound $\mathbf{h}^\top H(\mathbf{k})\mathbf{h} \leq M\|\mathbf{h}\|^2$ also holds.

In our setting these conditions hold locally:

(a) $\mathcal{L}$ is C^2 almost everywhere (see Section 4.2.1);
(b) choosing S as a small ball around the minimizer $\mathbf{k}^*$ makes it convex;
(c) the Hessian eigenvalues in Table 4.1 are strictly positive, providing a lower bound $m > 0$;
(d) the curvature is finite in this region, giving an upper bound M.

We state this as a conclusion below.

Conclusion. If the initial point $\mathbf{k}_0$ is chosen sufficiently close to the minimizer $\mathbf{k}^*$, then Theorem 4.1 applies and the BFGS converges.

4.3 Implementation

4.3.1 General notes

The numerical implementation builds directly on the image simulator introduced in Section 2.3, which is the computational backbone for evaluating the loss function throughout the optimization. The core parts that were added to this code are the following:

- **Gradients.** All gradients are computed using JAX automatic differentiation (see Section 4.3.2), avoiding finite-difference approximations.
- **Optimizer.** The BFGS algorithm is implemented using `scipy`'s `L-BFGS-B` routine. The L here stands for limited memory, while B stands for bounded domain.
- **Objective function.** The quantity that is minimized is the loss function defined in Section 2.3.
- **Convergence criterion.** Optimization stops when the relative reduction in the objective falls below machine precision or when the maximum number of iterations (typically set to 100) is reached.
- **Initialization.** The starting point $\mathbf{k_0}$ is supplied to the optimizer.
- **Parameter bounds.** The optimizer is constrained to remain within the allowed bounds defined by Ω..

4.3.2 Automatic differentiation

Automatic differentiation is a method to compute exact partial derivatives (up to machine-precision) when working with a numerical method that requires gradients. The idea is to first process the entire composite function, using the chain rule and known elementary function derivatives, to create a computational graph of the full derivative. Only once the entire graph is built, the intermediate derivatives are evaluated. This procedure guarantees that the numerical errors are extremely small, because each intermediate value is usually correct up to machine precision and is used in the evaluation of the next derivative all the way up to the outermost one.

We illustrate the method with a simple example inspired by Griewank and Walther (2008). Assume we have a scalar-valued function defined as a composition of elementary operations:

$$f(x) = \sin(x) \cdot (x - \exp(x)) \,.$$

We want to evaluate the derivative of this function at the point $x = x_0 = 1.5$. As a first step, we identify all intermediate variables that are going to be used during the procedure:

$$v_1 = \sin(x), \qquad v_2 = \exp(x), \qquad v_3 = x - v_2, \qquad y = v_1 \cdot v_3.$$

By interpreting y in this way it can be processed as a composite function of two variables, $y = y(v_1(x), v_3(x))$, where v_1 and v_3 are in turn functions of x. That means that the chain rule can be applied easily, with the exact expression

$$\frac{dy}{dx} = \frac{\partial y}{\partial v_1} \cdot \frac{\partial v_1}{\partial x} + \frac{\partial y}{\partial v_3} \cdot \frac{\partial v_3}{\partial x}. \tag{4.17}$$

The algorithm proceeds by first evaluating the intermediate values forward through the computational graph:

$$x_0 \longrightarrow v_1 = \sin(x_0) \longrightarrow v_2 = \exp(x_0) \longrightarrow v_3 = x_0 - v_2 \longrightarrow y = v_1 \cdot v_3,$$

and then computing the required partial derivatives along this chain. In *reverse mode*, the derivatives are accumulated backward from the output y. Here, that would mean:

1. Evaluate $\frac{\partial y}{\partial v_1} = v_3$ and $\frac{\partial y}{\partial v_3} = v_1$.
2. Evaluate $\frac{\partial v_1}{\partial x} = \cos(x)$ and $\frac{\partial v_3}{\partial x} = 1 - \exp(x)$.
3. Starting from the input $x_0 = 1.5$, evaluate the partial derivatives backwards in (4.17) to obtain the final value of $\frac{dy}{dx}$.

Now let us perform these steps for our specific example. Here we have

$$\begin{aligned} v_1 &= \sin(1.5), & v_2 &= \exp(1.5), & v_3 &= 1.5 - v_2, \\ \frac{\partial v_1}{\partial x}(1.5) &= \cos(1.5), & \frac{\partial v_3}{\partial x}(1.5) &= 1 - \exp(1.5), \\ \frac{\partial y}{\partial v_1}(1.5) &= v_3 = 1.5 - \exp(1.5), & \frac{\partial y}{\partial v_3}(1.5) &= v_1 = \sin(1.5). \end{aligned}$$

Substituting these values into Equation (4.17) gives

$$\frac{dy}{dx}(1.5) = \frac{\partial y}{\partial v_1}(1.5)\,\frac{\partial v_1}{\partial x}(1.5) + \frac{\partial y}{\partial v_3}(1.5)\,\frac{\partial v_3}{\partial x}(1.5).$$

Finally we can evaluate this numerically to get

$$\frac{dy}{dx}(1.5) = (1.5 - e^{1.5})\cos(1.5) + \sin(1.5)\big(1 - e^{1.5}\big) \approx -3.68388.$$

4.4 Experiments on images from nominal parameters

In this section we present the estimation results when using the BFGS method to minimize the squared difference between the observed image and the image produced by the current parameters $\mathbf{k}$. As convergence criterion we use relative reduction of the loss function being smaller than machine epsilon to make sure it is possible to get full numerical convergence.

4.4.1 Minimization for noiseless image

Figures 4.3 and 4.4 illustrate that the noiseless minimization problem (P) is well-behaved when the initial guess is reasonably close to the global minimum (here chosen as a small perturbation added to the true parameters). In the case of a full accumulation pulse, the raster frequencies f_x, f_y and phases ϕ_x, ϕ_y are held fixed, and the remaining six beam parameters can be accurately recovered. In the short 50 μs pulse case, only phases are held fixed, and the optimizer still converges successfully.

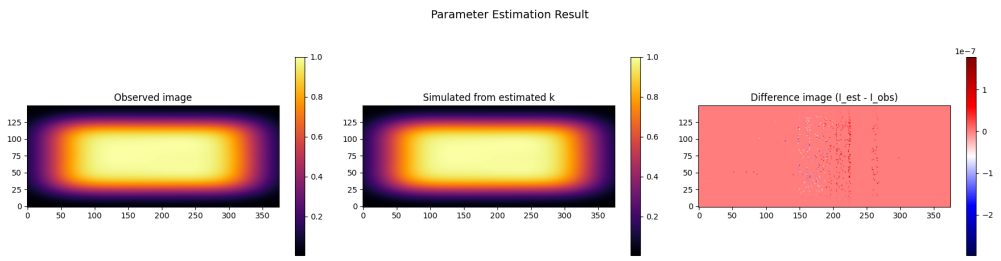

Figure 4.3: Parameter estimation from a full 3 ms pulse image. The model recovers the parameters $(A_x, A_y, \sigma_x, \sigma_y, c_x, c_y)$ with near perfect accuracy. The perturbation was chosen as `[5.0, -3.0, 1.0, -1.0, 2.0, -1.5, 1.0, -3.0]`

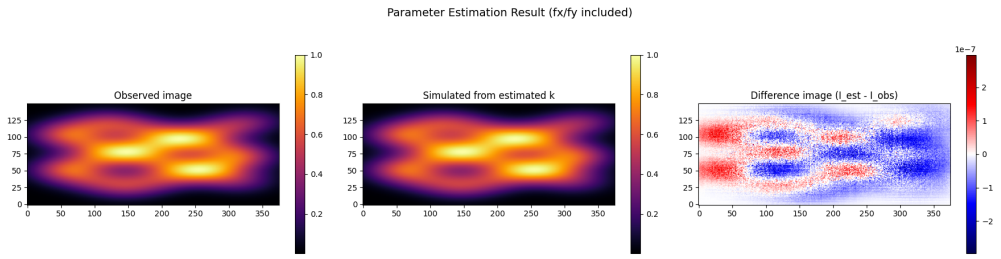

Figure 4.4: Parameter estimation from a short 50 μs exposure. Here, all 8 parameters are treated as unknown, including the raster frequencies f_x and f_y. The optimization recovers all values, indicating that frequencies are identifiable from short-pulse images.

Tables 4.3 and 4.2 give the numerical details of the same minimization. We note that the error is tiny for each individual parameter.

Estimated beam parameters: long pulse

Parameter	True value	Recovered value	Absolute error
A_x	6.00×10^{1}	6.00×10^{1}	0.00×10^{0}
A_y	2.00×10^{1}	2.00×10^{1}	0.00×10^{0}
σ_x	1.35×10^{1}	1.35×10^{1}	0.00×10^{0}
σ_y	5.05×10^{0}	5.05×10^{0}	0.00×10^{0}
c_x	0.00×10^{0}	2.84×10^{-7}	2.84×10^{-7}
c_y	0.00×10^{0}	-7.47×10^{-9}	7.47×10^{-9}
f_x	3.96×10^{1}	3.96×10^{1}	0.00×10^{0}
f_y	2.90×10^{1}	2.90×10^{1}	0.00×10^{0}

Table 4.2: Recovered beam parameters for the long-pulse case (3 ms), using gradient-based minimization from a perturbed initial guess. Final loss was 1.89×10^{-16} and parameter error norm was 2.85×10^{-7}.

Estimated beam parameters: short pulse

Parameter	True value	Recovered value	Absolute error
A_x	6.00×10^{1}	6.00×10^{1}	1.85×10^{-6}
A_y	2.00×10^{1}	2.00×10^{1}	2.38×10^{-7}
σ_x	1.35×10^{1}	1.35×10^{1}	4.77×10^{-7}
σ_y	5.05×10^{0}	5.05×10^{0}	4.05×10^{-7}
c_x	0.00×10^{0}	-3.86×10^{-6}	3.86×10^{-6}
c_y	0.00×10^{0}	1.77×10^{-7}	1.77×10^{-7}
f_x	3.96×10^{1}	3.96×10^{1}	9.06×10^{-7}
f_y	2.90×10^{1}	2.90×10^{1}	4.05×10^{-7}

Table 4.3: Recovered beam parameters using gradient-based minimization starting from a perturbed initial guess. Final loss was 1.74×10^{-14} and parameter error norm was 4.42×10^{-6}.

We finally show in figure 4.5 that both loss and parameter error actually converge numerically (here using only the short pulse case to limit the number of plots). The curves only flatten out when the loss values reach levels close to machine epsilon. At this level it becomes difficult for the solver to actually measure any improvements, and the subroutines in `scipy` force termination.

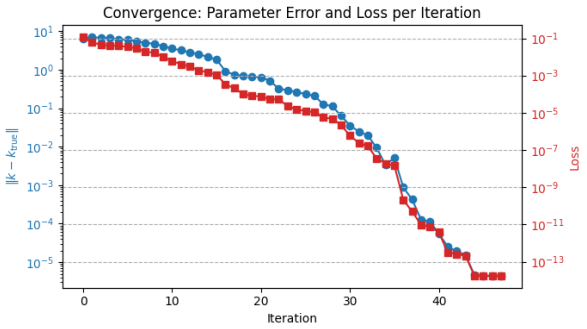

Figure 4.5: Convergence of the minimization algorithm. The blue curve shows the distance between the current parameter vector $\mathbf{k}$ and the true parameter vector $\mathbf{k}_{\text{true}}$ at each iteration, while the red curve shows the value of the loss function.

4.4.2 Minimization for noisy image

Next, we present the corresponding results to the ones in the previous subsection, but for problem $(\tilde{P})$. For brevity we only focus on the shorter pulse here. In Figure 4.6 we see the result of estimating $\mathbf{k}$ with noise corresponding to a Poisson count of $\lambda = 20\ 000$, as described in Section 2.2.2.

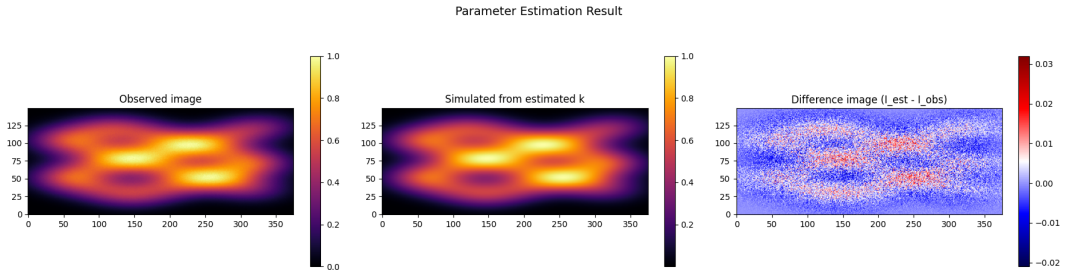

Figure 4.6: Parameter estimation under Poisson noise. The left panel shows the observed noisy image $\tilde{I}_{\text{obs}}$, and the right panel shows the image simulated from the estimated parameters $\mathbf{k}$. The estimated parameters give a final loss of $\mathcal{L} = 1.17 \times 10^{-4}$ and a parameter error $\|\mathbf{k} - \mathbf{k}^*\| = 7.44 \times 10^{-3}$. The optimizer stopped after 43 iterations due to the relative reduction in loss reaching machine precision.

4.4.3 Multi-start minimization: a way to deal with non-convexity

To be able to use the minimization algorithm as a more reliable way to find the global minimum, one option is to simply try *many* initial points and only pick out the ones that converge as potential minimizers. Then, we can compare the images that they generate to the observed one and (hopefully) conclude which of our candidates fit the best and are the most likely. The results of such an approach are shown below. Here we imagine the scenario that we do not know anything about the parameters a priori other than the fact that they fall into certain intervals. Therefore, the experiment is conducted by sampling uniformly from these intervals to get initial points. We use nominal parameters $\mathbf{k}_{\text{nom}}$ and short pulse length $T = 50\,\mu s$

Quantity	Value
Total runs	100
Convergence threshold	0.01
Successful convergences	8
Success rate	8.0%
Minimum distance	8.8×10^{-6}
Maximum distance	2.2×10^{4}

Table 4.4: Results of the multi-start optimization test using 100 random initial points. In this case 8% of the runs converged within the tolerance $\|\mathbf{k} - \mathbf{k}^*\| < 0.01$, indicating convergence depends heavily on the choice of initial point.

4.4.4 Multi-start success probability

Here we develop an idea to make the multi-start method more rigorous, by analyzing it statistically. When we launch many independent optimizations from random initial points, each run either "converges well" (in the sense of ending below a chosen loss threshold) or it does not. If we denote the success probability by p, then running n independent starts is modeled as repeating a Bernoulli experiment n times. In other words, the number of successful convergences, X, behaves as a random variable drawn from the binomial distribution,

$$X \sim \text{Binomial}(n, p),$$

which has probability mass function

$$\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \qquad k = 0, 1, \ldots, n.$$

The event we care about is simply "at least one success". This is the complement of getting zero successes, so

$$\mathbb{P}(X \geq 1) = 1 - (1-p)^n.$$

Using this expression we can answer the practical question of how many restarts we need to get, for example, $\alpha = 99\%$ certainty that at least one of them converges well. Solving $1-(1-p)^n \geq \alpha$ gives the following lower bound for the number of multi–starts required to achieve a confidence level α:

$$\boxed{n \;\geq\; \frac{\ln(1-\alpha)}{\ln(1-p)}} \tag{4.18}$$

Once p is estimated empirically (by running a large batch of initial tests), this can provide guidance for choosing the number of initializations.

In Figure 4.7 we fix $\alpha = 99\%$ and compute the required number of multi–start runs for a range of hypothetical success probabilities p. We see that for moderate values $p > 5\%$, a below one

hundred starts are sufficient for this confidence level, while small probabilities can require several hundred starts.

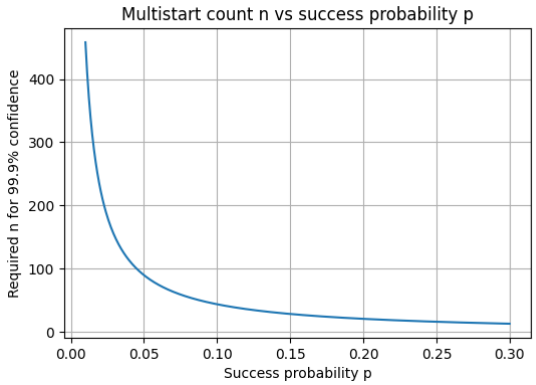

Figure 4.7: Required number of multi–start initializations n needed to reach a confidence level of 99% that at least one run converges well, as a function of the empirical success probability p. The curve follows the formula $n = \ln(1-\alpha)/\ln(1-p)$ with $\alpha = 0.99$ and $p \in [0.01, 0.30]$.

4.5 Performance on test data

In this section we present the test results of the optimizer in terms of the estimation errors; in the first subsection for the Synthetic-uniform data and in the second subsection for the Synthetic-prior data. The first table of each subsection shows average and median errors, while the second table shows the signed errors for each of the ten pulse lengths (corresponding to the ten different images).

4.5.1 Synthetic-uniform dataset

In Table 4.5 we see that the first six parameters are generally well estimated. The last four (the frequencies and the phases) are significantly worse.

Parameter	**Range**	**RMSE**	**Median abs**
A_x	0–65	0.315	0.054
A_y	0–25	2.317	0.131
σ_x	2–20	0.123	0.039
σ_y	2–20	0.299	0.080
c_x	-20–20	0.048	0.039
c_y	-20–20	0.159	0.019
f_x	20–50	9.370	2.266
f_y	20–50	12.042	7.640
ϕ_x (rad)	$[-\pi, \pi]$	1.652	0.553
ϕ_y (rad)	$[-\pi, \pi]$	1.227	0.300

Table 4.5: Absolute estimation errors for the uniform test set. Ranges are taken from the parameter bounds used in optimization.

Table 4.6 shows that the loss for the image corresponding to the parameter estimate for each

image is always small, indicating that even when parameter estimates were inaccurate the parameters explained the data well. This is in line with the theoretical results about ambiguity in Chapter 3.

	Pulse length (ms)									
Parameter	**0.050**	**0.079**	**0.124**	**0.196**	**0.309**	**0.486**	**0.766**	**1.208**	**1.903**	**3.000**
A_x	0.0090	−0.0040	−0.039	−0.076	0.16	0.070	−0.11	−0.97	−0.0070	−0.011
A_y	−0.017	−0.0030	−3.4	6.5	0.090	−0.71	−0.11	−0.15	−0.36	−0.020
σ_x	0.041	0.012	−0.0090	0.053	−0.34	0.031	0.036	0.16	0.047	0.015
σ_y	−0.0040	0.0020	0.29	−0.86	−0.077	0.18	−0.0010	0.083	0.16	0.060
c_x	−0.0070	−0.0030	−0.046	0.013	−0.073	0.10	−0.035	−0.042	−0.0050	0.053
c_y	−0.021	0.0060	−0.40	0.032	0.029	0.30	0.0050	0.018	−0.012	0.0040
f_x	0.93	−0.0040	−0.0020	−0.054	3.1	−13	21	−12	−10	−1.4
f_y	0.70	−0.017	18	−9.6	6.4	−13	−6.3	−27	8.8	−2.4
ϕ_x	0.000	0.0010	−2.0	2.9	0.15	−0.73	−2.8	0.37	−2.6	0.25
ϕ_y	0.0010	0.0010	0.20	0.19	0.078	−2.0	0.61	0.40	2.4	0.016
Loss	0.000 010	0.000 00	0.000 010	0.000 010	0.000 070	0.000 080	0.000 010	0.000 010	0.000 020	0.000 020

Table 4.6: Estimation errors for the Synthetic-uniform test set (angles in rad).

4.5.2 Synthetic-prior dataset

In Table 4.7 we see that all parameters are reasonably well estimated for the Synthetic-prior data. The same pattern can be seen in Table 4.8. Related to Chapter 3, most of the ambiguity that causes large errors for Synthetic-uniform data seems to disappear when the initial points of the optimizer are sampled sufficiently close the true parameters.

Parameter	**Range**	**RMSE**	**Median abs**
A_x	0–65	0.118	0.020
A_y	0–25	0.046	0.012
σ_x	2–20	0.174	0.031
σ_y	2–20	0.044	0.032
c_x	-20–20	0.019	0.007
c_y	-20–20	0.005	0.001
f_x	20–50	1.814	0.540
f_y	20–50	1.098	0.416
ϕ_x (rad)	$[-\pi, \pi]$	1.220	0.503
ϕ_y (rad)	$[-\pi, \pi]$	0.993	0.279

Table 4.7: Absolute estimation errors summary.

	Pulse length (ms)									
Parameter	**0.050**	**0.079**	**0.124**	**0.196**	**0.309**	**0.486**	**0.766**	**1.208**	**1.903**	**3.000**
A_x	0.018	−0.038	−0.0080	−0.0040	−0.0040	−0.35	0.014	0.13	−0.021	0.025
A_y	0.0090	−0.0020	−0.0060	−0.0040	−0.0030	−0.12	−0.017	0.066	−0.036	−0.015
σ_x	0.035	0.043	−0.014	0.015	0.0080	0.53	−0.010	−0.11	0.035	−0.028
σ_y	0.024	0.014	0.017	0.030	0.023	0.091	0.034	−0.049	0.063	0.035
c_x	−0.0020	0.000	−0.0070	−0.0080	−0.0060	0.024	−0.0020	−0.011	0.0070	0.050
c_y	0.000	0.0030	0.000	0.0020	0.000	0.000	0.011	0.0050	0.011	0.000
f_x	0.78	0.0020	0.0010	−0.0060	−0.0060	−0.30	1.3	2.6	4.7	1.3
f_y	0.59	−0.0040	−0.0010	−0.0040	−0.0010	−0.24	1.3	1.7	2.6	0.64
ϕ_x	−0.0010	−0.0010	−0.0040	−1.7	0.0030	1.0	2.5	−0.43	0.58	2.0
ϕ_y	0.0020	−0.0020	−0.0030	−0.49	0.0050	0.77	−0.37	−0.19	0.76	2.9
Loss	0.000 020	0.000 010	0.000 010	0.000 00	0.000 010	0.000 47	0.000 010	0.000 020	0.000 020	0.000 010

Table 4.8: Estimation errors for the Synthetic-prior test set (angles in rad).

4.6 Conclusions

From the experiments in the previous section we can conclude that the optimizer is likely to work well whenever we know approximately what the true parameters are, so that we can choose an initial point close to the minimum. When we do not have that prior knowledge, it is more difficult, but the multi-start approach works to some extent; a subset of the initial points tend to converge to the correct parameters. There are however issues with this method, which can be summarized in the following points:

- As concluded in earlier sections, we do not have strong identifiability in general. Several different combinations of parameters can yield very similar results in terms of both loss and the overall structure of the image, especially when noise is significant.
- There is therefore a need for a more sophisticated method of choosing the most likely solution. This selection will inherently depend on the user's prior beliefs about the parameters.
- The numerical optimization method does not offer a natural framework for analyzing uncertainty or quantifying probabilities in the way that is desirable.

This motivates alternative methods that are based on statistics, which will be the topic of the next chapter.

Chapter 5

Estimation with Markov chain Monte Carlo

In this chapter we mirror the structure of Chapter 4 as we define, implement and test our second numerical method called Markov Chain Monte Carlo.

5.1 A Bayesian problem formulation

5.1.1 Posterior distribution and point estimates

Up until this point we have exclusively worked with $(\tilde{P})$ as our formulations of the inverse problem. Now we turn our attention to the other main formulation typically used in the literature: a *Bayesian* approach (following Kaipio and Somersalo (2005), except where stated otherwise). Here, instead of viewing the parameter values as quantities that can be definitively right or wrong, we view them as more or less plausible. In other words we model them as random variables belonging to some probability distribution. The main quantities we are concerned with are

1. The **prior probability distribution** $p(\mathbf{k})$. This describes how likely we think a parameter vector is before we see the image. Naturally, this can use domain knowledge like possible parameter ranges and expected values (what we have set the parameters to on the machines).

2. The **likelihood** $p(\tilde{I}_{\text{obs}} \mid \mathbf{k})$. This describes how likely it is, for a given parameter vector $\mathbf{k}$, that the forward model would produce exactly the observed image $\tilde{I}_{\text{obs}}$. Intuitively, a parameter vector $\mathbf{k}$ for which the corresponding model image $I_{\text{model}}(\mathbf{k})$ differs significantly from $\tilde{I}_{\text{obs}}$ will give low likelihood.

3. The **posterior distribution** $p(\mathbf{k} \mid \tilde{I}_{\text{obs}})$. This describes how likely we think a given $\mathbf{k}$ is

after we have observed the image. For notational simplicity we will often denote it $\pi(\mathbf{k})$.

Given these probabilities the main problem in a Bayesian formulation becomes: using the prior and likelihood, estimate the posterior distribution $p(\mathbf{k} \mid \tilde{I}_{\text{obs}})$. If point estimates are also of interest, which they clearly are in our case, it is possible to define these in terms of the posterior distribution via the *posterior mean*

$$\mathbf{k}_{\text{PM}} = \mathbb{E}\Big[\, \mathbf{k} \mid \tilde{I}_{\text{obs}} \Big] = \int \mathbf{k}\, p(\mathbf{k} \mid \tilde{I}_{\text{obs}})\, d\mathbf{k}, \tag{5.1}$$

which, as shown in Kaipio and Somersalo (2005, Sec. 5.1.2), is also the estimator that minimizes the posterior expected squared error,

$$\mathbf{k}_{\text{PM}} = \arg\min_{\tilde{\mathbf{k}}} \mathbb{E}\Big[\| \mathbf{k} - \tilde{\mathbf{k}} \|_2^2 \,\Big|\, \tilde{I}_{\text{obs}} \Big].$$

In other words, the posterior mean is the point estimate that minimizes the parameter error "on average" over infinitely many draws from the posterior distribution. This is the estimator we use for all tests in this chapter. In practice, the posterior mean in (5.1) is approximated using samples $\{\mathbf{k}^{(i)}\}_{i=1}^{N_{\text{MC}}}$ drawn from the posterior distribution, yielding the Monte Carlo estimator

$$\mathbf{k}_{\text{PM}} \;\approx\; \frac{1}{N_{\text{MC}}} \sum_{i=1}^{N_{\text{MC}}} \mathbf{k}^{(i)}.$$

Remark. Note that we are no longer merely interested in solving the problem of finding a point estimate close to the true parameters, as in $(\tilde{P})$. However it is still an important subproblem, and the only way we can directly compare performance to the BFGS.

5.1.2 Bayes' formula

The fundamental mathematical tool that lets us compute posterior probabilities is the well-known Bayes' formula which states that

$$p(\mathbf{k} \mid \tilde{I}_{\text{obs}}) = \frac{p(\tilde{I}_{\text{obs}} \mid \mathbf{k})\, p(\mathbf{k})}{p(\tilde{I}_{\text{obs}})}.$$

In practice a slightly weaker version of Bayes' formula is often used. When we are only interested in how likely different parameter vectors are relative to each other, and not the exact probabilities that each of them has, it is sufficient to estimate the *unnormalized posterior*, defined as the right hand side of the proportionality expression

$$p(\mathbf{k} \mid \tilde{I}_{\text{obs}}) \propto p(\tilde{I}_{\text{obs}} \mid \mathbf{k})\, p(\mathbf{k}).$$

The only difference to the actual posterior distribution is that the denominator $p(\tilde{I}_{\text{obs}})$, known as the *evidence*, has been discarded. This simplifies the estimation process greatly, because the evidence, given by the integral $p(\tilde{I}_{\text{obs}}) = \int p(\tilde{I}_{\text{obs}} \mid \mathbf{k})\, p(\mathbf{k})\, d\mathbf{k}$, is often costly or even infeasible to compute in practice.

Remark. As we will in practice exclusively work with unnormalized posteriors in this thesis, we will mostly just call them "posteriors" for simplicity.

5.2 Markov chain Monte Carlo methods

5.2.1 Introduction to Markov chain Monte Carlo

In this section we will give a brief overview of the ideas behind the Markov Chain Monte Carlo methods, which form one of the most well known approaches to Bayesian inference. First we will focus on the most foundational implementation: the *Metropolis–Hastings* algorithm (sometimes just abbreviated MH). The theory is adapted from Kaipio and Somersalo (2005).

Motivation and Intuition

The basic idea behind all MCMC methods is to construct a sequence of parameter samples $\{\mathbf{k}_i\}_{i=1}^{N}$ such that the distribution of samples approximates the posterior distribution.

In theory, if we could sample infinitely many $\mathbf{k}$ values across the entire parameter space, compute their corresponding priors and likelihoods, and evaluate the product $p(\tilde{I}_{\text{obs}} \mid \mathbf{k})\, p(\mathbf{k})$ for each, we could reconstruct the entire posterior distribution exactly. In practice, this is of course infeasible. MCMC methods instead aim to generate a finite number of *informative* samples (concentrated in regions of high posterior probability) that still yield a good approximation. A good sampling strategy should therefore:

- Spend more time in high-probability regions, since these contribute most to the shape of the distribution.
- Occasionally explore lower-probability regions, to avoid getting stuck in local maxima and ensure that the entire distribution is captured.

MCMC achieves this by defining two key components:

- A **proposal distribution**, which suggests the next candidate sample given the current one.
- An **acceptance rule**, which determines whether the proposed sample is accepted or rejected.

Multiple Chains

An important part of the concept of MCMC is that one can run multiple chains (often in parallel), initialized from different locations in parameter space. This allows for broader exploration and better detection of multimodal distributions (meaning; distributions with several local peaks). In some MCMC variants, the chains may also interact with each other as we will see in Section 5.2.3 about a method called *parallel tempering* MCMC.

5.2.2 The Metropolis–Hastings algorithm

Before we describe the steps of the algorithm, we need to define the concepts of a *proposal distribution* and an *acceptance ratio* properly. The proposal distribution, denoted $q(\mathbf{k}' \mid \mathbf{k}_n)$, specifies how the next candidate sample $\mathbf{k}'$ is generated given the current sample $\mathbf{k}_n$. In the language of Markov chains, $\mathbf{k}'$ is simply the next state that the chain may transition to.

The acceptance ratio is a number between 0 and 1 that expresses how likely we are to accept the proposed sample $\mathbf{k}'$ compared to the current sample $\mathbf{k}_n$. It is defined as

$$\alpha_{\text{jump}} = \min\left(1,\ \frac{\pi(\mathbf{k}')\, q(\mathbf{k}_n \mid \mathbf{k}')}{\pi(\mathbf{k}_n)\, q(\mathbf{k}' \mid \mathbf{k}_n)}\right),$$

where $\pi(\mathbf{k})$ is used to denote the (unnormalized) posterior distribution. The term $q(\mathbf{k}' \mid \mathbf{k}_n)$ is the probability of proposing the reverse move, and is needed to ensure convergence to the correct distribution. Note that if the proposal distribution is symmetric, meaning

$$q(\mathbf{k}_n \mid \mathbf{k}') = q(\mathbf{k}' \mid \mathbf{k}_n),$$

the acceptance ratio simplifies to

$$\alpha_{\text{jump}} = \min\left(1, \frac{\pi(\mathbf{k}')}{\pi(\mathbf{k}_n)}\right).$$

This will always be the case throughout this thesis, since in our implementation we use a Gaussian proposal distribution, defined by

$$q(\mathbf{k}' \mid \mathbf{k}_n) = \mathcal{N}\big(\mathbf{k}';\, \mathbf{k}_n,\, \Sigma_{\text{prop}}\big),$$

where $\mathbf{k}_n$ is the expected value and Σ_{prop} is the (diagonal) covariance matrix controlling the step sizes in each parameter dimension.

Algorithm 2 Metropolis–Hastings

Given a starting value $\mathbf{k}_0$, the Metropolis–Hastings algorithm generates a sequence of samples $\{\mathbf{k}_n\}_{n=1}^{N}$ as follows:

1. **Propose a candidate:** Draw a new candidate parameter vector $\mathbf{k}' \sim q(\mathbf{k}_n \mid \cdot)$ from the proposal distribution.
2. **Compute acceptance ratio:** Evaluate the acceptance probability
$$\alpha_{\text{jump}} = \min\left(1, \frac{\pi(\mathbf{k}')\, q(\mathbf{k}' \mid \mathbf{k}_n)}{\pi(\mathbf{k}_n)\, q(\mathbf{k}_n \mid \mathbf{k}')}\right).$$
3. **Accept or reject:** Draw a uniform random number $u \sim \mathcal{U}[0,1]$.
   - If $u < \alpha_{\text{jump}}$, accept the proposal and set $\mathbf{k}_{n+1} = \mathbf{k}'$.
   - Otherwise, reject the proposal and set $\mathbf{k}_{n+1} = \mathbf{k}_n$.
4. **Repeat:** Increment n and return to step 1 until N total samples have been generated.

This process defines a Markov chain whose stationary distribution is $p(\mathbf{k})$, meaning that as $N \to \infty$, the empirical distribution of samples approximates the target posterior (under mild assumptions, see Theorem 5.1). It is worth noting that the method only requires knowledge of $p(\mathbf{k})$ up to a constant (any such constant will cancel in the fraction inside the acceptance ratio), which means that in our case we are free to use the unnormalized posterior instead of the normalized one.

5.2.3 Parallel tempering: extending the basic MH algorithm

One of the ways to improve the basic Metropolis–Hastings algorithm is to use something called *parallel tempering* (Hanada and Matsuura 2022). This is a kind of multi-chain MCMC that is especially suited for multimodal posteriors where the chains struggle to move between different modes. Since our inverse problem is likely to have such characteristics (based on ambiguity results in Chapter 3), we choose to pursue this particular algorithm.

The idea behind parallel tempering is the following: to allow the chains to move more freely across the posteriors, we "even out" the posteriors by raising the likelihood to a power $1/T$. In other words, we consider chains with tempered posteriors

$$p_T(\mathbf{k} \mid \tilde{I}_{\text{obs}}) \propto p(\mathbf{k})\, p(\tilde{I}_{\text{obs}} \mid \mathbf{k})^{1/T}. \tag{5.2}$$

where $T \geq 1$ is the temperature of the chain. Each of these chains is run with the usual Metropolis–Hastings steps, but on top of that, we periodically allow for *swapping* of samples between different chains according to a particular rule, similar to the acceptance ratio in M-H. The intuition behind this is that the warmer chains can find promising samples (ones that have

large posterior density) and transfer these to the colder chains, which are more static due to the sharp posteriors. At the end of the algorithm the samples from the coldest chain, corresponding to T = 1, should approximate the true posterior.

We now present an overview of the algorithm. For notational simplicity, we define the inverse temperature as $\beta_i = \frac{1}{T_i}$.

Algorithm 3 Parallel Tempering / Replica Exchange MCMC

Given N chains initialized at inverse temperatures $\beta_1 = 1 < \beta_2 < \cdots < \beta_N$, the parallel tempering algorithm proceeds as follows:

1. **Advance each chain independently:** For each chain $i = 1, \ldots, N$, run one Metropolis–Hastings according to Algorithm 2, giving a sample of the tempered posterior
$$\pi_{\beta_i}(\mathbf{k}) \;\propto\; p(\mathbf{k})\, p(\tilde{I}_{\text{obs}} \mid \mathbf{k})^{\beta_i}.$$
2. **Compute swap acceptance ratio:** For each adjacent pair of chains $(i, j = i + 1)$, compute the probability of swapping the current samples $\mathbf{k}_i \leftrightarrow \mathbf{k}_j$ using the likelihoods via
$$\alpha_{\text{swap}} = \min\left(1,\; \left[\frac{p(\tilde{I}_{\text{obs}} \mid \mathbf{k}_j)}{p(\tilde{I}_{\text{obs}} \mid \mathbf{k}_i)}\right]^{\beta_i - \beta_j}\right). \tag{5.3}$$
3. **Accept or reject swaps:** For each pair, draw $u \sim \mathcal{U}[0, 1]$.
   - If $u < \alpha_{\text{swap}}$, accept the swap.
   - Otherwise, keep the current samples in the respective chains.
4. **Repeat:** Iterate steps 1–3 for a fixed (large) number of iterations. The coldest chain (with $\beta = 1$) approximates the true posterior.

The intuitive idea behind parallel tempering is illustrated in Figure 5.1 below. In this example the true posterior distribution is essentially zero between the two different modes, making it impossible in practice for a chain to move between them. The warm chain's posterior however is significantly flattened. This makes movement between the modes far more likely, which allows the parallel tempering MCMC to correctly approximate the true posterior.

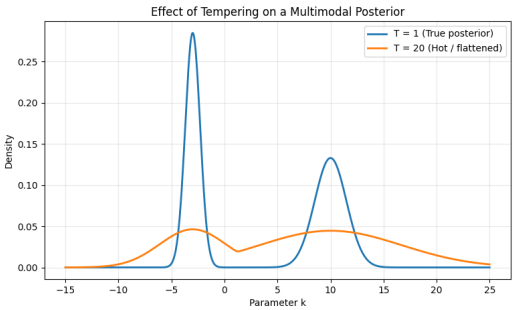

Figure 5.1: Illustration of how tempering affects a multimodal posterior distribution. The blue curve shows the original posterior ($T = 1$), consisting of two distinct modes. The orange curve shows a tempered posterior ($T = 20$), where raising the likelihood to the power $1/T$ flattens the distribution.

5.2.4 Convergence of MCMC methods

So far we have introduced both the basic Metropolis-Hastings method and the extended method of parallel tempering. Each of these defines a Markov chain that is supposed to approximate the true posterior distribution when the total number of iterations N is large. Here we give the theoretical motivation for why that is the case; we show that as N goes to infinity the distribution of the chain converges to a unique stationary distribution π.

We start by stating the fundamental convergence theorem for discrete Markov Chains, which can be found along with the proof in Häggström (2002). For definitions of all the necessary terminology, see Appendix D.2.

Theorem 5.1. *Let $\{X_i\}_{i=0}^{\infty}$ be an aperiodic, irreducible Markov chain defined by a transition matrix P and a finite state space S with $|S| = m$. Further, let $\pi^{(i)}$ denote the probability distribution of the m states at step i, and let π be the unique stationary distribution. Then, for any initial distribution $\pi^{(0)}$, the distribution $\pi^{(i)}$ converges to the stationary distribution in total variation:*

$$\pi^{(i)} \xrightarrow{\text{TV}} \pi$$

Here we are mainly interested in the application of the theorem, specifically the fact that we get convergence when running any Metropolis-Hastings chain on a computer for our particular beam problem. Below we state this as a proposition, and give an informal proof. Note that the results stated below are *not* taken from Häggström (2002). The same applies to Proposition 5.2.

Proposition 5.1. Consider the Metropolis–Hastings implementation in Algorithm 2, run on the bounded parameter domain described in Section 2.1.4, with a Gaussian proposal distribution and an arbitrary initial point $\mathbf{k}^{(0)} \in \Omega$. Let $\pi^{(i)}$ denote the distribution of $\mathbf{k}^{(i)}$. Then

$$\pi^{(i)} \xrightarrow{\text{TV}} \pi, \qquad \pi(\mathbf{k}) \propto p(\mathbf{k})\, p(\tilde{I}_{\text{obs}} \mid \mathbf{k}),$$

as $i \to \infty$, i.e. the distribution of the Metropolis–Hastings samples converges to the true posterior distribution.

Proof sketch. To apply the Markov chain convergence theorem, we only need to verify its assumptions for our Metropolis–Hastings chain.

- *Finite state space.* On a computer the parameter space consists of floating–point values in bounded intervals. Hence the Markov chain lives on a finite state space (though extremely large).
- *Aperiodicity.* The chain can stay in the same state in a single step whenever a proposal is rejected. Therefore, every state can return to itself in one step, so the chain is aperiodic.
- *Irreducibility.* All we need to check is that any state can eventually reach any other. This follows from the following observations:
  1. The unnormalized posterior $p(\mathbf{k})\, p(\tilde{I}_{\text{obs}} \mid \mathbf{k})$ is strictly positive everywhere on our bounded domain (neither the priors nor the Poisson likelihood are ever exactly zero). See the remark at the end of this section.
  2. The Gaussian proposal can in principle propose points arbitrarily close to any $\mathbf{k}$ in the parameter space.

With these conditions satisfied, the Markov chain convergence theorem guarantees that the distributions $\pi^{(i)}$ produced by the Metropolis–Hastings algorithm converge in total variation to the true posterior $p(\mathbf{k} \mid \tilde{I}_{\text{obs}})$. □

Finally, we state that the extension of a basic Metropolis-Hastings chain to parallel tempering does not change the convergence property.

Proposition 5.2. For the parallel tempering implementation in Algorithm 3, the coldest chain (that is, the chain with $\beta = 1$) converges in total variation to the true posterior distribution:

$$\pi^{(i)}_{\beta=1} \xrightarrow{\text{TV}} p(\mathbf{k} \mid \tilde{I}_{\text{obs}}) \qquad \text{as } i \to \infty.$$

Proof sketch. The idea is essentially the same as for a single Metropolis–Hastings chain. Each chain in Algorithm 3 behaves exactly like an ordinary MH chain targeting its tempered posterior,

except for occasional swap moves between neighboring chains. These swaps do not destroy irreducibility or aperiodicity: they only exchange states between chains and occur with positive probability of doing nothing. As a result, the full parallel tempering procedure (MH updates + swap steps) still defines a Markov chain with a unique stationary distribution whose marginals are the tempered posteriors. The coldest chain, corresponding to $\beta = 1$, therefore converges to the true posterior:

$$\pi^{(i)}_{\beta=1} \xrightarrow{\text{TV}} p(\mathbf{k} \mid \tilde{I}_{\text{obs}}) \qquad \text{as } i \to \infty.$$

□

Remark. The Poisson likelihood is never zero in our setting, since every pixel intensity is strictly positive after summing the Gaussian contributions in the forward model $\mathcal{F}$.

5.3 Implementation

The implementation of Algorithm 3 (parallel tempering) is done fully in JAX, allowing large-scale parallel computation on GPUs. The particular method used for swapping is called deterministic even-odd swapping, and is for example used in (Neuhaus and Kadau 2009).

For clarity we summarize the specific forms of the quantities appearing in Bayes' formula for our implementation:

- **Prior** $p(\mathbf{k})$. Depending on the test scenario (see Section 2.4), we use either uniform priors over the bounded parameter domain or Gaussian priors centered at nominal values.
- **Likelihood** $p(\tilde{I}_{\text{obs}} \mid \mathbf{k})$. From Section 2.2 we assume that the forward model produces a noiseless simulated image $I_{\text{model}}(\mathbf{k})$, and that each pixel of the observed image is generated by independent Poisson noise,

  $$\tilde{I}_{\text{obs}}(i,j) \sim \text{Poisson}(\lambda_{ij}), \qquad \lambda_{ij} := c \cdot I_{\text{model}}(i,j;\mathbf{k}).$$

  This gives the likelihood

  $$p(\tilde{I}_{\text{obs}} \mid \mathbf{k}) = \prod_{i,j} \frac{\lambda_{ij}^{\tilde{I}_{\text{obs}}(i,j)} e^{-\lambda_{ij}}}{\tilde{I}_{\text{obs}}(i,j)!}.$$

  For computation, we use the log-likelihood,

  $$\log p(\tilde{I}_{\text{obs}} \mid \mathbf{k}) = \sum_{i,j} \Big[\tilde{I}_{\text{obs}}(i,j) \, \log I_{\text{model}}(i,j;\mathbf{k}) - I_{\text{model}}(i,j;\mathbf{k}) \Big] + \text{const},$$

  where the constant term does not depend on $\mathbf{k}$ and can be ignored in Metropolis–Hastings.
- **Posterior** $p(\mathbf{k} \mid \tilde{I}_{\text{obs}})$. Defined by Bayes' formula as the product of prior and likelihood (up to a normalization constant), and approximated in practice using histograms of the MCMC samples.

5.4 Tuning of MCMC parameters

In this section we present information about the tuning process of the parameters in the parallel tempering algorithm.

5.4.1 Trace plots

One of the most fundamental diagnostic tools for MCMC tuning is the use of trace plots, which simply means examining the evolution of each sampled parameter over iterations. There are two main things one wants to ensure:

1. The chain must not get stuck in incorrect modes; it should reach values close to the true parameter.
2. Once the chain reaches the high–probability region, it should continue to explore. If it does not, it behaves more like an optimizer than a method intended to approximate the full posterior distribution.

As shown in Figure D.1 in the Appendix, the chain satisfies both of these conditions for our nominal beam images. A key tuning parameter in this is the proposal standard deviation, which must be large enough to keep the chain moving.

5.4.2 Autocorrelation and effective sample size

The next important diagnostic tools to consider are the ones related to correlation of the samples for each given parameter. Ideally the samples become uncorrelated quickly, since this means that we are truly proposing "new" samples in the sense that they generate a lot of new information.

For a parameter sequence $\{x_t\}_{t=1}^{N}$, the autocorrelation at lag k is defined as

$$\rho(k) = \frac{\mathrm{Cov}(x_t,\, x_{t+k})}{\mathrm{Var}(x_t)}, \qquad k = 0, 1, 2, \ldots$$

where

$$\mathrm{Cov}(x_t, x_{t+k}) = \frac{1}{N-k}\sum_{t=1}^{N-k}(x_t - \bar{x})(x_{t+k} - \bar{x}), \qquad \mathrm{Var}(x_t) = \frac{1}{N}\sum_{t=1}^{N}(x_t - \bar{x})^2,$$

and $\bar{x}$ is the sample mean. The autocorrelation can further be used to calculate the integrated autocorrelation time (IACT) via

$$\tau_{\mathrm{int}} = \sum_{k=-\infty}^{\infty} \rho(k) = 1 + 2\sum_{k=1}^{\infty}\rho(k).$$

which essentially measures the area under the autocorrelation curve and hence quantifies how fast a sample becomes uncorrelated with itself.

Finally, the so–called effective sample size can be computed. This measures the effectiveness of the Monte Carlo sampling, more specifically how many *independent* samples were generated in the posterior approximation of each parameter out of the N total ones:

$$N_{\text{eff}} = \frac{N}{\tau_{\text{int}}}.$$

What one considers to be an acceptable N_{eff} value depends on the difficulty of the problem, but literature seems to agree on that at least a few hundred samples are required for a "good" approximation of the posterior distribution (Asparouhov and Muthén 2025). Below we present the results for our nominal beam image. Additionally, an example of an autocorrelation and IACT plot is shown in Figure D.2.

Parameter	**IACT**	N_{eff}
A_x	170	240
A_y	900	44
σ_x	1100	36
σ_y	1100	35
c_x	210	190
c_y	850	47
f_x	99	400
f_y	160	250
ϕ_x	99	400
ϕ_y	210	190

Table 5.1: Integrated autocorrelation times (IACT) and effective sample sizes (N_{eff}) for each parameter.

5.4.3 Acceptance rates

As described in the theory section, for a Metropolis–Hastings chain to explore the posterior effectively it must accept new proposals "often enough". According to standard results in the MCMC literature, a reasonable target range is an average acceptance rate of approximately 10–30%. In parallel tempering we work with many chains, each at its own temperature T_j, and ideally each chain's acceptance rate should lie within this interval.

To achieve this, each chain is given its own proposal standard deviation, scaled according to its temperature. A simple heuristic that is often used is

$$\sigma^{(j)}_{\text{proposal}} = \sigma_{\text{base}} \sqrt{T_j},$$

so that higher-temperature chains take larger steps, reflecting the fact that their tempered posteriors are smoother and more uniform-like. The base scale σ_{base} is chosen as a small fraction of the parameter range, for example on the order of 1/1000 of the parameter interval length.

In Figure D.3 we show how the acceptance rate varies across the chains for the example image considered. For the cold and medium–temperature chains the rates lie within the recommended range above 10 %, while the hottest chains have slightly lower rates.

5.4.4 Swap rates

The swap rates determine how often chains of different temperatures exchange samples with each other, and is hence crucial for the idea of parallel tempering to work – that the warmer chains can explore the posteriors more freely and pass on high–probability samples all the way up to the coldest chain (representing the actual approximation of the posteriors). According to Neuhaus and Kadau (2009), a broad range of swap rates can produce efficient mixing for the deterministic even–odd exchange scheme (DOE) used in this thesis; in their experiments, swap rates anywhere between 10% and 80% worked well. Achieving these kinds of rates across the entire chain is mainly a matter of building a suitable temperature ladder for the problem, so that adjacent posteriors are "similar enough", meaning that $\beta_i - \beta_j$ is small. This can be seen from the swap acceptance formula (5.3), where the quotient is close to one when the two temperatures are close.

In Figure D.4 in Appendix we show the swap rates over chains for a geometrically spaced beta ladder that was found to work well for the nominal beam image.

5.4.5 Posterior plots

Finally, one may inspect the posterior approximation plot for the simulations. If they show that the posteriors have high probability mass around the true parameter values, while also showing some spread, it is natural to think that they capture the true posterior too (if there is one main parameter configuration that can generate the image; if there are several we expect a high probability mass in those areas as well). An example of this kind of posterior plots can be found in the next section.

5.5 Experiments on nominal images; using prior information

To demonstrate how the parallel tempering MCMC algorithm works we focus on two different images in this section: we assume nominal parameters $\mathbf{k}$ for both but with different pulse lengths $T = 50\,\mu$s and $T = 0.5\,$ms respectively. The full MCMC configuration is described in Appendix D.1.

Below we show the estimated posterior distribution for the short pulse. We see that in this case the MCMC finds a good estimate for all parameters and shows very little ambiguity; the samples are centered around the true values and the standard deviations shown in Table 5.2 are low.

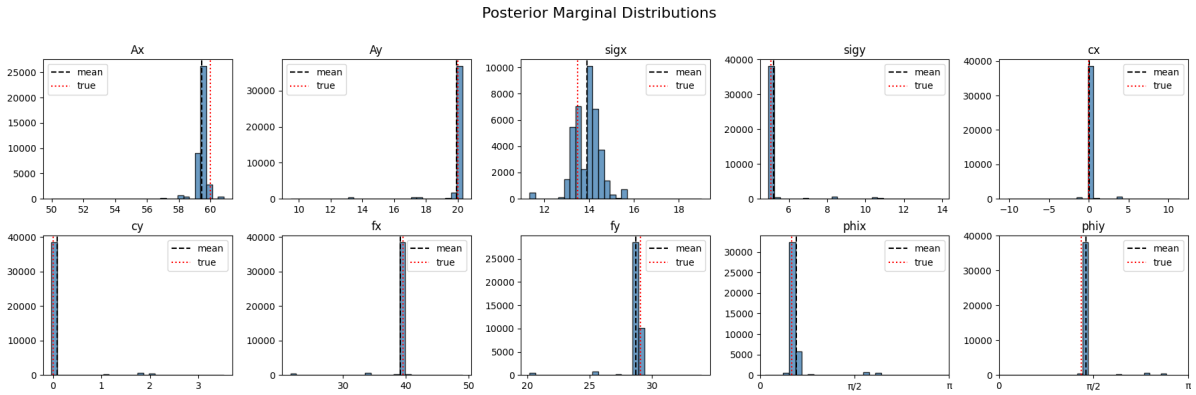

Figure 5.2: Posterior marginal distributions for all eight beam parameters in the non-ambiguous frequency case. Dashed black lines show the posterior means, and dotted red lines show the true values.

Parameter	**Mean**	**Std**	**True value**	**Error**
A_x	59.463	0.445	60.000	-0.537
A_y	19.940	0.833	20.000	-0.060
σ_x	13.903	0.595	13.500	$+0.403$
σ_y	5.195	0.767	5.050	$+0.145$
c_x	0.067	0.661	0.000	$+0.067$
c_y	0.092	0.333	0.000	$+0.092$
f_x	39.069	2.034	39.550	-0.481
f_y	28.715	1.027	29.050	-0.335
ϕ_x	0.602	0.211	0.524	$+0.079$
ϕ_y	1.449	0.194	1.366	$+0.083$

Table 5.2: Posterior means and standard deviations for the strong-identifiability case under uniform priors.

In the medium-length pulse case the frequency estimations are much larger than in the short pulse case, though still within 10 percent of the true parameter values. The marginal posterior distributions in Figure 5.3 and the standard deviations for the frequencies in Table 5.3 indicates that in fact the frequencies are not even close to being well identified; the posteriors clearly show two different frequency modes (one above the true frequencies and one below), and the standard deviations are huge. This shows a possible strength of MCMC: when it fails with some estimations, that tends to show up as large uncertainty, quantified by the standard deviations. In other words, even if the estimated values would have been exactly equal to the correct ones, large standard deviations would have revealed that we cannot trust these values.

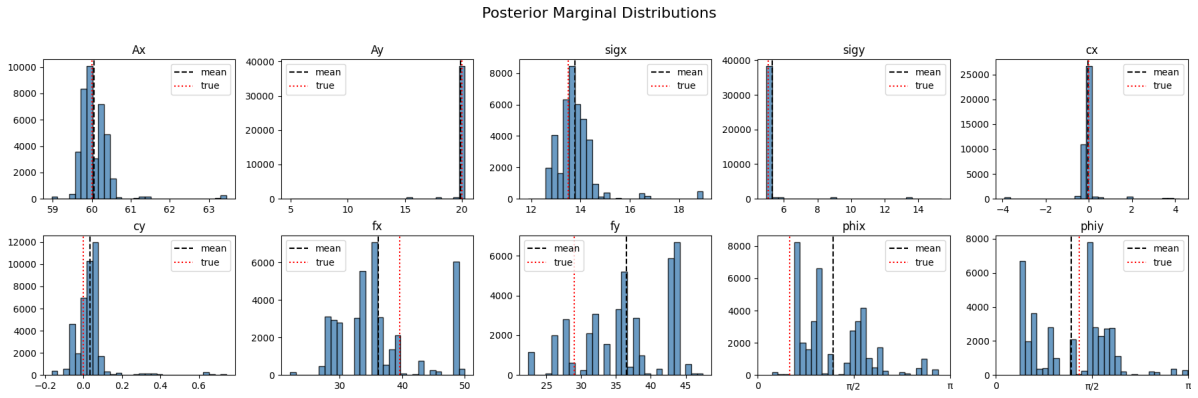

Figure 5.3: Posterior marginal distributions for all parameters.

Parameter	Mean	Std	True value	Error
A_x	60.066	0.419	60.000	$+0.066$
A_y	19.898	0.691	20.000	-0.102
σ_x	13.777	0.835	13.500	$+0.277$
σ_y	5.322	1.039	5.050	$+0.272$
c_x	-0.029	0.578	0.000	-0.029
c_y	0.033	0.088	0.000	$+0.033$
f_x	36.199	6.451	39.550	-3.351
f_y	36.481	6.186	29.050	$+7.431$
ϕ_x	1.228	0.555	0.524	$+0.705$
ϕ_y	1.232	0.611	1.366	-0.134

Table 5.3: Posterior means and standard deviations for the weak-identifiability case under **uniform priors**. Now the frecquencies and phases show high uncertainty, with large standard deviations and absolute errors.

Finally we show how prior information about the parameters can be integrated easily into the MCMC. Instead of having the priors be uniform as before we let them be Gaussians centered around the true (in reality *expected*) values with standard deviations that reflect how uncertain we are about the parameter values before receiving the image. In Table 5.4 below we assumed that the frequencies will be very close (on average within 1 percent) of the true values. If the data confirms this assumption, as in the present example, the Markov chains remain close to those values, giving accurate estimates with low standard deviations.

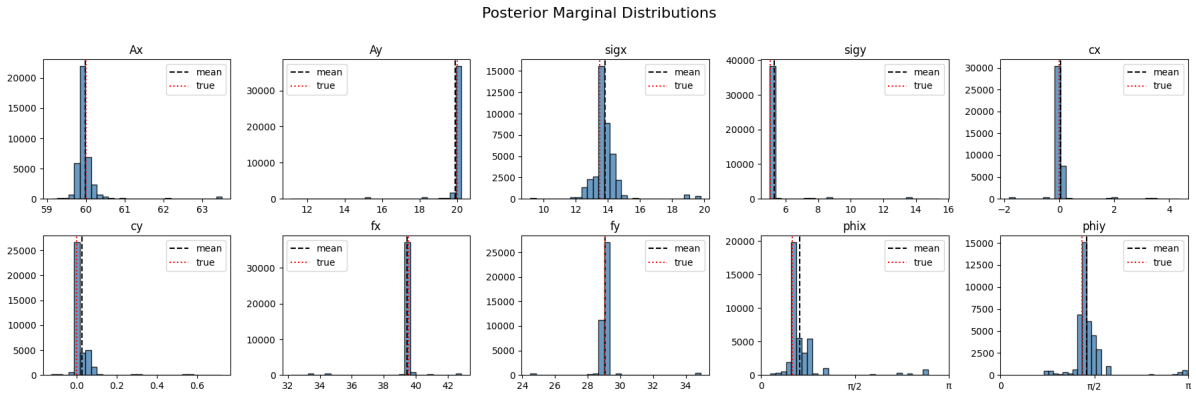

Figure 5.4: Posterior marginal distributions for the beam parameters in the Gaussian prior case.

Parameter	Mean	Std	True value	Error
A_x	59.987	0.369	60.000	-0.013
A_y	19.920	0.604	20.000	-0.080
σ_x	13.840	0.964	13.500	$+0.340$
σ_y	5.305	1.049	5.050	$+0.255$
c_x	0.051	0.458	0.000	$+0.051$
c_y	0.025	0.074	0.000	$+0.025$
f_x	39.458	0.920	39.550	-0.092
f_y	29.072	0.845	29.050	$+0.022$
ϕ_x	0.645	0.354	0.524	$+0.121$
ϕ_y	1.447	0.279	1.366	$+0.081$

Table 5.4: Posterior summary for the medium-length pulse using Gaussian priors. The priors were centered at the true parameter vector $\mathbf{k}_{\texttt{true}}$, with standard deviations set to 50% of the parameter range for amplitudes, widths, and positions, and only 0.1% for the frequencies (f_x, f_y). The strong priors for the frequencies removes ambiguity and leads to correct estimates with low standard deviation.

5.6 Performance on test data

5.6.1 Synthetic-uniform dataset

Table 5.5 shows a summary of how well the method performs on the Synthetic-uniform dataset. For each parameter, it reports two simple error measures taken over all test images: the root mean squared error (RMSE) and the median absolute error. We see that the estimations for the first six parameters are generally good, while frequencies and phases are not.

Parameter	Range	RMSE	Median abs
A_x	0–65	0.32	0.21
A_y	0–25	2.0	0.37
σ_x	2–20	0.20	0.14
σ_y	2–20	0.39	0.27
c_x	-20–20	0.12	0.080
c_y	-20–20	0.22	0.036
f_x	20–50	8.9	6.2
f_y	20–50	8.0	3.6
ϕ_x (rad)	$[-\pi, \pi]$	1.5	0.94
ϕ_y (rad)	$[-\pi, \pi]$	2.0	1.8

Table 5.5: Absolute estimation errors for Synthetic-uniform dataset (absolute errors; angles in rad). Ranges correspond to the parameter bounds used.

In Table 5.6 we show the full results with estimation errors for each image (corresponding to a unique pulse length). Among the first six parameters it is mainly A_x that was not always well estimated.

	Pulse length (ms)									
Parameter	**0.050**	**0.079**	**0.124**	**0.196**	**0.309**	**0.486**	**0.766**	**1.208**	**1.903**	**3.000**
A_x	0.0020	−0.049	−0.53	−0.63	0.15	−0.30	−0.096	0.41	−0.0050	−0.26
A_y	0.41	0.32	1.2	3.9	−0.34	−0.71	4.9	−0.23	−0.10	−0.077
σ_x	0.11	0.13	0.18	0.24	−0.046	0.45	0.023	−0.18	−0.045	0.16
σ_y	−0.80	−0.085	−0.48	−0.51	0.25	−0.20	−0.40	0.10	0.080	0.29
c_x	0.056	−0.051	−0.24	−0.12	−0.039	−0.14	0.019	−0.041	−0.10	0.17
c_y	−0.17	−0.040	−0.34	−0.074	0.033	0.58	0.0050	−0.018	0.013	−0.011
f_x	0.33	0.78	−3.1	15	0.45	−15	7.8	−4.5	−9.8	12
f_y	1.0	0.36	3.5	−3.3	−0.83	−16	−4.3	−14	12	−3.8
ϕ_x	0.017	0.72	1.1	0.49	1.8	−2.7	−2.2	−1.7	0.74	0.79
ϕ_y	1.2	1.5	−2.4	3.0	−1.3	2.0	−2.6	−2.3	1.2	−1.5
Loss	0.071	0.015	0.0070	0.000 83	0.0056	0.0048	0.000 30	0.000 15	0.000 13	0.0079

Table 5.6: All estimation errors for the uniform test set (angles in rad).

5.6.2 Synthetic-prior dataset

Below, in Table 5.7 and 5.8 we show similar tables as above but for the Synthetic-prior dataset. Here we note that all parameters are generally well estimated, including frequencies and phases.

Parameter	Range	RMSE	Median abs
A_x	0–65	0.11	0.022
A_y	0–25	0.023	0.015
σ_x	2–20	0.25	0.060
σ_y	2–20	0.096	0.096
c_x	-20–20	0.034	0.012
c_y	-20–20	0.0080	0.0020
f_x	20–50	1.1	0.43
f_y	20–50	0.92	0.23
ϕ_x (rad)	$[-\pi, \pi]$	0.22	0.090
ϕ_y (rad)	$[-\pi, \pi]$	0.10	0.069

Table 5.7: Absolute estimation errors for Synthetic-prior test set (absolute errors; angles in rad). Ranges correspond to the parameter bounds used in sampling.

	Pulse length (ms)									
Parameter	**0.050**	**0.079**	**0.124**	**0.196**	**0.309**	**0.486**	**0.766**	**1.208**	**1.903**	**3.000**
A_x	−0.0040	−0.13	−0.038	−0.018	−0.0050	−0.30	−0.024	−0.021	−0.026	−0.0010
A_y	−0.0040	0.015	−0.029	−0.0060	−0.031	−0.043	−0.029	−0.0080	−0.015	0.000
σ_x	0.011	0.20	−0.068	0.042	−0.13	0.74	0.051	0.046	0.078	0.023
σ_y	0.12	0.061	0.072	0.10	0.10	0.13	0.10	0.073	0.090	0.081
c_x	0.025	−0.015	−0.014	0.0010	−0.097	−0.030	−0.0020	−0.0030	−0.0090	−0.0030
c_y	−0.0010	0.013	0.0010	0.0010	−0.0020	0.0020	0.017	−0.0010	0.010	−0.0030
f_x	0.78	−0.016	−0.0060	0.0020	−0.076	−0.085	2.5	0.79	1.5	1.6
f_y	0.58	−0.021	−0.023	0.0040	−0.033	−0.077	2.0	0.38	1.5	1.3
ϕ_x	−0.0020	0.0020	−0.079	−0.18	0.015	0.57	−0.24	0.053	0.10	−0.21
ϕ_y	0.0040	−0.0030	−0.012	−0.053	0.027	−0.18	−0.12	0.086	0.12	−0.18
Loss	0.000 22	0.000 14	0.000 17	0.000 90	0.000 13	0.23	0.000 70	0.000 34	0.000 15	0.000 14

Table 5.8: All estimation errors for the Synthetic-prior test set (angles in rad).

5.7 Conclusions

The test results show a similar pattern to the one from numerical optimization. For the Synthetic-uniform set all parameters except frequencies and phases are in general well estimated across all pulse lengths. For shorter pulse lengths like 50 and 79 μs even frequencies and phases are possible to estimate. In the Synthetic-prior set where Gaussian priors are used, we see that all parameters are estimated much more accurately.

Chapter 6

Estimation on real images: initial tests

6.1 Long and short pulse examples

For long pulse lengths, the L^2 minimization reproduces the observed image well and yields parameter estimates that match the image visually. A typical result can be seen in Figure 6.1.

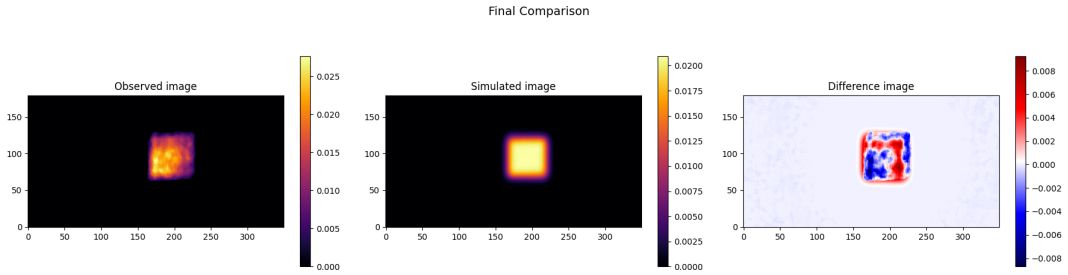

Figure 6.1: Long pulse image: L^2 minimization finds a visually accurate beam profile. Estimated parameters: $\mathbf{k} = (27.11,\ 26.91,\ 4.96,\ 5.68,\ 19.03,\ 4.99,\ 119.99,\ 137.09,\ 6.28,\ 0.00)$.

For short pulse lengths, the L^2 minimization tends to converge to a solution that matches only part of the observed image. An example of this is seen in Figure 6.2.

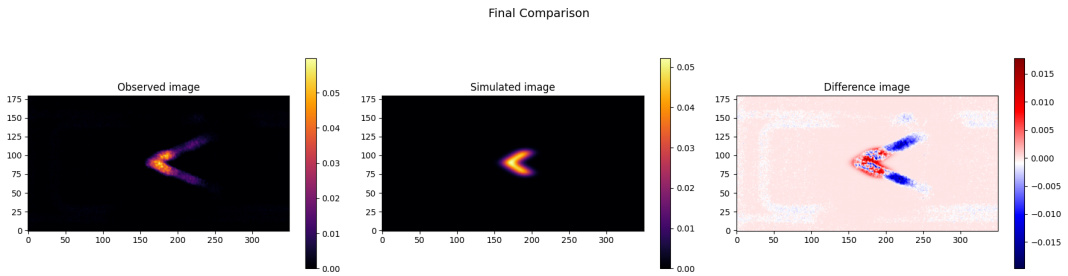

Figure 6.2: Short pulse image: L^2 minimization finds a local minimum that ignores a large part of the observed beam trajectory. Estimated parameters: $\mathbf{k} = (30.00,\ 17.17,\ 7.66,\ 3.17,\ 24.09,\ -0.29,\ 20.00,\ 21.00,\ 3.46,\ 5.07)$.

A way to prevent solutions that deviate too far from physically expected values is to introduce regularization terms as in (2.3), as described in Section 2.2. Applying this regularization to the raster amplitudes by defining

$$R(\mathbf{k}) = \gamma_1 \left(A_x - A_x^{\text{prior}}\right)^2 + \gamma_2 \left(A_y - A_y^{\text{prior}}\right)^2$$

yields a clear improvement, as shown in Figure 6.3. The reference values used were $A_x^{\text{prior}} = 90$ and $A_y^{\text{prior}} = 40$, with weights $\gamma_1 = \gamma_2 = 10^{-4}$.

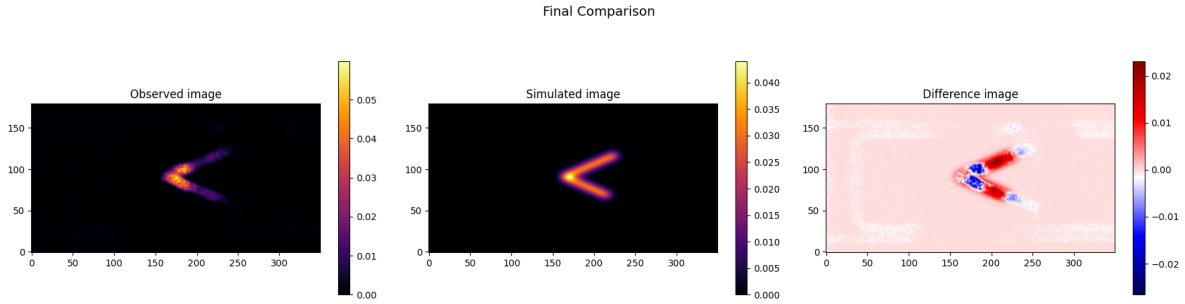

Figure 6.3: Short pulse image after having introduced regularization terms on the amplitudes. Estimated parameters: $\mathbf{k} = \big(77.6187, 32.0123, 5.2576, 4.5065, 66.1891, 0.1375, 20.0000, 20.0000, 3.4817, 5.0936\big)$.

6.2 Limitations of the L^2 norm

The L^2 norm measures pixel-wise intensity differences and does not take the geometric structure of the beam trace into account. As a result, visually important structures can be ignored if the overall squared error is reduced. Because of this, regularization and other tools are likely going to be needed when moving from simulated to real images, for example as shown above.

Chapter 7

Discussion, conclusions and future work

After having answered some fundamental questions about the inverse problem in Chapter 3 as well as implemented and tested the numerical optimization and Markov chain Monte Carlo methods in Chapter 4 and 5, we are now ready to summarize our findings. In Section 7.2 and 7.3 we also provide a practical path forward for ESS given the results, and concrete ideas for a longer-term research path.

7.1 Estimation method comparison

7.1.1 Summary of estimation performance

In Table 7.1 we see that for the Synthetic-uniform dataset, the optimizer gives more accurate estimates than the MCMC, except for the frequencies (though this is possibly a result of randomness). In Table 7.2 the results are more split; for the first six parameters the optimizer is slightly more accurate while for the frequencies and phases the MCMC performs better. For both tables it is however worth noting that all differences are relatively small; in other words, the performances are similar.

Parameter	**ΔRMSE**	**ΔMedian**
A_x	+0.007	+0.154
A_y	-0.272	+0.241
σ_x	+0.073	+0.102
σ_y	+0.088	+0.194
c_x	+0.070	+0.041
c_y	+0.063	+0.017
f_x	-0.461	+3.909
f_y	-3.995	-3.998
ϕ_x	-0.196	+0.385
ϕ_y	+0.771	+1.460

Table 7.1: Synthetic-uniform dataset. Difference between MCMC and Optimizer errors (MCMC − Optimizer). Negative values indicate that MCMC performs better.

Parameter	ΔRMSE	ΔMedian
A_x	-0.013	+0.002
A_y	-0.023	+0.003
σ_x	+0.076	+0.029
σ_y	+0.052	+0.064
c_x	+0.015	+0.005
c_y	+0.003	+0.001
f_x	-0.694	-0.108
f_y	-0.183	-0.187
ϕ_x	-1.002	-0.413
ϕ_y	-0.892	-0.210

Table 7.2: Difference between MCMC and optimizer errors for Synthetic-prior dataset (MCMC − Optimizer). Negative values indicate better MCMC performance.

7.1.2 Practical trade-offs between methods

It is worth noting that the evaluation of the best estimation method can take more factors into consideration than just how large errors the methods tend to make. Such factors are summarized in Table 7.3 below.

Table 7.3: Additional factors to consider when choosing between optimization and MCMC.

Factor	**Optimizer (BFGS + multi-start)**	**MCMC (Parallel Tempering)**
Computational cost	Fast per run, but total cost depends on how many starting points are needed to avoid local minima or divergence.	Inherently heavy: thousands of iterations, many chains, and swap steps. Scaling is significantly worse.
Convergence/exactness	With little/no noise the optimizer can come very close to the exact minimum.	The posterior mean does not necessarily converge to the true parameter values.
Type of output	Produces a single point estimate. No direct uncertainty information.	Produces full posterior distributions, giving uncertainty and correlations. Useful when decisions depend on risk or confidence level.
Robustness to noise	Can be sensitive to noise; the gradient signal becomes unreliable.	Noise is naturally included in the likelihood model.
Sensitivity to priors	No priors used; purely data-driven. Good if you have no reliable prior info.	Strongly influenced by priors. Very good when priors reflect reality, but risks bias if they do not.
Performance on multimodal / non-convex problems	Not designed for strong non-convexity. May converge to the wrong mode unless many restarts are used.	Handles multimodality; parallel tempering can explore separate modes.
Real-time usability	Practical for quick parameter estimates.	Slow unless it is run on a good GPU.
Interpretability of output	Simple: one estimate per parameter.	Rich: posterior means, variances, credible intervals, and joint structure. Useful when diagnosing ambiguous cases.
Implementation complexity	Relatively straightforward.	Much more complex: requires a lot of tuning.

7.2 Implications for ESS

7.2.1 Choice of estimation method

When deciding which estimation method to implement in practice at ESS for beam diagnostics, all Tables 6.1–6.3 must be taken into consideration. Because of this, it is not possible to give a definitive answer about which method is best. However, the following guidelines seem reasonable:

- Both methods produce accurate estimates for the parameters that are theoretically possible to estimate well. Hence, based on estimation errors alone, both methods are good choices.

- For the easiest path forward (where minimal additional work is needed for implementation) choose the optimizer and run it from as many initial points as possible for maximum confidence in the estimates. This is likely sufficient for an application like tuning, where wrong estimates are not critical.
- If the optimizer for some reason does not perform sufficiently well in practice on real images (for example due to higher noise than in simulations, or other deviations from the beam model), then consider MCMC for more robustness. Some additional work will be needed to find the right MCMC settings; it is probably best to implement an algorithm for adaptive tuning (such algorithms exist).
- Finally, if both methods seem interesting and someone is willing to do the work, consider implementing both and using them to obtain two independent estimates of the parameters. The methods could then be evaluated and compared further as real image data starts to be generated at ESS from this year and onward.

Additional note. Both methods benefit greatly from being run on a powerful GPU. If no GPU is used, the computation times become too long for practical use. Apart from GPU acceleration, other image processing techniques such as large binning and using a minimal region of interest are also worth considering.

7.2.2 Usefulness in ESS operations

If parameter estimation based on images from the IMG system can be made sufficiently fast and robust, it could support several aspects of ESS operations. Two applications stand out as the most immediately useful:

- **Beam tuning.** When tuning the quadrupoles to produce images with the desired underlying parameters, the estimation algorithms can show clearly what the current image's parameters are. This makes it clearer what needs adjusting at each step of tuning, and can guide the whole process.
- **Heat-load analysis on the target.** Given an estimated parameter vector $\mathbf{k}$, the transient and stationary temperature distribution on the target can be computed using the heat-equation model and numerical solver in Hansen and Thomas (2025). This makes it possible to estimate the thermal stress imposed by each beam pulse, which in turn can guide decisions about safe operating regions and expected proton beam window (PBW) lifetime.

Beyond these primary applications, the estimated parameters could support several additional tasks that may become relevant at ESS further down the line:

- **Detection of unusual or unsafe beam conditions.** From the heat-load analysis one may find that certain combinations of parameters, say too small A_x and A_y, can be dan-

gerous. Hence, if it is possible to do fast estimations one could use those in decisions regarding stopping the beam.

- **Long-term statistical monitoring.** By storing parameter estimates over time, long-term trends may be discovered. This could help in understanding how components in the raster or beamline drift or fail.
- **Cross-validation with other diagnostics.** IMG-based parameter estimates could be compared or used together with data from independent systems, in particular the GRID, to give more certainty in diagnostics.

7.3 Future work

This thesis touches on many interesting topics and has potential to branch off in many directions. Below we concentrate on the ones most relevant for ESS.

7.3.1 Improvements of current estimation methods

A reasonable next step to build upon the work in this thesis would be to try to improve the optimizer and the MCMC as much as possible in terms of speed and estimation accuracy. This can be done in several ways. For the optimizer one may for example try different update rules than quasi-Newton, different strategies for choosing initial points, and using regularization terms for some of the parameters to incorporate prior information. For the MCMC one may try algorithms for adaptive tuning to make sure its settings are always suited to the particular image being observed (R. and Jenkins 2025). It is also common to make use of the gradient and/or Hessian in the proposal function, which may lead to improvements if one is able to find a way to compute these faster than currently (the derivatives require many numerical integrations, as can be seen in Section 4.2.1).

7.3.2 Adapting to real images: signal processing and further modeling

This thesis has mainly demonstrated the usefulness of some estimation algorithms on simulated data, which naturally has its limitations. To make them perform well on real images they need to be tested more on such images. In this process one may discover that details are missing in the model (e.g. the noise behaves differently or the shape of the beam is skewed), or there are image distortions that need to be taken care of before applying the algorithms. Hence both further modeling work and image processing may be needed.

7.3.3 Using neural networks

As has been stated before in this thesis, although gradient-based optimization and MCMC are two of the most established estimation methods, they are not the only ones. Increasingly, people are making use of *neural networks* in similar problems. Below, we list two examples of how this can be done, and how it relates to our current methods.

- **Neural posterior estimation**. A neural network is trained on simulated data to learn an approximation of the posterior distribution. In other words it performs the same task as an MCMC, with the advantage that it can produce estimations nearly instantaneously once it is trained. A successful example in a similar image-based application can be found in Barret and Dupourqué (2024).
- **Neural networks as learned regularization**. A neural network can be used to represent prior information by adding a learned regularization term to the loss function. This could clearly be relevant for our real beam images, given the results in Chapter 6. A general discussion of regularization for inverse problems can be found in Arridge et al. (2019).

7.4 Final conclusions

We finish by tying back directly to the objectives set out in Section 1.2.

(a) Beam properties that can be inferred. Regarding the beam properties that can be inferred we can conclude from Tables 6.1 and 6.2 that all parameters can be inferred from short enough pulses, while for longer pulses all parameters except frequencies and phases can be inferred. This empirical result is also supported by the theoretical result from Section 3.4 that the limiting distribution as time goes to infinity is actually independent of frequency and phase, due to the ergodicity and measure-preserving properties of the triangle raster.

(b) Methods to extract the beam properties. In this thesis we have shown that at least two numerical methods, namely gradient-based optimization and Markov Chain Monte Carlo, can be used to extract the beam properties of interest, that is the parameters of the beam shape and the raster motion (with some limitations regarding frequency and phase). The fact that it is possible is demonstrated by the small estimation errors in for example Tables 4.7 and 5.8. In Section 7.3.3 we also gave examples of other methods that have been used successfully for similar problems.

(c) Analysis, comparison, and limitations of the methods. In Section 7.1 we see that a design of test sets can quantify how well the methods perform in terms of estimation errors. In the same section we also show how other factors can be taken into account: both theoretical knowledge about the methods and knowledge related to the practical implementation of them

in code. While the test sets have their limitations (they are for example rather small), they give a strong indication of which methods show promise; in this case both of them. The Table 7.3 can provide guidance in which method(s) to pursue further in situations like this where both of them work rather well on simulated data.

In summary, the results from this thesis suggest that optical imaging, combined with appropriate estimation methods, can reliably recover the parameters of the rastered proton beam. They also show that clear paths forward exist towards refining the methods and eventually integrating them into the ESS systems for real-world use.

Bibliography

Adli, E. et al. (2022). "Progress of the ESS Proton Beam Imaging Systems". In: *Proc. of the 31st Int. Linear Accelerator Conference (LINAC2022)*. JACoW Publishing, pp. 394–397. DOI: 10.18429/JACoW-LINAC2022-TUPOJO20. URL: https://inspirehep.net/files/b9b10f4e66d15e7934b6fb08d470c704.

Apostol, B. F. (2018). "The inverse problem in Seismology: Seismic moment and energy of earthquakes. Seismic hyperbola". In: arXiv: 1808.03049. URL: https://arxiv.org/abs/1808.03049.

Arridge, S. et al. (2019). "Solving inverse problems using data-driven models". In: *Acta Numerica* 28, pp. 1–174. DOI: 10.1017/S0962492919000059.

Asparouhov, T. and B. Muthén (2025). *Using the Effective Sample Size (ESS) to Monitor Convergence and Quality of Results with Bayesian Estimation*. Tech. rep. Mplus Technical Report. URL: https://www.statmodel.com/download/ESS.pdf.

Aster, R. C., B. Borchers, and C. H. Thurber (2018). *Parameter Estimation and Inverse Problems*. 3rd. Academic Press. ISBN: 9780128046515.

Barret, D. and S. Dupourqué (2024). "Simulation-Based Inference with Neural Posterior Estimation applied to X-ray spectral fitting: Demonstration of working principles down to the Poisson regime". In: arXiv: 2401.06061. URL: https://arxiv.org/abs/2401.06061.

Baudart, G., L. Mandel, and R. Tekin (2022). "JAX based parallel inference for reactive probabilistic programming". In: *Proceedings of the 23rd ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '22)*, pp. 26–36. DOI: 10.1145/3519941.3535066. URL: https://guillaume.baudart.eu/papers/lctes22.pdf.

Böiers, L.-C. (2010). *Mathematical Methods of Optimization*. Lund, Sweden: Studentlitteratur AB. ISBN: 978-91-44-07075-9.

Borukhov, V. T. and G. M. Zayats (2024). "The inverse problem of heat conduction in the case of non-uniqueness: A functional identification approach". In: *Journal of Inverse and Ill-Posed Problems* 32.5, pp. 891–902. DOI: 10.1515/jiip-2022-0056.

Bradbury, J. et al. (2018). *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. URL: http://github.com/jax-ml/jax.

Chao, J., E. S. Ward, and R. J. Ober (2016). "Fisher information theory for parameter estimation in single molecule microscopy: tutorial". In: *Journal of the Optical Society of America A* 33.7, B36–B57.

Church, B. V. (2021). *Diophantine Approximation and Transcendence Theory*. https://web.stanford.edu/~bvchurch/assets/files/talks/Diophantine.pdf. Talk notes, Stanford University.

Dajani, K. and C. Kraaikamp (2009). *Lecture Notes on Ergodic Theory*. Lecture notes, Utrecht University. Accessed: 2025-12-15. URL: https://webspace.science.uu.nl/~kraai101/lecturenotes2009.pdf.

Fawzi, H. (2023). *Smoothness and Strong Convexity*. Lecture notes, Topics in Convex Optimisation, DAMTP, University of Cambridge. Available at https://www.damtp.cam.ac.uk/user/hf323/L23-III-OPT/lecture3.pdf, accessed 2025-12-09.

Griewank, A. and A. Walther (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. 2nd. SIAM. ISBN: 978-0-89871-776-1.

Häggström, O. (2002). *Finite Markov Chains and Algorithmic Applications*. Vol. 52. London Mathematical Society Student Texts. Cambridge, UK: Cambridge University Press. ISBN: 9780521809398.

Hanada, M. and S. Matsuura (2022). *MCMC from Scratch: A Practical Introduction to Markov Chain Monte Carlo*. Springer Nature Singapore. ISBN: 978-981-19-2714-0. DOI: 10.1007/978-981-19-2715-7.

Hansen, E. and C. Thomas (2025). "Thermal modelling of the ESS proton beam window". Manuscript submitted for publication.

Kaipio, J. and E. Somersalo (2005). *Statistical and Computational Inverse Problems*. Springer.

Mandel, L. and E. Wolf (1995). *Optical Coherence and Quantum Optics*. reprint 2013. Cambridge University Press. ISBN: 9780521417112.

National Research Council (1996). "Mathematics and Physics of Emerging Biomedical Imaging". In: *Mathematics and Physics of Emerging Biomedical Imaging*. Chapter on inverse problems in medical imaging. National Academies Press. URL: https://www.ncbi.nlm.nih.gov/books/NBK232488/.

Neuhaus, T. and D. Kadau (2009). "Influence of replica-exchange scheme on the efficiency of parallel tempering simulations". In: *Chemical Physics Letters* 476.4–6, pp. 193–197. DOI: 10.1016/j.cplett.2009.06.018.

Nocedal, J. and S. J. Wright (2006). *Numerical Optimization*. 2nd. Springer Series in Operations Research and Financial Engineering. Springer. DOI: 10.1007/978-0-387-40065-5.

Preston, S. P. et al. (2025). "Think before you fit: Parameter identifiability, sensitivity and uncertainty in systems biology models". In: University of Nottingham, University of Sheffield, University of Warwick. arXiv: 2508.18853 [stat.ME]. URL: https://arxiv.org/abs/2508.18853.

R., P. A. Peña and J. S. Jenkins (Sept. 2025). "Closing the Evidence Gap: reddemcee, a Fast Adaptive Parallel Tempering Sampler". In: arXiv: 2509.24870 [astro-ph.IM]. URL: https://arxiv.org/abs/2509.24870.

Rego, B. V. et al. (2021). "Uncertainty quantification in subject-specific estimation of local vessel mechanical properties". In: *International Journal for Numerical Methods in Biomedical Engineering* 37.12, e3535. DOI: 10.1002/cnm.3535.

Sarig, O. (2009). *Lecture Notes on Ergodic Theory*. https://www.weizmann.ac.il/math/sarigo/sites/math.sarigo/files/uploads/ergodicnotes.pdf. Lecture notes, Weizmann Institute of Science.

Sidford, A. (2019). *Chapter 3: Convexity*. https://web.archive.org/web/20230000000000/https://web.stanford.edu/~sidford/. Lecture notes for MS&E 213 / CS 269O.

Wen, H. et al. (2025). "JANC: A cost-effective, differentiable compressible reacting flow solver featured with JAX-based adaptive mesh refinement". In: arXiv: 2504.13750. URL: https://arxiv.org/abs/2504.13750.

Appendix A

Test set construction

In both synthetic test sets, parameter vectors $\mathbf{k}$ are drawn from a probability distribution $p(\mathbf{k})$ supported on the admissible parameter domain Ω. For the Synthetic–uniform test set we let $p(\mathbf{k}) = \mathcal{U}(\Omega)$, i.e. all parameters are drawn uniformly from their allowed intervals. For the Synthetic–prior test set we instead use a Gaussian distribution $p(\mathbf{k}) = \mathcal{N}(\mathbf{k}_{\text{nom}}, \Sigma)$, where $\Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_{10}^2)$ with $\sigma_i = 0.2\,|k_{\text{nom},i}|$. Samples drawn outside Ω are discarded by rejection sampling. Below we define the procedure used to generate the test sets.

Procedure 1 Construction of a synthetic test set

Require: Parameter bounds Ω, pulse-length interval $[T_{\min}, T_{\max}]$, number of samples N, parameter distribution $p(\mathbf{k})$, noise scale $s > 0$

1: Define pulse lengths $\{T_i\}_{i=1}^{N}$ as logarithmically spaced values in $[T_{\min}, T_{\max}]$
2: **for** $i = 1, \ldots, N$ **do**
3: Draw $\mathbf{k}_i \sim p(\mathbf{k})$
4: Generate clean image $I_i = \mathcal{F}(\mathbf{k}_i, T_i)$
5: Add Poisson noise $\tilde{I}_i = \mathcal{P}(I_i; s)$
6: Normalize $\tilde{I}_i \leftarrow \tilde{I}_i / s$
7: Store $(\tilde{I}_i, \mathbf{k}_i, T_i)$
8: **end for**

Appendix B

Ergodic theory terminology

Here we define some of the fundamental concepts in ergodic theory that are used in Section 3.4.

Definition B.1 (Borel σ-algebra)**.** The Borel σ-algebra on $\mathbb{R}^n$ is the smallest σ-algebra containing all open subsets of $\mathbb{R}^n$.

A measure is a function that assigns a nonnegative "size" to subsets of a set X.

Definition B.2 (Measure)**.** A measure on a σ-algebra Σ is a function $\mu : \Sigma \to [0, \infty]$ satisfying:

1. *Empty set:* $\mu(\varnothing) = 0$.
2. *Non-negativity:* For all $E \in \Sigma$, $\mu(E) \geq 0$.
3. *Countable additivity:* For any countable collection of pairwise disjoint sets $\{E_k\}_{k=1}^{\infty} \subset \Sigma$,

$$\mu\left(\bigcup_{k=1}^{\infty} E_k\right) = \sum_{k=1}^{\infty} \mu(E_k).$$

In this thesis, μ is always the normalized Lebesgue measure on the raster domain X, so that $\mu(X) = 1$. Intuitively, the Lebesgue measure corresponds to the usual notion of area in $\mathbb{R}^2$.

Definition B.3 (Lebesgue measure)**.** The Lebesgue measure is the unique extension of the notion of area of rectangles to the Borel σ-algebra.

Definition B.4 (Measure-preserving map)**.** Let (X, Σ, μ) be a measure space. A measurable transformation $T : X \to X$ is called *measure preserving* if

$$\mu\big(T^{-1}(A)\big) = \mu(A) \qquad \text{for every } A \in \Sigma.$$

That is, when T is applied to a set it does not change its size.

The definition of ergodicity uses the notion of a *probability space*, which is a measure space in which the total measure of the whole set X is equal to one. Put simply, ergodicity of a

transformation T means that all invariant sets are either "everything" (sets of full measure) or "nothing" (sets of measure zero).

Definition B.5 (Ergodicity)**.** Let (X, Σ, μ) be a probability space and let $T : X \to X$ be a measure-preserving transformation. The map T is called *ergodic* if every measurable set $A \in \Sigma$ satisfying

$$T^{-1}(A) = A$$

has either $\mu(A) = 0$ or $\mu(A) = 1$.

Appendix C

Additional notes for BFGS convergence proof

C.1 Angle interpretation

For completeness we recall the Cauchy–Schwarz inequality.

Theorem C.1 (Cauchy–Schwarz)**.** *For any vectors $u, v \in \mathbb{R}^n$,*

$$|\langle u, v\rangle| \;\le\; \|u\|\,\|v\|,$$

with equality if and only if u and v are linearly dependent.

This result motivates the definition of the angle θ between two vectors u and v via

$$\cos\theta \;=\; \frac{\langle u, v\rangle}{\|u\|\,\|v\|}.$$

In the BFGS convergence analysis we set

$$u = \nabla\mathcal{L}_k, \qquad v = B_k^{-1}\nabla\mathcal{L}_k,$$

so that equation (4.9) is exactly this expression for $\cos\theta$.

C.2 Strong convexity lemma; convergence implication

To prove that given $\liminf_{n\to\infty} \|\nabla\mathcal{L}_n\| \to 0$ and $\mathcal{L}$ strongly convex, we have $\mathbf{k}_n \to \mathbf{k}^*$, we can use a standard inequality for strongly convex functions from Sidford (2019). We state the inequality as a lemma below, using the same notation as in Section 4.1.3.

Lemma C.1. *Assume that $\mathcal{L}$ is m–strongly convex, meaning that it is strongly convex with the same $m > 0$ as in Assumptions 4.1. Then for all $\mathbf{k} \in S$,*

$$\frac{m}{2}\|\mathbf{k} - \mathbf{k}^*\|^2 \;\leq\; \mathcal{L}(\mathbf{k}) - \mathcal{L}(\mathbf{k}^*) \;\leq\; \frac{1}{2m}\|\nabla\mathcal{L}(\mathbf{k})\|^2.$$

We state the convergence result we want to prove as a proposition.

Proposition C.1. *Let $\mathcal{L}$ be strongly convex on the level set S, and let $\mathbf{k}^*$ denote its unique minimizer. Suppose $\{\mathbf{k}_n\}$ is a sequence such that*

$$\liminf_{n\to\infty} \|\nabla\mathcal{L}(\mathbf{k}_n)\| = 0.$$

Then the full sequence converges:

$$\mathbf{k}_n \to \mathbf{k}^*.$$

Proof. Convergence of the infimum of the gradient to zero, $\liminf_{n\to\infty} \|\nabla\mathcal{L}(\mathbf{k}_n)\| = 0$, means that there exists a subsequence $\mathbf{k}_{n_j}$ such that $\|\nabla\mathcal{L}(\mathbf{k}_{n_j})\| \to 0$ as $j \to \infty$. By Lemma C.1 this implies that $\|\mathbf{k}_{n_j} - \mathbf{k}^*\| \to 0$ as $j \to \infty$, or equivalently $\mathbf{k}_{n_j} \to \mathbf{k}^*$.

Now we want to show that it follows that the whole sequence $\mathbf{k}_n$ must also converge to $\mathbf{k}^*$. The first step is to show that $\mathcal{L}(\mathbf{k}_n)$ converges to $\mathcal{L}(\mathbf{k}^*)$; more precisely, that for each $\varepsilon > 0$ there exists an $N > 0$ such that

$$|\mathcal{L}(\mathbf{k}_n) - \mathcal{L}(\mathbf{k}^*)| < \varepsilon \qquad \text{for all } n \geq N.$$

To do this, assume an arbitrary $\varepsilon > 0$. Note that the Wolfe conditions (4.5)–(4.6) for a convex problem ensure that the algorithm produces a decreasing sequence, i.e. $\mathcal{L}(\mathbf{k}_{n+1}) \leq \mathcal{L}(\mathbf{k}_n)$. This essentially means that by choosing a sufficiently large index J in the subsequence $\{\mathbf{k}_{n_j}\}$, we can get close enough to the optimum $\mathcal{L}(\mathbf{k}^*)$ not only for the subsequence but also for the full sequence. Defining $\mathbf{k}_{n_J} = \mathbf{k}_N$, we obtain:

$$\mathcal{L}(\mathbf{k}_n) - \mathcal{L}(\mathbf{k}^*) \;<\; \mathcal{L}(\mathbf{k}_{n_J}) - \mathcal{L}(\mathbf{k}^*) \;<\; \varepsilon \qquad \text{for all } n \geq N.$$

Because $\varepsilon > 0$ was arbitrary, this shows that $\mathcal{L}(\mathbf{k}_n) \to \mathcal{L}(\mathbf{k}^*)$.

The second and final step is to make use of Lemma C.1 again to relate convergence of $\mathcal{L}(\mathbf{k}_n)$ to convergence of $\mathbf{k}_n$. From the lower bound, we get

$$\frac{m}{2}\|\mathbf{k}_n - \mathbf{k}^*\|^2 \;\leq\; \mathcal{L}(\mathbf{k}_n) - \mathcal{L}(\mathbf{k}^*),$$

and therefore conclude that $\mathbf{k}_n \to \mathbf{k}^*$. This completes the proof. □

Appendix D

Markov chain Monte Carlo

D.1 Additional figures for MCMC tuning

This appendix contains all diagnostic figures used during the tuning of the parallel–tempering MCMC algorithm, including trace plots, autocorrelation and IACT curves, effective sample size illustrations, and acceptance/swap–rate profiles across the temperature ladder.

All plots correspond to the nominal synthetic beam image discussed in Section 5.4.

Trace plots

The full trace plots for all ten beam parameters are shown in Figure D.1. These plots illustrate the evolution of each parameter throughout the sampling process after burn-in.

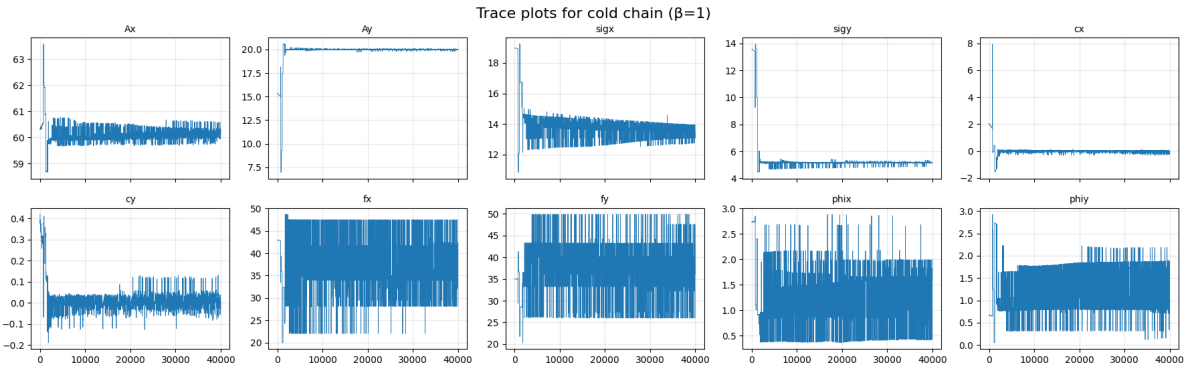

Figure D.1: Full trace plots for the cold chain ($\beta = 1$) after burn-in.

Autocorrelation and integrated autocorrelation time

Figures D.2 shows an example of the autocorrelation function and integrated autocorrelation time for one parameter.

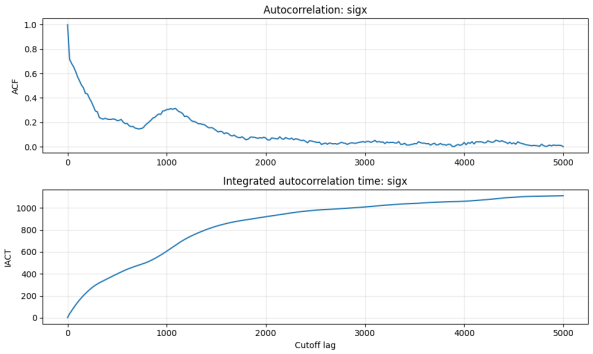

Figure D.2: Autocorrelation function (top) and IACT (bottom) for σ_x.

Per-chain acceptance rates

Figure D.3 shows the Metropolis–Hastings acceptance rates for each chain in the parallel–tempering ladder.

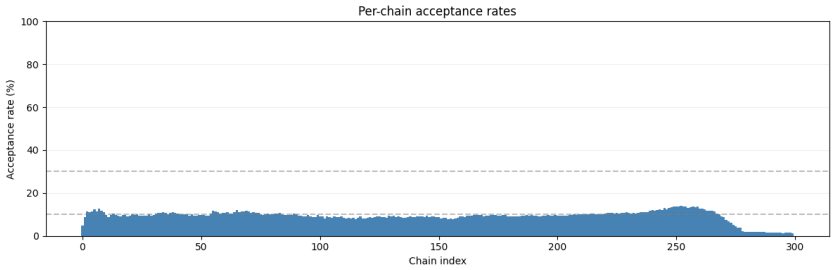

Figure D.3: Acceptance rates across the temperature ladder.

Swap rates between adjacent chains

Figure D.4 presents the swap acceptance rates for all adjacent pairs of chains in the temperature ladder.

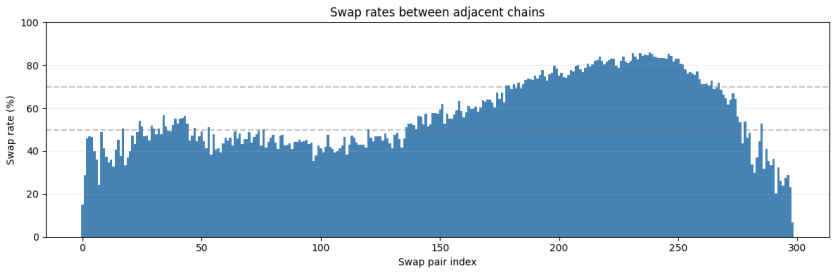

Figure D.4: Swap rates between adjacent temperatures in the PT ladder.

This section lists the full configuration used in all parallel–tempering MCMC experiments presented in Chapter 5. The same settings are used for both nominal parameter images corresponding to pulse lengths $T = 50\,\mu$s and $T = 0.5\,$ms.

Temperature ladder

A total of 300 replicas are used, with inverse temperatures $\beta \in [1.0, 0.0025]$. The ladder is spaced geometrically,

$$\beta_i = \beta_{\max}\, r^{\,i}, \qquad i = 0, \ldots, 300,$$

where the ratio r is chosen such that $\beta_{299} = 0.0025$.

Sampling schedule

Each chain runs for a total of 50,000 iterations. The first 10,000 iterations (where we do no swaps) are discarded as burn–in, while the remaining 40,000 samples are retained for posterior estimation.

Swaps between adjacent temperature levels are attempted every 10th iteration, resulting in 4,000 swap attempts throughout the run. All swaps use the standard symmetric proposal

$$\alpha = \min\bigl(1,\ \exp\bigl[(\beta_i - \beta_{i+1})\bigl(\log \pi(\theta_{i+1}) - \log \pi(\theta_i)\bigr)\bigr]\bigr)\,,$$

where $\pi(\theta)$ denotes the (unnormalized) posterior distribution. The swapping scheme that is used is called deterministic even–odd swapping.

Proposal distribution

Within each chain, a Gaussian random–walk proposal is used:

$$\theta' = \theta + \eta, \qquad \eta \sim \mathcal{N}(0,\, \sigma^2 I).$$

The proposal standard deviations σ were tuned according to the procedure described in Section 5.4.

Prior distribution

In this thesis, two kinds of priors are considered: uniform and Gaussian. The uniform priors are defined over the parameter bounds specified in Section 2.2. The Gaussian priors are defined for each parameter via a mean value and a standard deviation.

Initial conditions

Each chain is initialized by sampling independently from the prior. When Gaussian priors are used, rejection sampling is used whenever a value falls outside the allowed parameter bounds.

D.2 Markov chain terminology

Definition D.1 (Irreducible chain)**.** Consider a Markov chain with finite state space $S = \{s_1, \ldots, s_k\}$ and transition matrix P. We say the chain is *irreducible* if every state can eventually reach every other state. Formally, for any pair of states $s_i, s_j \in S$ there exists an integer $n \geq 1$ such that $(P^n)_{i,j} > 0$.

We next define what an aperiodic chain means. Intuitively it means that is has no deterministic cycle structure in how it returns back to its state, e.g. that it can only return in 2 or 4 steps.

Definition D.2 (Aperiodic chain)**.** For a state $s_i \in S$, define its *period* as

$$d(s_i) = \gcd\{\, n \geq 1 : (P^n)_{i,i} > 0 \,\}.$$

A state is called *aperiodic* if $d(s_i) = 1$. A Markov chain is said to be aperiodic if every state in the chain is aperiodic.

Below we state the definition of *total variation*, which is a way to measure the distance between two probability distributions.

Definition D.3 (Total variation distance)**.** Let μ and ν be probability distributions on a finite state space $S = \{s_1, \ldots, s_k\}$. The *total variation distance* between μ and ν is then defined as

$$d_{\mathrm{TV}}(\mu, \nu) \;=\; \frac{1}{2} \sum_{i=1}^{k} \big|\mu(s_i) - \nu(s_i)\big|.$$

Master's Theses in Mathematical Sciences 2026:E12
ISSN 1404-6342

LUTFNA-3054-2026

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/