Scalable block Gibbs sampling for image deblurring in X-ray radiography

by

Jesse Adams

A Dissertation Submitted to the Faculty of the

GRADUATE INTERDISCIPLINARY
PROGRAM IN APPLIED MATHEMATICS

In Partial Fullfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2019

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

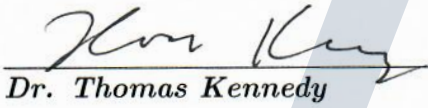
As members of the Dissertation Committee, we certify that we have read the dissertation prepared by **Jesse Adams**, titled **Scalable block Gibbs sampling for image deblurring in X-ray radiography** and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Date: ***April 15, 2019***

Dr. Matthias Morzfeld

Date: ***April 15, 2019***

Dr. Aaron Luttman



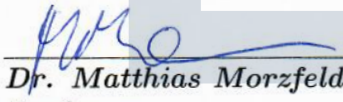
Date: ***April 15, 2019***

Dr. Thomas Kennedy

Date: ***April 15, 2019***

Dr. Kevin Lin

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.
I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.



Date: ***April 15, 2019***

Dr. Matthias Morzfeld
Professor
Mathematics

Date: ***April 15, 2019***

Dr. Aaron Luttman
Program Manager
Nevada National Security Site

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of the requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that an accurate acknowledgement of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Jesse John William Adams

Acknowledgements

This work would not have been possible without the patience, effort, and guidance of my advisors Dr. Matthias Morzfeld and Dr. Aaron Luttman. They believed in me when I didn't, and pushed me when I needed it. From mountain biking excursions with Matti and dessert hiking ventures with Aaron, they lead me through this dissertation both on the trail and in the office. Their commitment and diligence week after week served as a great example of what it means to be a successful mathematician, and

I must also thank Dr. Marylesa Howard, Dr. Kevin Joyce, and Dr. Maggie Lund from the Nevada National Security Site. I would likely pursued an entirely different research path if not for Marylesa, who introduced me to Las Vegas via a summer internship. Kevin not only provided insightful comments on this work, but also served as an irreplaceable friend, coworker, and roommate over the years. Maggie was there for me during some of my hardest times, and with me for some of my best.

I am grateful for all of the professors I have had the opportunity to learn from and work with at the University of Arizona. Dr. Kevin Lin, Dr. Thomas Kennedy, and Dr. Shankar Venkataramani all deserve special thanks for serving on one or more of my dissertation and comprehensive exam committees. Their insightful questions and thoughtful input greatly improved the work presented here.

My friends and family deserve the most praise for their help on this journey. Those I want to give special thanks to include Kiralyse Nissman, Victoria Gershuny, Adam Martinez, and Dr. Brian Hong, without whose support and camaraderie I may not have survived all these years at Arizona. Spence Lunderman and Kyle Gwirtz ensured our office had good coffee and better conversation. Adam and Kiana Martinez provided immeasurable support, from last minute dog-sitting to late night conversations. Kin Luu and Matt Grossman made sure I was well fed during my most stressful weeks. Dr. Nick Henscheid not only helped me momentarily get away from math by dangling me off cliffsides on ropes, but was a wonderful friend and mentor. My parents, Jim and Tami Adams, and my brother Cody Adams shaped who I am today. Their unconditional love and acceptance were an indispensable ingredient in my success.

Finally, I am grateful to have been supported by a Site Directed Research and Development grant from the Nevada National Security Site and the Applied Mathematics Program at the University of Arizona, which made much of this work possible.

This manuscript has been authored by Mission Support and Test Services LLC, under Contract No. DE-NA0003624 with the U.S. Department of Energy and supported by the Site-Directed Research and Development Program, National Nuclear Security Administration, Office of Defense Nuclear Nonproliferation Research and Development.

DOE/NV/03624--0482

Dedication

To my parents, Jim and Tami, and my brother Cody.

Contents

List of Figures

List of Tables

List of Algorithms

Notations

Images

$\mathbf{B} \in \mathbb{R}^{m_b \times n_b}$ ..... Image data
$\mathbf{X} \in \mathbb{R}^{m_x \times n_x}$ ..... Image reconstruction
$\mathbf{b} = \mathbf{B}(:) \in \mathbb{R}^{M}$ ..... Column stack of $\mathbf{B}$
$\mathbf{x} = \mathbf{X}(:) \in \mathbb{R}^{N}$ ..... Column stack of $\mathbf{X}$
$\mathbf{X}_i \in \mathbb{R}^{m_d \times n_d}$ ..... Sub-image i
$\mathbf{x}_i = \mathbf{X}_i(:) \in \mathbb{R}^{m_d n_d}$ ..... Column stack of $\mathbf{X}_i$

Model Components

$\mathbf{a} \in \mathbb{R}^{m_a \times n_a}$ ..... Convolution kernel
$\lambda \in \mathbb{R}_{>0}$ ..... Likelihood precision
$\delta \in \mathbb{R}_{>0}$ ..... Prior precision
$\boldsymbol{\varepsilon} \in \mathbb{R}^{M}$ ..... Additive Gaussian noise
$\widehat{\mathbf{A}} \in \mathbb{R}^{N \times N}$ ..... Convolution matrix with classical boundary conditions
$\mathbf{D} \in \mathbb{R}^{M \times N}$ ..... Cropping matrix
$\mathbf{A} = \mathbf{D}\widehat{\mathbf{A}} \in \mathbb{R}^{M \times N}$ ..... Convolution matrix
$\mathbf{L} \in \mathbb{R}^{N \times N}$ ..... Prior precision matrix
$\mathbf{H} = \lambda \mathbf{A}^\top \mathbf{A} + \delta \mathbf{L} \in \mathbb{R}^{N \times N}$ ..... Posterior precision matrix
$\mathbf{m} = \lambda \mathbf{H}^{-1} \mathbf{A}^\top \mathbf{b} \in \mathbb{R}^{N}$ ..... Posterior mean
$\mathbf{A}_{:,i} \in \mathbb{R}^{M \times m_d n_d}$ ..... Block column of $\mathbf{A}$
$\mathbf{H}_{ij} = \lambda \left(\mathbf{A}_{:,i}\right)^\top \mathbf{A}_{:,j} + \delta \mathbf{L}_{ij} \in \mathbb{R}^{m_d n_d \times m_d n_d}$ ..... Sub-matrix of $\mathbf{H}$

Dimensions

m_a ..... First dimension of the kernel $\mathbf{a}$
n_a ..... Second dimension of the kernel $\mathbf{a}$
m_b ..... First dimension of the data $\mathbf{B}$
n_b ..... Second dimension of the data $\mathbf{B}$
$m_x = m_b + m_a - 1 = m_d \cdot m_B$ ..... First dimension of the reconstruction $\mathbf{X}$
$n_x = n_b + n_a - 1 = n_d \cdot n_B$ ..... Second dimension of the reconstruction $\mathbf{X}$
$M = m_b \cdot n_b$ ..... Number of pixels in $\mathbf{B}$
$N = m_x \cdot n_x$ ..... Number of pixels in $\mathbf{X}$
m_d ..... First dimension of a sub-image $\mathbf{X}_i$
n_d ..... Second dimension of a sub-image $\mathbf{X}_i$
m_B ..... Number of sub-images in first dimension

n_B ..... Number of sub-images in second dimension

Probability

$p(\cdot)$ ..... Probability density
$p(\cdot|\cdot)$ ..... Conditional probability density
$x \sim p$ ..... Random variable x has distribution p

Sampling

N_e ..... Sample size
N_{eff} ..... Effective sample size
τ_{int} ..... Integrated autocorrelation time

Math

$\mathbb{R}$ ..... Real numbers
$||\cdot||$ ..... Euclidean Norm
$\top$ ..... Transpose
$\odot$ ..... Element-wise multiplication
$\otimes$ ..... Kronecker product

Abstract

Quantitative image analysis in the security sciences formulates an image deblurring problem as a Bayesian inverse problem to reduce and quantify noise and blur. We consider images of size 16 megapixels and, since each pixel represents an unknown, the dimension of the Bayesian inverse problem is on the order of 10^7. The large dimension poses numerical and computational difficulties for two reasons. First, Markov chain Monte Carlo (MCMC), typically used to solve a Bayesian inverse problem, is generally slow to converge in high dimensions. Second, even generating one step in a Markov chain is challenging at this size. We present a Gibbs sampler that is scalable to the large dimension required in the security sciences and its scalability is achieved in two steps. We (i) accelerate MCMC convergence by exploring banded structure in the posterior precision matrix; and (ii) use a matrix-free implementation, because constructing and storing even sparse matrices is infeasible in our target application.

Chapter 1

Introduction

Imaging is commonly used as a qualitative source of information, where the goal is to produce the best looking image possible. An imaging technique used in the security sciences is pulse-powered X-ray radiography, which, in contrast, is used as a quantitative diagnostic tool for scientific experiments. A radiograph is produced by pulsing a high energy source that emits X-rays through the scene. The scene consists of objects of varying materials and densities which contribute to the attenuation of X-rays, and the X-rays that are not attenuated are absorbed by a scintillator which converts X-rays to visible light. The light intensity from the scintillator is photographed by a Charge-Coupled Device (CCD) within an optical system to produce a radiograph.

The US Department of Energy (DOE) owns several high-energy X-ray imaging systems, including the Dual-Axis Radiographic Hydrodynamic Test Facility (DARHT) at Los Alamos National Laboratory [26], the Flash X-ray machine at the Contained Firing Facility (CFF) of Lawrence Livermore National Laboratory [29], and the Cygnus Dual Beam Radiographic Facility at the Nevada National Security Site (NNSS) [38, 39]. These systems are part of the DOE's stockpile stewardship program and are used to image hydrodynamic materials experiments. The materials studied can be very dense, and the experiments occur over a very short time scale. Thus

high energy X-rays are required, with precise timing and a short pulse to effectively capture radiographic images of the experiments.

The radiographic images produced by these systems are corrupted by noise and blur, which depend on inherent characteristics of the system, such as X-ray scatter, the intensity profile of the X-ray source, and the physical and optical components that produce the radiographs. In order to effectively model the system, both the noise and blur must be taken into account. The blur in the system can be modeled as a convolution of a true image, $\mathbf{X}$, with the system response, $\mathbf{a}$, which is called the convolution kernel. This model assumes spatial invariance, i.e. that the kernel is independent of the location in the image. A discrete convolution model is considered, due to the discrete nature of the data collection. The noise is modeled as additive and Gaussian. This linear model can be written in the form $\mathbf{B} = \mathbf{a} * \mathbf{X} + \boldsymbol{\varepsilon}$, where $\mathbf{B}$ and $\mathbf{X}$ are the data and true images, respectively, $\boldsymbol{\varepsilon}$ is the Gaussian noise, and '$*$' represents discrete convolution. The goal is to reconstruct the true image, $\mathbf{X}$, given the data $\mathbf{B}$ and the kernel $\mathbf{a}$. This process is called deconvolution and is an ill-posed inverse problem [42]. That is, small changes in the image data due to noise have drastic effects on the solution. This is one of the hurdles that needs to be addressed in order to produce accurate reconstructions.

Aside from ill-posedness, several other challenges exist that are the focus of this work. Firstly, the data of interest is produced by Cygnus at the NNSS, and the images captured are large, on the order of 16 megapixels. The dimension of the problem is equal to the number of pixels in the image, resulting in an extremely large matrix in the linear system associated with the linear convolution model. The high dimensionality makes matrix based solutions infeasible. Secondly, in discrete 2D convolution, the value of a pixel in the data $\mathbf{B}$ depends on the values of neighboring pixels in the true image $\mathbf{X}$, determined by the extent of the kernel $\mathbf{a}$. For pixels near the border of the data $\mathbf{B}$, the kernel extends beyond the boundary of the data. Thus

the natural domain for the reconstruction $\mathbf{X}$ is larger than the domain of $\mathbf{B}$. The boundaries must be carefully considered in order to avoid computational artifacts in the reconstructions.

A classical means of performing deconvolution is via Fourier-based methods, which do not require construction of the forward matrix [18]. These methods require assumptions on the boundary – such as periodicity – that are unnatural for many images, and can result in poor reconstructions. In order to account for these boundary effects, an extended boundary on the reconstruction is assumed in this work, as in [1, 5]. Unfortunately, these boundary conditions are incompatible with classical Fourier-based deconvolution methods and also make the inverse problem under-determined. In this work, a deconvolution method is formulated which utilizes a block Gibbs sampler, and careful blocking of the data into sub-images which makes the deconvolution computationally feasible and makes the Gibbs sampler efficient. The matrices required for the sub-images can still be prohibitively large. Functional versions of necessary operators are developed, which do not require explicit construction of the full matrix. Additionally, to overcome the ill-posed and under-determined nature of the inverse problem, a Bayesian approach is taken. Prior assumptions are imposed on the reconstruction $\mathbf{X}$, and samples are generated from the corresponding posterior distribution. This is similar to regularization, and is a commonly used technique [1, 2, 14, 37]. Samples are computed by numerically solving the inverse problem using a Markov chain Monte Carlo (MCMC) method. Specifically, a block Gibbs sampler is constructed that uses sub-images as variable blocks. The sampler computes a deconvolution of each sub-image conditioned on relevant neighbors to form a complete reconstruction at each iteration of the sampler. This blocking scheme takes advantage of structure in the Gaussian posterior precision matrix and the two-dimensional nature of the data.

In this work, we present a matrix-free implementation of deconvolution within a

Bayesian framework, which can effectively produce deblurred samples of large images. The Blocked Gibbs sampler incorporates data-driven boundary conditions as described in [1], and utilizes a blocking scheme that takes advantage of the structure of the posterior precision matrix and the two-dimensional nature of the data. This construction is scalable with image size allowing effective convergence of the MCMC chain, which generally scales poorly with dimension [7, 17, 40]. Both the blurring kernel used and the images reconstructed are larger than those analyzed in the current literature [8, 9, 13, 20, 23, 41, 45], which makes our method applicable to a new space of applications, including radiographs produced at DOE facilities. The method presented also provides uncertainty estimates and results are presented on real data from the Cygnus Dual Beam Radiography Facility at the NNSS.

Chapter 2

Background on discrete convolution, Bayesian modeling, and Gibbs sampling

In this chapter, necessary background information for image deblurring via 2D deconvolution in a Bayesian setting is provided. Discrete convolution is introduced in both 1D and 2D, and different boundary conditions are discussed. Next, the Bayesian formulation of the problem is given and the relevant probability distributions are derived. Finally, the Gibbs sampler is introduced.

2.1 Discrete convolution and boundary conditions

As noted in the introduction, 2D convolution with a known kernel is used to model the blur in the image data. The data obtained from the imaging systems is discrete, which lends itself to a discrete 2D convolution model. The 2D model can be difficult to follow due to all of the necessary indices to consider. In order to introduce the problem structure, 1D convolution is introduced. Considering the 1D convolution provides useful insight into the 2D convolution problem.

2.1.1 1D convolution and classical boundary conditions

Discrete convolution in one dimension can be written as

$$b_i = \sum_{k=-\infty}^{\infty} a_{i-k} x_k \qquad \text{for } i = 1, \ldots, m_b,$$

where the b_i represents the collected data, a_i represents the blurring kernel, and x_i represents the true 1D signal. Here, there are m_b output measurements b_i. The kernel is assumed to be known – either by a model or discrete measurement – and has finite extent. We define a to be non-zero for indices 1 to m_a and, similarly, x is non-zero starting at index 1. Assuming that the kernel $\mathbf{a}$ is smaller than the true signal $\mathbf{x}$, we have

$$b_i = \sum_{k=i-m_a}^{i-1} a_{i-k} x_{k+m_a} = \sum_{k=0}^{m_a-1} a_{m_a-k} x_{k+i} \qquad \text{for } i = 1, \ldots, m_b. \tag{2.1.1}$$

Equation (2.1.1) can be written in matrix notation as

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{m_b} \end{bmatrix} = \begin{bmatrix} a_{m_a} & a_{m_a-1} & \cdots & \cdots & a_2 & a_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & & & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{m_a} & a_{m_a-1} & \cdots & \cdots & a_2 & a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{m_b+m_a-2} \\ x_{m_b+m_a-1} \end{bmatrix}, \tag{2.1.2}$$

or $\mathbf{b} = \mathbf{A}\mathbf{x}$. The matrix notation shows that although $\mathbf{b} \in \mathbb{R}^{m_b}$, the true signal $\mathbf{x} \in \mathbb{R}^{m_x}$, where $m_x = m_b + m_a - 1$, and $\mathbf{A} \in \mathbb{R}^{m_b \times m_x}$. That is, the natural domain of $\mathbf{x}$ is larger than the domain of $\mathbf{b}$, and depends on the size of the kernel. The domain of $\mathbf{b}$ is commonly referred to as the field of view (FOV) in image processing. The FOV corresponds to the center m_b points in the true signal, $\mathbf{x}$. The offset on the left side of $\mathbf{x}$ is $c_m = \lfloor \frac{m_a-1}{2} \rfloor$ indices, and $m_a - c_m$ on the right, where $\lfloor \cdot \rfloor$ represents the floor function. Figure 2.1.1 shows how these two vectors align. The fact that $\mathbf{x}$ necessarily extends beyond the FOV makes the system for the deconvolution problem

$$\begin{array}{ccccccccccccccc} & & & & b_1 & \cdots & & \cdots & & \cdots & b_{m_b} & & & & \\ [& \underbrace{x_1 \quad \cdots \quad x_{c_m}} & | & x_{c_m+1} & \cdots & x_{2c_m} & \cdots & x_{m_b+1} & \cdots & x_{m_b+c_m} & | & \underbrace{x_{m_b+c_m+1} \quad \cdots \quad x_{m_x}} &] \\ & \text{left boundary} & & & & & \text{center } m_b \text{ elements} & & & & & \text{right boundary} & \end{array}$$

Figure 2.1.1: The alignment of the true signal, $\mathbf{x}$, with the data, $\mathbf{b}$, is shown to provide a visual representation of the field of view. (2.1.3) and (2.1.4) define two classical choices for boundary conditions.

under-determined.

The classical means of overcoming this issue is to make assumptions about the boundary of the true signal $\mathbf{x}$, called boundary conditions. The true signal $\mathbf{x}$ is assumed to have the same domain as the data $\mathbf{b}$ and outside the FOV assumptions are enforced on the values of $\mathbf{x}$ based on those within the FOV. Common choices for boundary conditions include periodic, zero, and reflecting boundary conditions [18]. Periodic boundary conditions assume that $\mathbf{x}$ is periodic and thus the signal repeats outside the FOV. Zero (or Dirichlet) boundary conditions assume that $\mathbf{x}$ is zero outside the FOV. Reflecting (or Neumann) boundary conditions assume that $\mathbf{x}$ is reflected over the boundary. These boundary conditions can be described mathematically as

$$(x_1, \cdots, x_{c_m}) = \begin{cases} (x_{m_b+1}, \cdots, x_{m_b+c_m}) & \text{periodic} \\ (0, \cdots, 0) & \text{zero} \\ (x_{2c_m}, \cdots, x_{c_m+1}) & \text{reflecting} \end{cases}, \tag{2.1.3}$$

on the left hand side, and

$$(x_{c_m+m_b+1}, \cdots, x_{m_x}) = \begin{cases} (x_{c_m+1}, \cdots, x_{m_a}) & \text{periodic} \\ (0, \cdots, 0) & \text{zero} \\ (x_{m_b+c_m}, \cdots, x_{m_b+2c_m-m_a}) & \text{reflecting} \end{cases}, \tag{2.1.4}$$

on the right hand side. Visual examples of the BCs for a 1D signal are shown in Figure 2.1.2. The figure illustrates a 1D signal and its extension beyond the FOV,

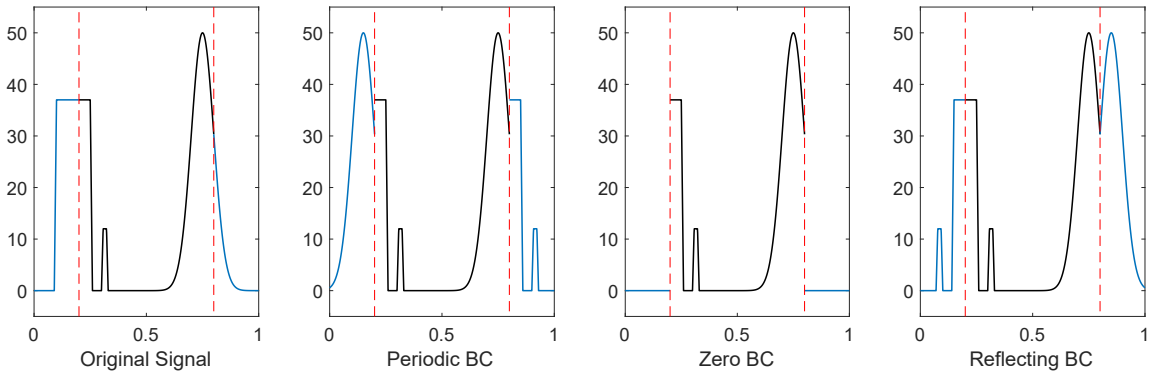


Figure 2.1.2: Boundary conditions for a 1D signal. The vertical dotted red lines denote the domain (FOV) of the measured data. The true signal is shown on the left, and the rest of the graphs depict the assumption of what the true signal looks like based on various boundary conditions: periodic, zero, and reflecting (left to right).

and depicts a comparison to what the signal would look like if it satisfied the assumed boundary conditions. Each of the signals satisfying the boundary condition choices differ considerably outside the FOV from the true signal.

Classical boundary conditions result in a system that is no longer under-determined. The system can be written $\mathbf{b} = \widehat{\mathbf{A}}\tilde{\mathbf{x}}$, where $\widehat{\mathbf{A}}$ is the convolution operator with classical boundary conditions, and $\tilde{\mathbf{x}}$ represents the central m_b elements of the vector $\mathbf{x}$. In this case, $\widehat{\mathbf{A}} \in \mathbb{R}^{m_b \times m_b}$ and the boundary conditions are applied to the vector in the FOV. Each of the above classical boundary conditions is associated with an efficient spectral method for solving the deconvolution problem. For periodic boundary conditions, $\widehat{\mathbf{A}}$ is a circulant matrix, and the system is diagonalizable by the discrete Fourier transform (DFT)[18, 42]. With zero boundary conditions, $\mathbf{A}$ is a Toeplitz matrix, which can be embedded in a circulant matrix and also solved via the DFT [1]. Reflecting boundary conditions result in an $\mathbf{A}$ matrix with Toeplitz-plus-Hankel structure. If the kernel is also symmetric ($(a_1, \cdots, a_{m_a}) = (a_{m_a}, \cdots, a_1)$), then the system is diagonalizable by a discrete cosine transform (DCT) [31].

2.1.2 2D convolution and classical boundary conditions

The 2D case follows a structure similar to that of the 1D case. In general, 2D convolution is written as

$$b_{i,j} = \sum_{\ell=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} a_{i-k,j-\ell} x_{k,\ell} \qquad \text{for } i = 1, \ldots, m_b, \;\; j = 1, \ldots, n_b. \tag{2.1.5}$$

As before, the kernel is assumed to be discrete, $\mathbf{a} \in \mathbb{R}^{m_a \times n_a}$, and the 2D signal – or image – representing the data has a known size $\mathbf{B} \in \mathbb{R}^{m_b \times n_b}$. Then for any pixel in the data, (2.1.5) can be rewritten after re-indexing as

$$b_{i,j} = \sum_{\ell=0}^{n_a-1} \sum_{k=0}^{m_a-1} a_{m_a-k,n_a-\ell} x_{i+k,j+\ell}, \quad \text{for } i = 1, \ldots, m_b, j = 1, \ldots, n_b. \tag{2.1.6}$$

In order to write (2.1.6) as a linear system, it is necessary to vectorize the images $\mathbf{B}$ and $\mathbf{X}$. Vectorization is generally done by column-stacking, i.e. given $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_{n_b} \end{bmatrix}$, then

$$\mathbf{b} = \mathbf{B}(:) = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{n_b} \end{bmatrix}$$

The notation $\mathbf{b} = \mathbf{B}(:)$ is used to represent the column stacking operation. The vector $\mathbf{b} \in \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^N$, where $M = m_b \cdot n_b$ and $N = m_x \cdot n_x$.

The inner sum of (2.1.6) has the same structure as the sum in the 1D convolution (2.1.1). Over all i's, the inner sum is equivalent to the 1D convolution between the $(n_a - \ell)^{th}$ column of the kernel and the $(j + \ell)^{th}$ column of the true image. Thus, the

matrix $\mathbf{A}_\ell$ can be defined as

$$\mathbf{A}_\ell = \begin{bmatrix} a_{m_a,\ell} & a_{m_a-1,\ell} & \cdots & \cdots & a_{2,\ell} & a_{1,n_a-\ell} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & & & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{m_a,\ell} & a_{m_a-1,\ell} & \cdots & \cdots & a_{2,\ell} & a_{1,\ell} \end{bmatrix}, \tag{2.1.7}$$

similar to the matrix in (2.1.2). This matrix has size $m_b \times m_x$. The subscript denotes the corresponding column in the kernel. Then we have

$$\mathbf{b}_j = \sum_{\ell=0}^{n_a-1} \mathbf{A}_{n_a-\ell}\mathbf{x}_{j+\ell}.$$

This has a structure similar to the 1D convolution (2.1.1), except the scalar x's and b's have been replaced with vectors, and the a's have been replaced with matrices. The full 2D operator has an overall structure that looks similar to the 1D case, with a block diagonal structure made up of the $\mathbf{A}_\ell$'s instead of the scalar a_ℓ's. The system can be written as

$$\begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{n_b} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{n_a} & \mathbf{A}_{n_a-1} & \cdots & \cdots & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \ddots & \ddots & & & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{A}_{n_a} & \mathbf{A}_{n_a-1} & \cdots & \cdots & \mathbf{A}_2 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{n_x} \end{bmatrix}, \tag{2.1.8}$$

or $\mathbf{b} = \mathbf{A}\mathbf{x}$, where there are $n_b \times n_x$ sub-matrices in $\mathbf{A}$.

As in the 1D case, this system is generally under-determined. The same classical boundary conditions (periodic, zero, and reflecting) can be implemented to produce a system with a square matrix $\widehat{\mathbf{A}} \in \mathbb{R}^{M\times M}$. The size of $\widehat{\mathbf{A}}$ here implies the boundary conditions are applied to the FOV (the central $M = m_b \cdot n_b$ pixels of the image). In 2D, the boundary conditions are applied in both the vertical and horizontal dimensions of the image. An example with a synthetic image is provided in Figure 2.1.3, which shows a synthetic image with an identified FOV, and how the image would look if

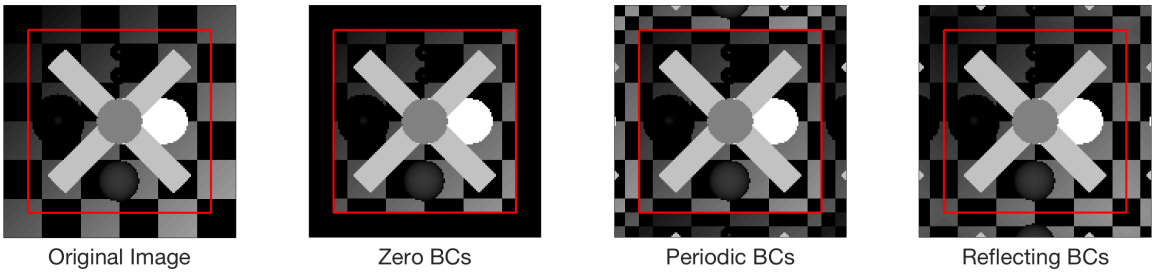


Figure 2.1.3: Boundary conditions for a synthetic image. The true image is included on the left. The white box represents the field of view. The next three images show what the figure would actually look like if it satisfied periodic, zero, or reflecting boundary conditions (left to right). All three boundary conditions lead to images that are different from the true image.

it satisfied classical boundary conditions. As before, outside the FOV, these images differ considerably from the true image. As in the 1D case, each of the classical boundary conditions have associated computationally efficient spectral methods to solve the 2D deconvolution. These methods are similar to the 1D case, except they use the 2D DFT and DCT [1]. Assuming classical boundary conditions in order to use these methods can result in artifacts in the reconstructions, because prior assumptions – the boundary conditions – enforced on the unknown $\mathbf{x}$ are restrictive and lack physical motivation.

Rather than making strict boundary condition assumptions, the problem is instead solved on the extended domain of $\mathbf{x}$. That is, we consider solving the problem as defined in (2.1.8). In [1], it is noted that this system can be represented as

$$\mathbf{b} = \mathbf{A}\mathbf{x} = \mathbf{D}\widehat{\mathbf{A}}\mathbf{x} \tag{2.1.9}$$

where $\widehat{\mathbf{A}} \in \mathbb{R}^{N\times N}$, and $\mathbf{D}$ crops the output to the central M pixels. The operator $\widehat{\mathbf{A}}$ can have any classical boundary conditions, and these boundary conditions act on the pixels outside the FOV. The choice of classical boundary condition is arbitrary since the pixels within the FOV are not affected. Thus spectral methods can be used to perform the action of $\widehat{\mathbf{A}}$ on the extended domain, before cropping the output with $\mathbf{D}$ to the FOV to perform the convolution.

The unresolved issue, as previously noted, is that this system is under-determined. Additionally, although $\widehat{\mathbf{A}}$ is diagonalizable via 2D spectral methods, $\mathbf{A}$ is not, making deconvolution require prohibitively large matrices when the image is large. The solution to this under-determined system is achieved by taking a Bayesian approach, detailed in Section 2.2, and the issue of scalability with dimension is handled via the block Gibbs sampler described in Chapter 3.

2.2 Bayesian formulation

Images are captured as a grid of cells, such as those found in Charge-Coupled Devices (CCD) or Complementary metal–oxide–semiconductor (CMOS) image sensors. Each cell in the grid measures light over a small portion of the field of view and the light intensity can be represented as a scalar quantity. The collection of intensity values from each pixel is stored in the 2D array $\mathbf{B} \in \mathbb{R}^{m_b \times n_b}$ which represents the image data. The intensity value at each pixel location (i, j) is corrupted by blur and noise. Common choices for modeling the noise include Poisson and Gaussian [2, 4], and we focus on the Gaussian case here. The noise is modeled as an additive random component,

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\varepsilon}, \tag{2.2.1}$$

where $\mathbf{b}$, $\mathbf{A}$, and $\mathbf{x}$ represent the column stacked data, the convolution matrix, and the column stacked true image, respectively, as in the 2D convolution equation (2.1.8), and $\boldsymbol{\varepsilon}$ represents the additive Gaussian noise. The random error in each pixel ε_i is assumed to be independently and identically distributed (iid) with mean 0 and precision $\lambda > 0$, or equivalently variance λ^{-1}. This is represented by $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$, where $\mathcal{N}$ represents the Gaussian distribution, and $\mathbf{I}$ is the identity matrix. This

gives rise to the Gaussian likelihood,

$$p_l(\mathbf{b}|\mathbf{x}) \propto \exp\left(-\frac{\lambda}{2}\left|\left|\mathbf{A}\mathbf{x}-\mathbf{b}\right|\right|^2\right), \tag{2.2.2}$$

where $||\cdot||$ denotes the standard Euclidean norm, $||\mathbf{x}|| = \sqrt{\mathbf{x}^\top\mathbf{x}}$, where '$\top$' denotes transpose, and '$\propto$' denotes proportionality.

Maximizing the likelihood $p_l(\mathbf{b}|\mathbf{x})$ in (2.2.2) is not well-posed. The Bayesian technique for dealing with this issue is to impose a prior probability on $\mathbf{x}$. A Gaussian prior

$$p_0(\mathbf{x}) \propto \exp\left(-\frac{\delta}{2}\left|\left|\mathbf{L}^{1/2}\mathbf{x}\right|\right|^2\right), \tag{2.2.3}$$

is assumed, where $\delta > 0$ is the prior precision parameter, and $\mathbf{L}$ is a sparse, symmetric positive semidefinite matrix. In this work, the precision matrix $\mathbf{L}$ is specified as the 2D discrete negative Laplacian. This choice indicates some expectation of smoothness on the image. The action of $\mathbf{L}$ can be represented as a convolution with the kernel

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \tag{2.2.4}$$

and the boundary conditions of the Laplacian are chosen to match those of $\widehat{\mathbf{A}}$ in (2.1.9). Due to the extended boundary conditions of $\mathbf{A} = \mathbf{D}\widehat{\mathbf{A}}$, the choice of boundary conditions has a negligible impact on the reconstructions. Because there are only five non-zero elements in the kernel, the proportion of non-zero elements is at most $5/(m_x \cdot n_x)$ which makes $\mathbf{L}$ sparse for large images. The structure of this matrix is discussed further in Section 3.2.5.

Since both the likelihood and prior are Gaussian, applying Bayes' Theorem results

in a Gaussian posterior distribution:

$$\begin{aligned} p(\mathbf{x}|\mathbf{b}) &\propto p(\mathbf{b}|\mathbf{x})p(\mathbf{x}) \\ &\propto \exp\left(-\frac{\lambda}{2}\left|\left|\mathbf{A}\mathbf{x}-\mathbf{b}\right|\right|^2 - \frac{\delta}{2}\left|\left|\mathbf{L}^{1/2}\mathbf{x}\right|\right|^2\right) \\ &= \mathcal{N}\left(\mathbf{m}, \mathbf{H}^{-1}\right), \end{aligned} \tag{2.2.5}$$

where the posterior precision and mean are given by

$$\mathbf{H} = \lambda \mathbf{A}^\top \mathbf{A} + \delta \mathbf{L}, \tag{2.2.6}$$

$$\mathbf{m} = \lambda \mathbf{H}^{-1} \mathbf{A}^\top \mathbf{b}. \tag{2.2.7}$$

The goal is to produce samples from the posterior distribution to characterize the distribution of the true image given the blurred and noisy data.

Although the posterior distribution is Gaussian, it is still difficult to sample from due to the large dimension. The precision matrix $\mathbf{H}$ has size $N \times N$, where $N = m_x \cdot n_x$. The target application addressed in Chapter 4 considers images with $n_x \approx m_x \approx 4000$, resulting in a problem dimension of $N \approx 16 \cdot 10^6$. This poses computational difficulties even under the Gaussian assumption, as the standard procedure for sampling from a Gaussian distribution involves computing matrix square roots, or other matrix factorizations [14]. The banded structure of the posterior precision matrix makes such factorizations infeasible at the full problem size. However, the local and sparse nature of the operator can be exploited to break the problem into computationally manageable pieces.

2.3 Markov Chains and Gibbs sampling

In the Bayesian setting, uncertainty quantification is performed by statistically analyzing samples drawn from the posterior distribution. It is often non-trivial to sample directly from posterior distributions, generally due to non-standard structure, but in

Algorithm 1: Gibbs Sampler

input : Initial state $\mathbf{x}^{(0)} = \left[{\mathbf{x}_1^{(0)}}^\top, {\mathbf{x}_2^{(0)}}^\top, \cdots, {\mathbf{x}_n^{(0)}}^\top \right]^\top$, maximum iteration N_e, and conditional distributions $p(\mathbf{x}_i|\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \cdots, \mathbf{x}_n, \mathbf{b})$

output: $\left\{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N_e)}\right\}$ where $\mathbf{x}^{(k)} \sim p(\mathbf{x}|\mathbf{b})$

```
for k = 1, ..., N_e do
    for i = 1, ..., n do
        x_i^(k) ~ p( x_i | x_1^(k), ..., x_{i-1}^(k), x_{i+1}^(k-1), ..., x_n^(k-1), b )
    end
    x^(k) = [ x_1^(k)ᵀ, x_2^(k)ᵀ, ..., x_n^(k)ᵀ ]ᵀ
end
```

this case due primarily to the high problem dimension. Markov chain Monte Carlo (MCMC) methods are popular tools for simulating draws from these probability distributions that are otherwise intractable to sample from [7, 17]. These algorithms produce an iterative sequence (or chain) of samples $\{\mathbf{x}^{(k)}\}$, which satisfy the Markov property:

$$p\left(\mathbf{x}^{(k)} \,\middle|\, \mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(k-1)}\right) = p\left(\mathbf{x}^{(k)} \,\middle|\, \mathbf{x}^{(k-1)}\right), \tag{2.3.1}$$

for all $k \geq 2$. That is, each sample $\mathbf{x}^{(k)}$ depends only on the immediately preceding sample, $\mathbf{x}^{(k-1)}$, and is independent of the history of the chain $\left\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(k-2)}\right\}$. The Markov chain is designed so that its stationary distribution is the posterior distribution of interest [7, 17, 40].

The Gibbs sampler named in [16] is a widely used MCMC method in Bayesian inference. The Gibbs sampler requires an N dimensional random variable $\mathbf{x}$ that is made up of (potentially multivariate) components as its input. That is, $\mathbf{x} = \left[\mathbf{x}_1^\top, \mathbf{x}_2^\top, \cdots, \mathbf{x}_n^\top\right]^\top$, where $\mathbf{x}_i \in \mathbb{R}^{d_i}$, and $\sum_{i=1}^n d_i = N$. Starting with an initial point $\mathbf{x}^{(0)}$, the sampler iteratively generates samples $\mathbf{x}^{(k)}$ incrementally via the components $\mathbf{x}_i^{(k)}$. The conditional distributions $p(\mathbf{x}_i|\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \cdots, \mathbf{x}_n, \mathbf{b})$ are used to draw samples for each component. The Gibbs sampler is summarized in Algorithm 1. The

Gibbs sampler presented here is systematic; the components $\mathbf{x}_i$ are updated as a cycle in the labeled order $1 \to \cdots \to n$, and every component is updated. The stationary distribution of the Gibbs sampler is independent of this ordering [17, 36].

Regardless of the sampling order, samples $\{\mathbf{x}^{(k)}\}$ from a Gibbs sampler are not independent, since they are constructed with the Markov property (2.3.1). This affects the convergence and efficiency of the chain, as well as any statistics calculated based on these samples. Setting aside issues of transient behavior [10], the integrated autocorrelation time (IACT), τ_{int}, is a useful measure of the efficiency of an MCMC sampler. The correlation between samples causes statistical error to be a factor of $2\tau_{\text{int}}$ larger than in independent samples [40]. In the results, the estimator fo the IACT proposed by [44] is used. An interpretation of this result that the number of samples of the total that are effectively independent is given by

$$N_{\text{eff}} = \frac{N_e}{2\tau_{\text{int}}}, \tag{2.3.2}$$

where N_e is the total number of samples. IACT and effective sample size are indicators of the efficiency of an MCMC algorithm. It is well known that convergence rates of MCMC methods decrease with dimension. For example, [6], [34], and [33], show that the proposal variance of MCMC samplers decreases with dimension of the problem, which in turn implies that IACT grows with dimension.

Gibbs samplers can also suffer from poor scaling as dimension increases. The convergence rates for a Gibbs sampler are affected by the update scheme used. Both the order the variables are drawn and the way the components are blocked can have a significant effect on convergence rates [35]. In the case of a Gaussian posterior, choosing highly correlated components results in faster convergence [35]. It was also shown in [28] that when the precision and covariance matrices are block-tridiagonal, the Gibbs sampler is effective for sampling high-dimensional Gaussians. Under those assumptions, the convergence rate is independent of dimension.

In image deblurring via 2D convolution, the precision matrix of the Gaussian posterior (2.2.5) is not block-tridiagonal, due to the 2D problem structure. The precision matrix is, however, sparse with a more generally banded structure. Taking advantage of this structure allows one to effectively sample high-dimensional images. For 2D deconvolution, the efficiency of the Gibbs sampler depends critically on the way the components are chosen by blocking the image $\mathbf{x}$. The following section introduces a scalable implementation of a blocked Gibbs sampler for high-dimensional image deblurring problems.

Chapter 3

Scalable Gibbs sampling for high-dimensional posterior distributions in imaging

This chapter covers the implementation of a block Gibbs sampler that is effective in the high dimensional target application. The implementation exploits the local nature of convolution and the resulting sparsity structures of the convolution matrix $\mathbf{A}$ and of the prior precision matrix $\mathbf{L}$. The banded structures of both $\mathbf{A}$ and $\mathbf{L}$ are captured in the posterior precision matrix $\mathbf{H} = \lambda \mathbf{A}^\top \mathbf{A} + \delta \mathbf{L}$, whose structure can be leveraged by carefully partitioning the state, $\mathbf{x}$, of the Markov chain using sub-images to define the components. When the images are relatively small, i.e. when it is possible to store the posterior precision matrix $\mathbf{H}$ in memory, then the sampler can be directly implemented. In this case, solutions can be computed via matrix computations on the sub-images, with matrix sizes defined by the partitions of $\mathbf{x}$. This matrix notation is used to introduce the block Gibbs sampler as a conceptual first step toward describing the overall approach. For larger images, matrix based methods are impractical due to memory constraints. The target application has dimension $16 \cdot 10^6$, making even an

off-line assembly of the posterior precision prohibitively expensive. A function based alternative is introduced to avoid these issues.

3.1 Effective partitioning for Gibbs sampling in imaging

As discussed in Section 2.3, Gibbs samples are generated using the conditional distributions $p(\mathbf{x}_i|\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \cdots, \mathbf{x}_n)$, and sequentially generating samples, $\mathbf{x}_i$, from each conditional [15–17, 37]. In the target application, which has a Gaussian posterior (2.2.5), the conditionals are also Gaussian. The conditional distributions in the case of scalar components are provided in [3, 14] and can be extended to multivariate components. Specifically, the conditional distribution for the i^{th} component $\mathbf{x}_i$ is

$$\begin{aligned}\mathbf{x}_i^{(k)} \Big| \mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_{i-1}^{(k)}, \mathbf{x}_{i+1}^{(k-1)}, \cdots, \mathbf{x}_n^{(k-1)}, \mathbf{b}, \lambda, \delta \\ \sim \mathcal{N}\left(\mathbf{m}_i - \left(\sum_{j>i} \mathbf{H}_{ij}\left(\mathbf{x}_j^{(k-1)} - \mathbf{m}_j\right) + \sum_{j<i} \mathbf{H}_{ij}\left(\mathbf{x}_j^{(k)} - \mathbf{m}_j\right)\right), \mathbf{H}_{ii}^{-1}\right), \quad (3.1.1)\end{aligned}$$

where the $\mathbf{H}_{ij}$ are sub-matrices of the posterior precision matrix $\mathbf{H}$, and the superscript (k) denotes time in the Markov chain. In the context of imaging, the $\mathbf{x}_i$'s correspond to groups of pixels in the image. The structure of any sub-matrix $\mathbf{H}_{ij}$ depends on how the state, or image, $\mathbf{x}$ is partitioned into components as sub-images, $\mathbf{x}_i$.

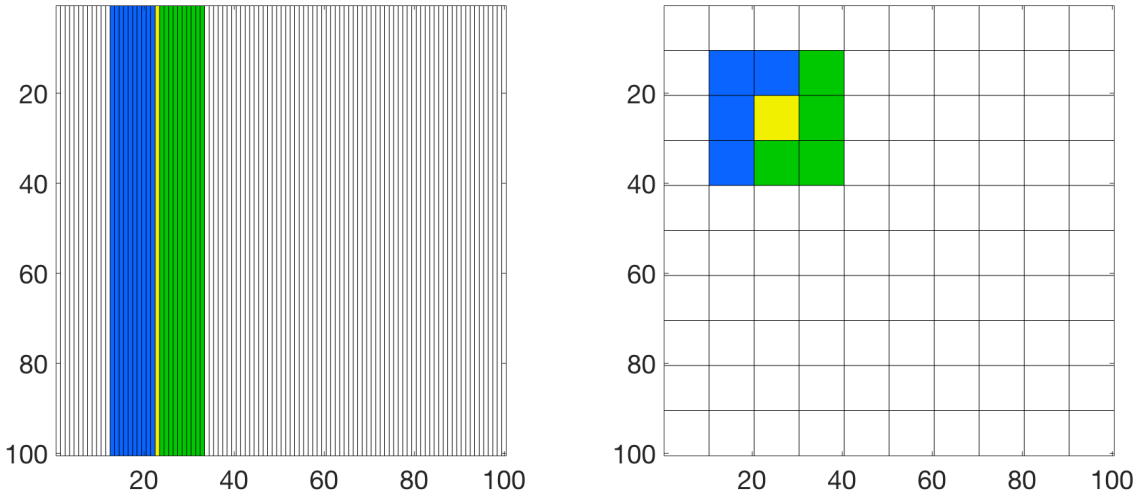
Since the convolution matrix $\mathbf{A}$ defined in (2.1.8) assumes that the vector $\mathbf{x}$ is a column stack of the image $\mathbf{X}$, a naïve choice for the components $\mathbf{x}_i$ corresponds to columns in the image. In this case, the components are simply sequential sub-vectors in the column stacked $\mathbf{x}$. This choice is not practical, since the blurring kernel $\mathbf{a}$ extends in the vertical and horizontal directions. Each pixel in $\mathbf{x}_i$ depends on the

pixels within a 2D-neighborhood based on the size of the kernel, requiring $\mathbf{x}_i$ to be conditioned on all columns to the left and to the right of the column i that are contained in this neighborhood. When $\mathbf{x}_i$ is a single column, this corresponds to the n_a neighboring columns, and thus $\mathbf{x}_i$ depends on n_a of the remaining columns, $\mathbf{x}_j$. For large kernels, n_a is large, and $\mathbf{x}_i$ must be conditioned on a large number of neighboring columns, which reduces the efficiency of the sampler [28]. An example of this for an image $\mathbf{X}$ of size 100×100 is shown in Figure 3.1.1a. With a kernel of size 21×21, the yellow column $\mathbf{x}_i$ depends on the 10 neighboring columns to the left and right. When the Gibbs sampler sequentially samples the columns from left to right, the 10 blue columns to the left represent the location of the $\mathbf{x}_j^{(k-1)}$, and the 10 green columns to the right represent the location of the $\mathbf{x}_j^{(k)}$ in (3.1.1). Reducing the number of components that each $\mathbf{x}_i$ is conditioned on results in a more efficient sampler [28], but, such a reduction requires a different partition of $\mathbf{x}$.

Since $\mathbf{X}$ represents an image stored as a 2D array, and since blur is expected to occur in the vertical and horizontal directions, a better choice for dividing the image into components involves sub-images that have 2D structure. One way to make this partition is to divide the full image $\mathbf{X}$ – which has size $m_x \times n_x$ – into a set of $m_B \cdot n_B$ sub-images $\mathbf{X}_i$,

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_{1+m_B} & \cdots & \mathbf{X}_{1+m_B(n_B-1)} \\ \mathbf{X}_2 & \mathbf{X}_{2+m_B} & \cdots & \mathbf{X}_{2+m_B(n_B-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}_{m_B} & \mathbf{X}_{2m_B} & \cdots & \mathbf{X}_{m_B n_B} \end{bmatrix}. \tag{3.1.2}$$

The subscript on each $\mathbf{X}_i$ indicates one possible sampling order for the Gibbs sampler, i.e. cycling from top to bottom, left to right. In the sections that follow, it is assumed that all sub-images have the same size – denoted $\mathbf{X}_i \in \mathbb{R}^{m_d \times n_d}$ – to avoid more cumbersome notation, but this is not a necessary requirement. When the sub-images are all the same size, the number of sub-images (m_B, n_B) and the size of the sub-



(a) Column based components (b) Sub-image components

Figure 3.1.1: Panels (a) and (b), depict possible blocking schemes for an image $\mathbf{X}$, with 100×100 pixels and kernel size 21×21. In each panel, the yellow component represent the location of $\mathbf{x}_i$, while the blue and green components represent the locations of $\mathbf{x}_j^{(k)}$ and $\mathbf{x}_j^{(k-1)}$ from (3.1.1), respectively. The white components represent the locations of the $\mathbf{x}_j$ that are conditionally independent of $\mathbf{x}_i$. In (a), the components are chosen as columns of the image (100 pixels each). For a kernel of size 21×21, $\mathbf{x}_i$ depends on 20 neighboring columns (10 to the left and right). In (b), the components are 10×10 sub-images (still 100 pixels each), and depend only on the 8 neighboring sub-images (4 in blue and 4 in green).

images (m_d, n_d) can be related to the size of the entire image (m_x, n_x) as follows: $m_B \cdot m_d = m_x$ and $n_B \cdot n_d = n_x$. An example of this blocking scheme is shown in Figure 3.1.1b for an image $\mathbf{X}$ of size 100×100 pixels. There are $m_B \cdot n_B = 10 \cdot 10$ sub-images and each has size $m_d \times n_d = 10 \times 10$ pixels. The yellow component in the center of the highlighted components represents the location of $\mathbf{x}_i$, which depends on the eight neighboring components highlighted in blue and green. When the Gibbs sampling order is defined by the indices in (3.1.2), then the four blue sub-images above and to the left represent the locations of the components that have been sampled in the k^{th} Gibbs iteration, $\mathbf{x}_j^{(k)}$. Similarly, the four green sub-images below and to the right represent the locations of the components that have not been sampled in the k^{th} iteration of the Gibbs sampler, so the previous iteration samples, $\mathbf{x}_j^{(k-1)}$, are used in (3.1.1). Compared to the column-based blocks in Figure 3.1.1a, fewer neighboring

blocks are required at each iteration in the Gibbs sampler.

The blocked Gibbs sampler specific to the conditional distributions (3.1.1) is provided in Algorithm 2. The sums are over the sets $\mathcal{S}_{\text{pre}} \subset \{j : 1 \leq j < i\}$ and $\mathcal{S}_{\text{post}} \subset \{j : i < j \leq n\}$, which contain only the relevant indices j that the i^{th} block is conditioned on. With the sub-image based blocking scheme, any block will always only depend on at most eight neighboring blocks provided that the block size is large enough compared to the kernel size. This is due to the local nature of the blurring process, the relative scales of support of the kernel and the sub-image size. Only pixels that are sufficiently close to the sub-image have an effect on the blur, which depends on the size of the kernel. Specifically, the block and kernel sizes must satisfy $m_a \leq 2m_d + 1$ and $n_a \leq 2n_d + 1$. When these conditions are satisfied, the sets can be $\mathcal{S}_{\text{pre}} = \{i - m_B - 1, i - m_B, i - m_B + 1, i - 1\} \cap \{1, \cdots, n\}$ and $\mathcal{S}_{\text{post}} = \{i + 1, i + m_B - 1, i + m_B, i + m_b + 1\} \cap \{1, \cdots, n\}$. This is possible due to the fact that the rest of the $\mathbf{H}_{ij}$ sub-matrices are zero under this blocking scheme.

The largest possible block size is the entire image, which is the trivial case of solving the original problem: there is only one sub-image, $\mathbf{X}_1 = \mathbf{X}$, and (3.1.1) simplifies to $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{m}, \mathbf{H}^{-1})$. Each sample of $\mathbf{X}$ is independent ($\tau_{\text{int}} = 1/2$), but the required operations are computationally infeasible when the dimension is large. On the other extreme, the smallest possible block size is a single pixel, which requires only scalar operations. Each pixel, however, is dependent on a number of surrounding pixels (dependent on the size of the kernel) and, thus, the IACT increases and the effective sample size decreases. There is thus a trade off between the computability of the sub-problems and the efficiency of the sampler in terms of IACT. Such an optimum is problem dependent and in the target application addressed in Chapter 4 a large block size is required for effective sampling.

Algorithm 2: Block Gibbs Sampler

input : Precision matrix $\mathbf{H}$, mean $\mathbf{m}$, initial state $\mathbf{x}^{(0)} = \left[\mathbf{x}_1^{(0)\top}, \mathbf{x}_2^{(0)\top}, \cdots, \mathbf{x}_n^{(0)\top}\right]^\top$, and maximum iteration N_e

output: $\left\{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N_e)}\right\}$ where $\mathbf{x}^{(k)} \sim \mathcal{N}\left(\mathbf{m}, \mathbf{H}^{-1}\right)$

for $k = 1, \ldots, N_e$ **do**
  for $i = 1, \ldots, n$ **do**
    $\mathcal{S}_{\text{pre}} = \{j \ : \text{ visited close blocks, left or above block } i\}$
    $\mathcal{S}_{\text{post}} = \{j \ : \text{ future close blocks, right or below block } i\}$
    sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_i^{(k)} = \mathbf{m}_i +$
    $\mathbf{H}_{ii}^{-1}\left[\mathbf{H}_{ii}^{\top/2}\mathbf{z} - \left(\sum_{j\in\mathcal{S}_{\text{post}}} \mathbf{H}_{ij}\left(\mathbf{x}_j^{(k-1)} - \mathbf{m}_j\right) + \sum_{j\in\mathcal{S}_{\text{pre}}} \mathbf{H}_{ij}\left(\mathbf{x}_j^{(k)} - \mathbf{m}_j\right)\right)\right]$
  end
  $\mathbf{x}^{(k)} = \left[\mathbf{x}_1^{(k)\top}, \mathbf{x}_2^{(k)\top}, \cdots, \mathbf{x}_n^{(k)\top}\right]^\top$
end

3.2 Matrix free implementation for large images

Generating samples from the conditional distributions

$$\mathbf{x}_i^{(k)} = \mathbf{m}_i + \mathbf{H}_{ii}^{-1}\left[\mathbf{H}_{ii}^{\top/2}\mathbf{z} - \left(\sum_{j\in\mathcal{S}_{\text{post}}} \mathbf{H}_{ij}\left(\mathbf{x}_j^{(k-1)} - \mathbf{m}_j\right) + \sum_{j\in\mathcal{S}_{\text{pre}}} \mathbf{H}_{ij}\left(\mathbf{x}_j^{(k)} - \mathbf{m}_j\right)\right)\right], \tag{3.2.1}$$

in the Gibbs sampler from Algorithm 2 is computationally intensive when the dimension of $\mathbf{x}_i$ is large. This is because the vectors $\mathbf{x}_i^{(k)}$ have dimension equal to the number of pixels in the sub-image they represent, $m_d \times n_d$. Thus, the sub-matrices $\mathbf{H}_{ij}$ have dimension $m_d n_d \times m_d n_d$. Instead of explicitly building the sub-matrices, it is possible to exploit the linearity and structure in the operators $\mathbf{A}$ and $\mathbf{L}$. In doing so, it is possible to construct functions that perform the actions of the sub-matrices $\mathbf{H}_{ij}$ on sub-images of $\mathbf{X}$.

There are four separate computational elements in (3.2.1), that require a matrix-free implementation. The first operation requires performing a backsolve to compute

the action of the inverse of the sub-matrix $\mathbf{H}_{ii}$,

$$f_{\text{diag}}(\mathbf{y}, i) = \mathbf{H}_{ii}^{-1}\mathbf{y}. \tag{3.2.2}$$

The second operation is

$$f_{\text{sqrt}}(\mathbf{z}, i) = \mathbf{H}_{ii}^{\top/2}\mathbf{z}, \tag{3.2.3}$$

which produces the random component in the Gibbs sampler, i.e. a vector distributed $\mathcal{N}(\mathbf{0}, \mathbf{H}_{ii})$. The final two elements are the pre- and post-sums

$$f_{\text{post}}(\mathbf{y}, i) = \sum_{j \in \mathcal{S}_{\text{post}}} \mathbf{H}_{ij}\mathbf{y}_j \quad \text{and} \quad f_{\text{pre}}(\mathbf{y}, i) = \sum_{j \in \mathcal{S}_{\text{pre}}} \mathbf{H}_{ij}\mathbf{y}_j, \tag{3.2.4}$$

which produce the boundary conditions for the i^{th} sub-image, by using the neighboring sub-images. All of these operations depend on the actions of $\mathbf{A}$ and $\mathbf{L}$. Functions related to $\mathbf{A}$ and $\mathbf{L}$ are derived first and used to produce the final functions for (3.2.3)–(3.2.2), all of which return a sub-image of size $m_d \times n_d$.

Equations (3.2.2) – (3.2.4) all contain at least one sub-matrix $\mathbf{H}_{ij}$, and a function computing the action of $\mathbf{H}_{ij}$ is essential to each of these functions. Note that

$$\mathbf{H}_{ij} = \lambda \left(\mathbf{A}_{:,i}\right)^{\top} \mathbf{A}_{:,j} + \delta \mathbf{L}_{ij} \tag{3.2.5}$$

where the ':' represents all block indices, i.e. $\mathbf{A}_{:,j} = \left[\mathbf{A}_{1,j}^{\top} \cdots \mathbf{A}_{m_B n_B, j}^{\top}\right]^{\top}$. Thus an implementation of the action of $\mathbf{H}_{ij}$, depends on implementations of $\left(\mathbf{A}_{:,i}\right)^{\top}$, $\mathbf{A}_{:,j}$, and $\mathbf{L}_{ij}$. Equation (3.2.3) also requires the action of the square root of $\mathbf{L}_{ii}$. These all rely on matrix free implementations of convolution with a kernel $\mathbf{a}$ for on arbitrarily sized sub-image and a similar implementation for the prior precision. Sections 3.2.1–3.2.3 provide a high level overview assuming these matrix free implementations are available. Sections 3.2.4–3.2.6 provide the implementations of the required functions.

3.2.1 Computing the action of $\mathbf{H}_{ii}^{-1}$

The first computational element of the conditional distribution (3.2.1) is computing the action of the inverse of $\mathbf{H}_{ii}$ as stated in (3.2.2). The action of the forward operator denoted $f_{\mathbf{H}_{ii}}(\cdot)$ is a combination of the functions $f_{(A_{:,i})^\top}(\cdot)$, $f_{A_{:,j}}(\cdot)$, and $f_{L_{ij}}(\cdot)$ which are defined in Sections 3.2.4 and 3.2.5 (see equations (3.2.14), (3.2.17), and (3.2.22)). It is given by

$$f_{\mathbf{H}_{ii}}(\mathbf{x}_i, i) = \lambda f_{(\mathbf{A}_{:,i})^\top}\left(f_{\mathbf{A}_{:,i}}(\mathbf{X}_i)\right) + \delta f_{\mathbf{L}_{ii}}(\mathbf{X}_i), \tag{3.2.6}$$

Given the function (3.2.6) that performs the action of the forward operator, there are a host of different methods for finding the inverse. One such method is the Conjugate Gradient (CG) method, which is an iterative method for solving linear systems with symmetric, positive definite (SPD) matrices. Since $\mathbf{L}$ is SPD, $\mathbf{A}^\top\mathbf{A}$ is symmetric positive semidefinite, and $\lambda, \delta > 0$, then $\mathbf{H}$ is SPD. The CG algorithm produces an exact solution (assuming no numerical error) in n iterations, where $\mathbf{x} \in \mathbb{R}^n$ [22, 43]. It is common to stop well before n iterations are reached, which effectively acts as further regularization for the problem. The specific stopping criteria for CG used in the application are covered in Chapter 4.

3.2.2 Generating Gaussian random vectors with mean 0 and precision $\mathbf{H}_{ii}$

In Algorithm 2, the purpose of the the square root matrix $\mathbf{H}_{ii}^{\top/2}$ acting on a random vector $\mathbf{z}$ in the conditional distribution (3.2.1) is to generate a sample from $\mathcal{N}(\mathbf{0}, \mathbf{H}_{ii})$. For small matrices $\mathbf{H}_{ii}$, the usual method of producing such a sample is to compute a matrix square root, e.g. a Choleskey factor [43]. When $\mathbf{H}_{ii}$ is large, this is both computationally expensive and memory intensive. To avoid this, we instead consider the structure of $\mathbf{H}_{ii}$, and generate samples via sums of independent Gaussians. Recall

that

$$\mathbf{H}_{ii} = \lambda \left(\mathbf{A}_{:,i}\right)^{\top} \mathbf{A}_{:,i} + \delta \mathbf{L}_{ii}. \tag{3.2.7}$$

Consider the first term, $\lambda \left(\mathbf{A}_{:,i}\right)^{\top} \mathbf{A}_{:,i}$. It is easy to generate normally distributed random vectors $\mathbf{z}$ with mean zero and covariance $\mathbf{I}$ for any dimension. The dimension of the required vector $\mathbf{z}_1 = \mathbf{Z}_1(:)$ depends on the location of the sub-image $\mathbf{X}_i$, where $\mathbf{Z}_1$ has size $(m_4 - m_3 + 1, n_4 - n_3 + 1)$ (defined in (3.2.16)). Then the function $f_{(\mathbf{A}_{:,i})^{\top}}(\cdot)$, defined in (3.2.17) can be used to produce output with the distribution

$$f_{(\mathbf{A}_{:,i})^{\top}}(\mathbf{Z}_1) \sim \mathcal{N}\left(\mathbf{0}, \left(\mathbf{A}_{:,i}\right)^{\top} \mathbf{A}_{:,i}\right).$$

Similarly, given a function $f_{\mathbf{L}_{ii}^{\top/2}}(\cdot)$ and appropriately size $\mathbf{Z}_2$ where $\mathbf{z}_2 = \mathbf{Z}_2(:)$ has mean zero and covariance $\mathbf{I}$, then

$$f_{\mathbf{L}_{ii}^{\top/2}}(\mathbf{Z}_2) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{L}_{ii}\right).$$

An efficient implementation for the negative 2D Laplacian makes use of the fact that $\mathbf{L}$ can be written as $\mathbf{L} = \mathbf{D}_v^{\top}\mathbf{D}_v + \mathbf{D}_h^{\top}\mathbf{D}_h$, where $\mathbf{D}_v$ and $\mathbf{D}_h$ depend on the boundary conditions of $\mathbf{L}$ and are covered extensively in Section 3.2.5. Due to the dependence on $\mathbf{D}_v$ and $\mathbf{D}_h$, the functions $f_{\mathbf{D}_v}(\cdot)$ and $f_{\mathbf{D}_h}(\cdot)$ are required (see Equations (3.2.26) and (3.2.26)). Then computing two independent random vectors $\mathbf{z}_2, \mathbf{z}_3 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ where $\mathbf{Z}_2, \mathbf{Z}_3$ have size $m_d \times n_d$, we have

$$f_{\mathbf{D}_v^{\top}}(\mathbf{Z}_2) + f_{\mathbf{D}_h^{\top}}(\mathbf{Z}_3) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{L}_{ii}\right).$$

Scaling these outputs by $\lambda^{1/2}$ and $\delta^{1/2}$, we get

$$\lambda^{1/2} f_{(\mathbf{A}_{:,i})^{\top}}(\mathbf{Z}_1) + \delta^{1/2}\left(f_{\mathbf{D}_v^{\top}}(\mathbf{Z}_2) + f_{\mathbf{D}_h^{\top}}(\mathbf{Z}_3)\right) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{H}_{ii}\right), \tag{3.2.8}$$

i.e. this function produces outputs with the same distribution produced by the square root matrix $\mathbf{H}_{ii}^{\top/2}$ in (3.2.3).

3.2.3 Efficient computation of the pre- and post-sums

In this section, an efficient implementation of the pre- and post-sums, given in (3.2.4), are presented by taking advantage of the linearity of $\mathbf{H}$, and carefully constructing sub-images $\mathbf{X}_{\text{pre}}$ and $\mathbf{X}_{\text{post}}$ (see (3.2.31), (3.2.32)). The pre-sum can be computed using the functions $f_{(\mathbf{A}_{:,i})^\top}(\cdot)$, $f_{\mathbf{A}}(\cdot)$, and a slight alteration of the $f_{\mathbf{L}}(\cdot)$ function $f_{\mathbf{L},\text{pre}}(\cdot)$ (see (3.2.17), (3.2.33) and (3.2.36) respectively) as

$$f_{\text{pre}}(\mathbf{x}, i) = \lambda f_{(\mathbf{A}_{:,i})^\top}\left(f_{\mathbf{A}}(\mathbf{X}_{\text{pre}})\right) + \delta f_{\mathbf{L},\text{pre}}(\mathbf{X}). \tag{3.2.9}$$

Similarly, the post-sum can be computed using the functions $f_{(\mathbf{A}_{:,i})^\top}(\cdot)$, $f_{\mathbf{A}}(\cdot)$, and a slight alteration of the $f_{\mathbf{L}}(\cdot)$ function $f_{\mathbf{L},\text{post}}(\cdot)$ (see (3.2.17), (3.2.34) and (3.2.37) respectively) as

$$f_{\text{post}}(\mathbf{x}, i) = \lambda f_{(\mathbf{A}_{:,i})^\top}\left(f_{\mathbf{A}}(\mathbf{X}_{\text{post}})\right) + \delta f_{\mathbf{L},\text{post}}(\mathbf{X}) \tag{3.2.10}$$

Combining CG with (3.2.6) and using (3.2.8), (3.2.9), and (3.2.10) provides all the necessary components for an implementation of (3.2.1) for problems where the dimension of the sub-image is so large that a matrix based implementation is infeasible.

3.2.4 The action of the convolution matrix A

In order to compute the four actions of the $\mathbf{H}_{ij}$ sub-matrices defined in equations (3.2.3)–(3.2.2) that make up the conditional distribution (3.1.1), it is necessary to have a matrix free convolution function given a kernel $\mathbf{a}$, denoted as $f_{\mathbf{A}}$. This is the standard convolution as described in [1], and provided in Algorithm 3.

Algorithm 3: Convolution $f_{\mathbf{A}}$

input : Image $\mathbf{X} \in \mathbb{R}^{m_x \times n_x}$, and kernel $\mathbf{a} \in \mathbb{R}^{m_a \times n_a}$

output: Image $\mathbf{Y} \in \mathbb{R}^{m_b \times n_b}$

```
for i = 1, ..., m_b do
    for j = 1, ..., n_b do
        y_ij = ã · X(i : i + m_a − 1, j : j + n_a − 1)
    end
end
```

In Algorithm 3 the "$\cdot$" represents the dot product defined in (2.1.6), and $\tilde{\mathbf{a}}$ represents the kernel $\mathbf{a}$ flipped both vertically and horizontally (note: this is equivalent to a rotation of the position of the elements in $\mathbf{a}$ by 180°). For any pixel position (i, j) and sub-image size (m, n) satisfying $i, j \geq 1$, $m_a \leq m \leq m_x - i$, $n_a \leq n \leq n_x - j$, we have

$$f_{\mathbf{A}}(\mathbf{X}(i : i + m - 1, j : j + n - 1)) = \mathbf{Y}(i : i + m - m_a - 1, j : j + n - n_a - 1). \tag{3.2.11}$$

That is, the function $f_{\mathbf{A}}$ applied to any sub-image of $\mathbf{X}$ produces the corresponding *smaller* sub-image in $\mathbf{Y}$. This will prove useful when considering sub-images $\mathbf{X}_j$ of the entire image $\mathbf{X}$.

It is possible to use variations of the function $f_{\mathbf{A}}(\cdot)$ to implement actions of $\mathbf{A}_{:,j}$ as required for Equations (3.2.2) – (3.2.4). Recall that the index ":" represents all block indices, i.e. $\mathbf{A}_{:,j} = \left[\mathbf{A}_{1,j}^\top \cdots \mathbf{A}_{m_B n_B, j}^\top\right]^\top$. The matrix $\mathbf{A}_{:,j}$ has size $M \times m_d \cdot n_d$, and produces an output which is the size of the entire blurred image (M pixels) corresponding only to the j^{th} sub-image of the input. Due to the local nature of the blur, only a small portion of the M pixels in the output are non-zero. This is illustrated in Figure 3.2.1, showing the output for an $\mathbf{X}_j$ in the interior of the image $\mathbf{X}$, and another on the border of $\mathbf{X}$. In both cases, the output is only nonzero in areas

close to $\mathbf{X}_j$, and the extent depends on the size of the kernel. This can be written using the entire operator $\mathbf{A}$ by setting every other block $\mathbf{x}_i$ with $i \neq j$ to zero,

$$\mathbf{A}_{:,j}\mathbf{x}_j = \mathbf{A} \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{x}_j \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}. \tag{3.2.12}$$

As an image, i.e. removing the column stacks, this is equivalent to

$$\mathbf{A}_{:,j}\mathbf{x}_j = f_{\mathbf{A}} \left(\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_j & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}_{m_x \times n_x} \right), \tag{3.2.13}$$

where the $\mathbf{0}$ blocks are appropriately sized so that the $\mathbf{X}_j$ block is in the j^{th} position. The action of $f_{\mathbf{A}}(\cdot)$ on an image with a single non-zero sub-image is depicted in Figure 3.2.2, for the same interior and boundary sub-images as in Figure 3.2.1. Since the non-zero portion of the output is in a relatively small section of the output, it is possible to compute just the non-zero portion and keep track of its location in the output. Producing just the non-zero portion of the output requires zero padding $\mathbf{X}_j$ based on the size of the kernel, as opposed to zero padding to the size of the whole image $\mathbf{X}$. Blocks on the boundary of the image require less padding, as shown in figure 3.2.3. This can be written as

$$f_{\mathbf{A}_{:,j}}(\mathbf{X}_j) = f_{\mathbf{A}} \left(\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_j & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \right), \tag{3.2.14}$$

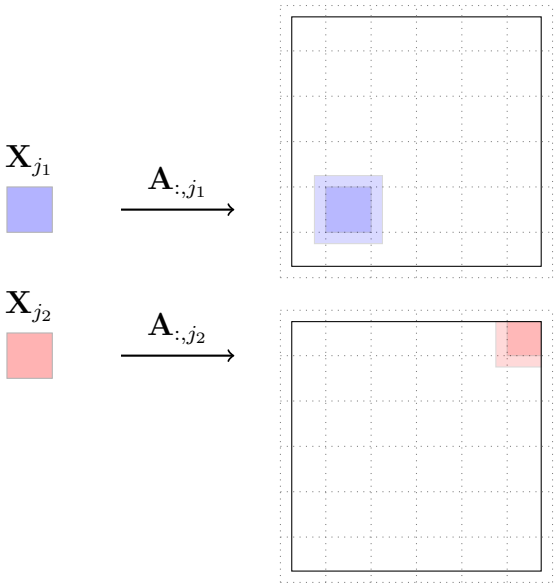


Figure 3.2.1: $\mathbf{A}_{:,j}$ acts on a single sub-image $\mathbf{X}_j$ to produce an output image of size $m_b \times n_b$. The dotted lines in the output indicate the sub-images, which are the same size as in $\mathbf{X}$ for internal sub-images such as $\mathbf{X}_{j_1}$, but differ due to the extended boundary for sub-images on a border of the image $\mathbf{X}$ such as $\mathbf{X}_{j_2}$.

where the $\mathbf{0}$ blocks in (3.2.14) are much smaller than in (3.2.13), and depend on the size of the kernel. Specifically, $\mathbf{X}_j$ is padded with $(m_a - 1)$ zeros on the top and bottom, and $(n_a - 1)$ zeros on the left and right, so long as it is not on the edge of the image, in which case there is no zero padding. For arbitrary sub-image size (m_d, n_d) and kernel size (m_a, n_a), the padding can be calculated as

$$\begin{aligned} m_1 &= \min\left\{m_a - 1, (k_j - 1)m_d\right\}, \\ n_1 &= \min\left\{n_a - 1, (\ell_j - 1)n_d\right\}, \\ m_2 &= \min\left\{m_a - 1, m_x - k_j m_d\right\}, \\ n_2 &= \min\left\{n_a - 1, n_x - \ell_j n_d\right\}, \end{aligned} \tag{3.2.15}$$

where (k_j, ℓ_j) are the unique subscripts corresponding to $j = k_j + (\ell_j - 1)m_B$, and satisfy $1 \leq k_j \leq m_B$, $1 \leq \ell_j \leq n_B$. These values correspond to the padding to the left (m_1), right (m_2), top (n_1) and bottom (n_2) of the sub-image $\mathbf{X}_j$.

Next, the action of $(\mathbf{A}_{:,i})^\top$ is considered, as it is required to compute the action

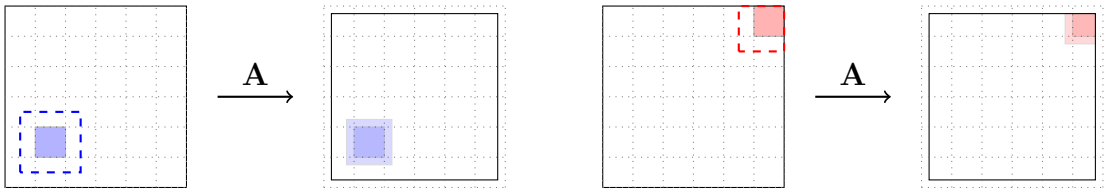


Figure 3.2.2: On the left, the original image is shown, with two different sub-images $\mathbf{X}_j$ shaded in red and blue. The entire $\mathbf{A}$ acting on an image the size of $\mathbf{X}$, where only the $\mathbf{X}_j$ sub-image is non-zero results in the same output as in Figure 3.2.1. The fact that much of the output is zero indicates that it is possible to apply the convolution to just a sub-image to produce the non-zero portion of the output (see Figure 3.2.3).

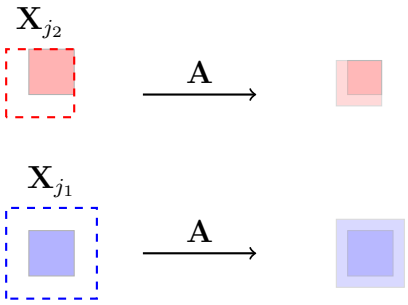


Figure 3.2.3: In order to produce only the non-zero portion of the output from Figure 3.2.2, the sub-images require only a small amount of zero-padding indicated by the dotted lines. The padding differs for sub-images on the interior of the image such as $\mathbf{X}_{j_1}$ as compared to those on the boundary of the image such as $\mathbf{X}_{j_2}$.

of $\mathbf{H}_{ij}$ in (3.2.5). This block row matrix has size $m_d \cdot n_d \times M$, and acts on an object the size of the entire $M = m_b \cdot n_b$ pixel image $\mathbf{Y}$. It produces the i^{th} sub-image of the output (size $m_d \times n_d$) convolved with the blurring kernel $\mathbf{a}$ flipped in the vertical and horizontal directions, denoted $\tilde{\mathbf{a}}$. Although $(\mathbf{A}_{:,i})^\top$ acts on M elements, only a small proportion of the elements affect the output, due to the sparsity in $\mathbf{A}$. These elements consist of all those in the i^{th} sub-image, as well as the elements bordering the sub-image, and depend on the size of the kernel. Figure 3.2.4 shows how these bounds change depending on the location of the sub-image. For interior sub-images, the bounds are within the neighboring sub-images. For sub-images on the border of the image, zero-padding is required.

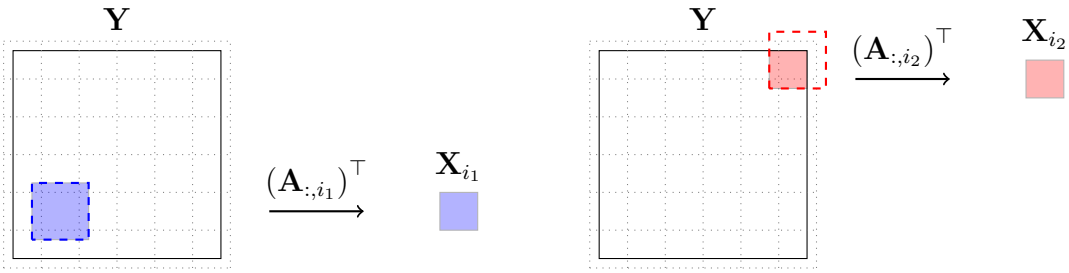


Figure 3.2.4: $(\mathbf{A}_{:,i})^\top$ acts on the entire image $\mathbf{Y}$ to produce the output of the i^{th} sub-image when convolved with $\tilde{\mathbf{a}}$. For any sub-image the relevant part of the input has size $(m_d + m_a - 1, n_d + n_a - 1)$. For sub-images on the interior of $\mathbf{Y}$ such as indexed by i_1, the boundary is made up of the neighbors. For sub-images on the edge of $\mathbf{y}$ such as indexed by i_2, part of the boundary is zero-padded, indicated by the dashed lines.

Figure 3.2.5: The bounds defined in (3.2.16) for the two different example indices in figure 3.2.4. The dotted lines represent the zero-padding done by $f_{\text{pad}}(\cdot)$ in (3.2.17).

The relevant bounds in the image can be defined as

$$\begin{aligned} m_3 &= 1 + \max\{0, (k_i - 1)m_d - (m_a - 1), \} \\ n_3 &= 1 + \max\{0, (\ell_i - 1)n_d - (n_a - 1), \} \\ m_4 &= \min\{k_i m_d, m_b\}, \\ n_4 &= \min\{\ell_i n_d, n_b\}, \end{aligned} \tag{3.2.16}$$

where (k_i, ℓ_i) are the unique subscripts corresponding to $i = k_i + (\ell_i - 1)m_B$, and satisfy $1 \le k_i \le m_B$, $1 \le \ell_i \le n_B$. Thus the relevant portion of the image is given by $\mathbf{Y}(m_3 : m_4, n_3 : n_4)$. Similar to the $\mathbf{0}$ padding in (3.2.14), the size of the $\mathbf{Y}$ sub-image changes with i. The central blocks of the input $\mathbf{Y}$ are all the same size: $(m_d + m_a - 1, n_d + n_a - 1)$, while those on the border of $\mathbf{Y}$ are smaller.

In order to understand how $(\mathbf{A}_{:,i})^\top$ acts on this block, recall that $\mathbf{A}$ can be written as the product $\mathbf{D}\widehat{\mathbf{A}}$ as in (2.1.9). Here $\widehat{\mathbf{A}} \in \mathbb{R}^{N \times N}$ is the convolution matrix with

known boundary conditions, and $\mathbf{D} \in \mathbb{R}^{M \times N}$ is the cropping operator, which crops out the M elements corresponding to the output image. The transpose, $\mathbf{A}^\top = \widehat{\mathbf{A}}^\top \mathbf{D}^\top$, first extends the image from size M to size N, and then applies the convolution $\widehat{\mathbf{A}}^\top$. As previously noted, this is just the convolution with $\tilde{\mathbf{a}}$ rather than $\mathbf{a}$. Note that

$$\mathbf{A}_{:,i} = (\mathbf{D}\widehat{\mathbf{A}})_{:,i} = \sum_j \mathbf{D}_{:,j}\widehat{\mathbf{A}}_{j,i} = \mathbf{D}\widehat{\mathbf{A}}_{:,i},$$

and $(\mathbf{A}_{:,i})^\top = (\widehat{\mathbf{A}}_{:,i})^\top \mathbf{D}^\top$. That is, the image is first extended via $\mathbf{D}^\top$, then the relevant portion is convolved. As with (3.2.14), the sparsity of $\mathbf{A}$ allows us to neglect much of the extended input, as it does not affect the output. In the case of the i^{th} block, only blocks close enough to the edge are extended. After the extension, the total size of any block is the same as the central block: $(m_d + m_a - 1, n_d + n_a - 1)$. Applying the convolution function with the appropriate kernel then returns the output with size (m_d, n_d). Thus,

$$f_{(\mathbf{A}_{:,i})^\top}(\mathbf{Y}) = f_{\tilde{\mathbf{A}}}\left(f_{\text{pad}}\left(\mathbf{Y}(m_3 : m_4, n_3 : n_4)\right)\right), \tag{3.2.17}$$

where $f_{\text{pad}}(\cdot)$ zero-pads the input to size $(m_d + m_a - 1, n_d + n_a - 1)$, based on the block index i, and $f_{\tilde{\mathbf{A}}}$ represents convolution with $\tilde{\mathbf{a}}$.

3.2.5 The action of the prior precision matrix L

There are many choices for the prior precision matrix $\mathbf{L}$. The basic requirement for this matrix is that it needs to be sparse and symmetric positive definite (SPD). In order to use the method provided in this paper, a functional version of $\mathbf{L}$ is required that is capable of returning the action of the sub-matrix $\mathbf{L}_{ij}$. In this section, a function is defined for $\mathbf{L}$ as the discrete negative 2D Laplacian.

In Section 2.2, it is noted that the action of the 2D discrete negative Laplacian is

equivalent to convolution with the kernel

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix},$$

with some specified boundary conditions. Thus the functional version for the matrix $\mathbf{L}$ is similar to $\mathbf{A}$, but in this case the boundary conditions need to be taken into consideration. First, the image must be padded according to the boundary conditions, which only require a single pixel of padding in each direction For example, with periodic boundary conditions, the padding is defined as

$$\mathbf{X}_{BC} = \begin{bmatrix} X_{m_x,n_x} & \mathbf{X}_{m_x,:} & X_{m_x,1} \\ \mathbf{X}_{:,n_x} & \mathbf{X} & \mathbf{X}_{:,1} \\ X_{1,m_x} & \mathbf{X}_{1,:} & X_{1,1} \end{bmatrix}, \tag{3.2.18}$$

where $X_{i,j}$ is the $(i,j)^{\text{th}}$ pixel of $\mathbf{X}$, $\mathbf{X}_{i,:}$ represents the i^{th} row of pixels in $\mathbf{X}$ and $\mathbf{X}_{:,j}$ represents the j^{th} column of pixels in $\mathbf{X}$. For zero boundary conditions, the padding is just a single zero in each direction,

$$\mathbf{X}_{BC} = \begin{bmatrix} 0 & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{X} & \mathbf{0} \\ 0 & \mathbf{0} & 0 \end{bmatrix}. \tag{3.2.19}$$

Note that the corner zeros are not bold since they are scalars. In both cases, $\mathbf{X}_{BC} \in \mathbb{R}^{(m_x+2)\times(n_x+2)}$, and $\mathbf{X}$ is the center $m_x \times n_x$ pixels. Then

$$\begin{aligned} f_{\mathbf{L}}(\mathbf{X}) = 4\mathbf{X} - \Big(& \mathbf{X}_{BC}(1:m_x, 2:n_x+1) + \mathbf{X}_{BC}(3:m_x+2, 2:n_x+1) \\ & + \mathbf{X}_{BC}(2:m_x+1, 1:n_x) + \mathbf{X}_{BC}(1:m_x, 3:n_x+2) \Big). \end{aligned} \tag{3.2.20}$$

Similar to $f_{\mathbf{A}}$, this function can also be applied to any sub-image of $\mathbf{X}$, e.g. for any pixel position (i,j) that satisfies $1 \leq i < m_x$ and $1 \leq j < n_x$, and sub-image size

(m, n) that satisfies $1 \leq m \leq m_x - i$ and $1 \leq n \leq n_x - i$,

$$\begin{aligned} f_{\mathbf{L}}(\mathbf{X}(i : i+m, j : j+n)) = 4\mathbf{X}(i : i+m, j : j+n) \\ - \Big(\mathbf{X}_{BC}(i : i+m, j+1 : j+n+1) \\ + \mathbf{X}_{BC}(i+2 : i+m+2, j+1 : j+n+1) \\ + \mathbf{X}_{BC}(i+1 : i+m+1, j : j+m) \\ + \mathbf{X}_{BC}(i+1 : i+m+1, j+2 : j+n+2) \Big). \end{aligned}$$

The function for the sub-block $\mathbf{L}_{ii}$ corresponding to the sub-image $\mathbf{X}$ is very similar. Note that $\mathbf{L}_{ii}$ produces the portion of the output block i due only to the input sub-image i. This is equivalent to the output of $f_{\mathbf{L}}$ with zero boundary conditions on the sub-image. That is,

$$f_{\mathbf{L}_{ii}}(\mathbf{X}_i) = f_{\mathbf{L}}(\mathbf{X}_i) \tag{3.2.21}$$

with zero boundary conditions applied. For the sub-block $\mathbf{L}_{ij}$, which corresponds to the portion of the output sub-image i due to the input sub-image $\mathbf{X}_j$, the output is only nonzero when $\mathbf{X}_i$ and $\mathbf{X}_j$ share a border. Recall the block subscripts are as defined in (3.1.2), so the sub-images sharing a border with $\mathbf{X}_i$ are indexed by

$j \in \{i-1, i+1, i-m_B, i+m_B\}$. The function for any (i,j) pair for $\mathbf{L}_{ij}$ is given by

$$f_{\mathbf{L}_{ij}}(\mathbf{X}_j) = \begin{cases} f_{\mathbf{L}}(\mathbf{X}_j) \quad \text{(with 0 BCs)} & \text{if } j = i \\ -\begin{bmatrix} \mathbf{0}_{m_d \times (n_d-1)} & \mathbf{X}_j(1:m_d, 1) \end{bmatrix} & \text{if } j = i - m_B \\ -\begin{bmatrix} \mathbf{0}_{(m_d-1)\times n_d} \\ \mathbf{X}_j(1, 1:n_d) \end{bmatrix} & \text{if } j = i-1 \\ -\begin{bmatrix} \mathbf{X}_j(m_d, 1:n_d) \\ \mathbf{0}_{(m_d-1)\times n_d} \end{bmatrix} & \text{if } j = i+1 \\ -\begin{bmatrix} \mathbf{X}_j(1:m_d, n_d) & \mathbf{0}_{m_d \times (n_d-1)} \end{bmatrix} & \text{if } j = i + m_B \\ \mathbf{0}_{m_d \times n_d} & \text{otherwise.} \end{cases} \tag{3.2.22}$$

Combining this function with the functions for $\mathbf{A}$ given in (3.2.14) and (3.2.17), it is possible to produce the action of any sub-matrix $\mathbf{H}_{ij}$. These are used to derive the functions for the sums in (3.2.4).

The next function to consider is the random component, which requires the action of the square root matrix $\mathbf{H}_{ii}^{\top/2}$ defined in (3.2.3). The action of $\mathbf{H}_{ii}^{\top/2}$ requires a means of computing the square root of $\mathbf{L}$. In [1], it is shown that for periodic boundary conditions, $\mathbf{L}$ can also be written in the form

$$\mathbf{L} = \mathbf{D}_h^\top \mathbf{D}_h + \mathbf{D}_v^\top \mathbf{D}_v, \tag{3.2.23}$$

and $\mathbf{D}_v = \mathbf{D}_p \otimes \mathbf{I}_{m_x}$ and $\mathbf{D}_h = \mathbf{I}_{n_x} \otimes \mathbf{D}_p$, where $\otimes$ represents the Kronecker product

and $\mathbf{D}_p$ is the forward difference operator with periodic boundary conditions:

$$\mathbf{D}_p = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & -1 & 1 \\ 1 & 0 & \cdots & 0 & -1 \end{bmatrix}. \tag{3.2.24}$$

The zero boundary condition case can be written in a similar form, except with

$$\mathbf{D}_z = \begin{bmatrix} \sqrt{2} & -\sqrt{1/2} & 0 & \cdots & 0 \\ 0 & \sqrt{3/2} & -\sqrt{2/3} & \ddots & \vdots \\ 0 & \ddots & \sqrt{4/3} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -\sqrt{\frac{n-1}{n}} \\ 0 & \cdots & \cdots & 0 & \sqrt{\frac{n+1}{n}} \end{bmatrix}, \tag{3.2.25}$$

in place of $\mathbf{D}_p$, where n is either m_x or n_x in order to create the appropriately sized matrix. Thus we can define two functions for the matrices $\mathbf{D}_v$ and $\mathbf{D}_h$ which differ depending on the boundary conditions. In both cases, they can be formulated as

$$f_{\mathbf{D}_v^\top}(\mathbf{X}) = \mathbf{W}_1 \odot \mathbf{X}_{BC}(3 : m_x + 2, 2 : n_x + 1) - \mathbf{W}_2 \odot \mathbf{X}, \tag{3.2.26}$$

$$f_{\mathbf{D}_h^\top}(\mathbf{X}) = \mathbf{W}_1^\top \odot \mathbf{X}_{BC}(2 : m_x + 1, 3 : n_x + 2) - \mathbf{W}_2^\top \odot \mathbf{X}. \tag{3.2.27}$$

where $\odot$ represents element-wise multiplication. For periodic boundary conditions, $\mathbf{W}_1 = \mathbf{W}_2 = \mathbf{1}$, i.e. the matrix of all 1s. For zero boundary conditions, $\mathbf{W}_1$ and $\mathbf{W}_2$

are defined as

$$\mathbf{W}_1 = \begin{bmatrix} \sqrt{2} & \sqrt{3/2} & \sqrt{4/3} & \cdots & \sqrt{\frac{n+1}{n}} \\ \sqrt{2} & \sqrt{3/2} & \sqrt{4/3} & \cdots & \sqrt{\frac{n+1}{n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sqrt{2} & \sqrt{3/2} & \sqrt{4/3} & \cdots & \sqrt{\frac{n+1}{n}} \end{bmatrix},$$

$$\mathbf{W}_2 = \begin{bmatrix} 0 & \sqrt{1/2} & \sqrt{2/3} & \sqrt{3/4} & \cdots & \sqrt{\frac{n-1}{n}} \\ 0 & \sqrt{1/2} & \sqrt{2/3} & \sqrt{3/4} & \cdots & \sqrt{\frac{n-1}{n}} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \sqrt{1/2} & \sqrt{2/3} & \sqrt{3/4} & \cdots & \sqrt{\frac{n-1}{n}} \end{bmatrix}.$$

$\mathbf{W}_1$ contains the elements on the main diagonal of $\mathbf{D}_z$, while $\mathbf{W}_2$ contains the elements of the first super-diagonal of $\mathbf{D}_z$ in (3.2.25). Here $n = n_x$ for $f_{\mathbf{D}_v^\top}$, and $n = m_x$ for $f_{\mathbf{D}_h^\top}$. The primary interest in these functions is producing random sub-images that have distribution $\mathcal{N}(\mathbf{0}, \mathbf{L}_{ii})$. Since

$$\begin{aligned} \mathbf{L}_{ii} &= \left(\mathbf{D}_h^\top \mathbf{D}_h + \mathbf{D}_v^\top \mathbf{D}_v\right)_{ii} \\ &= \left(\mathbf{D}_h(:,i)\right)^\top \mathbf{D}_h(:,i) + \left(\mathbf{D}_v(:,i)\right)^\top \mathbf{D}_v(:,i), \end{aligned}$$

we require functions for $\left(\mathbf{D}_h(:,i)\right)^\top$ and $\left(\mathbf{D}_v(:,i)\right)^\top$. These block row matrices have size $(m_d \cdot n_d) \times N$, and produce the output pertaining to the i^{th} sub-image, given the entire image. The only relevant portion of the entire image are the pixels directly bordering them, with the relevant boundary conditions applied to the sub-images on the boundary of the image. Then we can just apply the $f_{\mathbf{D}_v^\top}$ and $f_{\mathbf{D}_h^\top}$ from Equations (3.2.26) and (3.2.27) to produce outputs equivalent in distribution when applied to random vectors. Such vectors are an essential component of random sub-images that have distribution $\mathcal{N}(\mathbf{0}, \mathbf{H}_{ii})$.

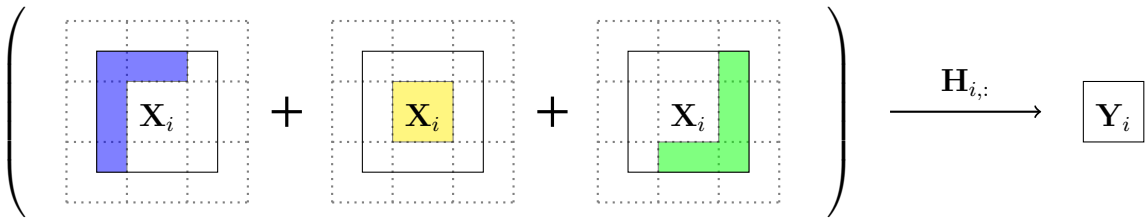


Figure 3.2.6: Due to linearity, $\mathbf{H}_{i,:}$ can act on the three pieces of the image separately to produce the output $\mathbf{Y}_i$, which are the "pre" (blue, left), i (yellow, center), and "post" (green right) pieces in the figure. For sub-images $\mathbf{X}_i$ on the border of the image, part or all of the "pre" or "post" pieces will not exist. The size of the pre- and post-pieces depend on the kernel $\mathbf{a}$, but are at most the size of the dotted outline.

3.2.6 Actions required to compute the pre- and post-sums

This section provides the details on calculating the pre- and post-sums, given in (3.2.4). These sums together contain all of the non-zero sub-matrices $\mathbf{H}_{ij}$ such that $j \neq i$. Therefore it will be useful to consider the collection of all these sub-matrices, $\mathbf{H}_{i,:}$ in order to produce functions for these two sums. In order to produce functions for these sums, it is useful to note that the i^{th} sub-image of the output from the blurring process $\mathbf{y} = \mathbf{H}\mathbf{x}$ can be written as

$$\begin{aligned}\mathbf{y}_i = \mathbf{H}_{i,:}\mathbf{x} &= \sum_{j=1}^{m_B n_B} \mathbf{H}_{ij}\mathbf{x}_j \\ &= \left(\sum_{j \in \mathcal{S}_{\text{pre}}} \mathbf{H}_{ij}\mathbf{x}_j\right) + \mathbf{H}_{ii}\mathbf{x}_i + \left(\sum_{j \in \mathcal{S}_{\text{post}}} \mathbf{H}_{ij}\mathbf{x}_j\right) \end{aligned} \tag{3.2.28}$$

That is, the result in the i^{th} sub-image $\mathbf{y}_i$ of the output can be partitioned into three components: (1) the contribution of the blocks to the left or directly above the i^{th} sub-image, i.e. the pre-blocks, (2) the contribution from the i^{th} sub-image $\mathbf{x}_i$, and (3) the contribution from the blocks to the right or directly below the i^{th} sub-image i.e. the post-blocks. Figure 3.2.6 provides a visual representation of this.

Due to linearity, each term in (3.2.28) can be acted on separately by the matrix

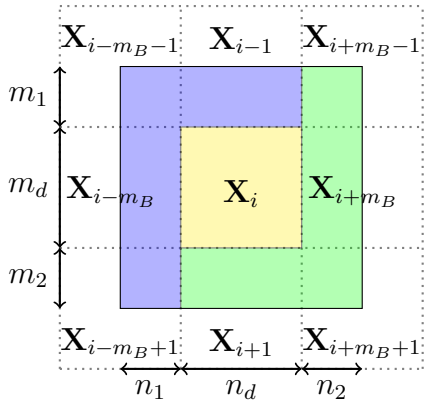


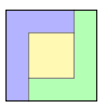
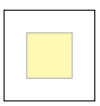
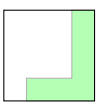
Figure 3.2.7: The blue section on the top left indicates pixels from the relevant portion of sub-images $\mathbf{X}_j$ with $j \in \mathcal{S}_{\text{pre}}$, the green section on the bottom right indicates pixels from the relevant portion of sub-images $\mathbf{X}_j$ with $j \in \mathcal{S}_{\text{post}}$, and the center yellow sub-image is $\mathbf{X}_i$. Each of the sub-images has size $m_d \times n_d$, and the sizes m_1, m_2, n_1, n_2 are defined in (3.2.15).

$\mathbf{H}_{i,:}$, which has the form $\mathbf{y}_i = \mathbf{H}_{i,:}\mathbf{x}$, where

$$\begin{aligned} \mathbf{H}_{i,:} &= \lambda \left(\mathbf{A}^\top \mathbf{A}\right)_{i,:} + \delta \mathbf{L}_{i,:} \\ &= \lambda \left(\mathbf{A}_{:,i}\right)^\top \mathbf{A} + \delta \mathbf{L}_{i,:}. \end{aligned} \tag{3.2.29}$$

The first term corresponds to convolving the entire image with the kernel $\mathbf{a}$, and then performing the partial convolution with $(\mathbf{A}_{:,i})^\top$ which produces the i^{th} block of the output. As shown with the function defined in (3.2.17), the partial convolution $(\mathbf{A}_{:,i})^\top$ only acts on a small portion of its input, whose size and location are defined by the bounds m_3, n_3, m_4, n_4 in (3.2.16). Thus, it is only necessary to apply the initial convolution via $\mathbf{A}$ to produce the portion within these bounds. Figure 3.2.7 depicts the size of the relevant portion of the image for both the pre- and post-sums.

Similar to Equations (3.2.12) and (3.2.14), the components of the image in Figure 3.2.7 can all be acted on separately due to linearity,

$f_{\mathbf{A}}$  $= f_{\mathbf{A}}$  $+ f_{\mathbf{A}}$  $+ f_{\mathbf{A}}$  , (3.2.30)

where the first term corresponds to the sum containing the pre-blocks, and the final

term corresponds to the sum containing the post-blocks, and the center term is exactly the same as (3.2.14). For the first and last terms, the sub-images can be defined as

$$\mathbf{X}_{\text{pre}} = f_{\text{crop}}\left(\begin{bmatrix} \mathbf{X}_{i-m_B-1} & \mathbf{X}_{i-1} & \mathbf{0} \\ \mathbf{X}_{i-m_B} & \mathbf{0} & \mathbf{0} \\ \mathbf{X}_{i-m_B+1} & \mathbf{0} & \mathbf{0} \end{bmatrix}\right), \tag{3.2.31}$$

$$\mathbf{X}_{\text{post}} = f_{\text{crop}}\left(\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{X}_{i+m_B-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{X}_{i+m_B} \\ \mathbf{0} & \mathbf{X}_{i+1} & \mathbf{X}_{i+m_B+1} \end{bmatrix}\right), \tag{3.2.32}$$

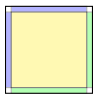
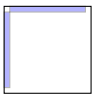
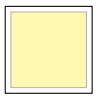
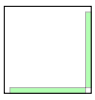
where the center $\mathbf{0} \in \mathbb{R}^{m_d \times n_d}$ is the location of the i^{th} block for both inputs. $\mathbf{X}_{\text{pre}}$ corresponds to the blue part and $\mathbf{X}_{\text{post}}$ corresponds to the green part of Figure 3.2.7. The function $f_{\text{crop}}(\cdot)$ crops the input from all nine sub-images to the appropriate size, which is illustrated by the colored blocks in Equation (3.2.30) and Figure 3.2.7. $\mathbf{X}_{\text{pre}}$ represents the components of $\mathbf{X}_j$ with $j \in \mathcal{S}_{\text{pre}}$, and $\mathbf{X}_{\text{post}}$ represent the components of $\mathbf{X}_j$ with $j \in \mathcal{S}_{\text{post}}$. Thus we can write

$$f_{\mathbf{A}}\left(\mathbf{X}_{\text{pre}}\right), \tag{3.2.33}$$

$$f_{\mathbf{A}}\left(\mathbf{X}_{\text{post}}\right), \tag{3.2.34}$$

which produce the pixels within the bounds (3.2.16). To complete the computation of the first term for the action of $\mathbf{H}_{i,:}$ in (3.2.29), we need only apply $f_{(\mathbf{A}_{:,i})^\top}$ defined in (3.2.17).

Now consider the second term in (3.2.29). $\mathbf{L}_{i,:}$ produces the portion of the i^{th} output block, which can be partitioned into the pre- and post-neighboring blocks, and the center sub-image. That is,

$f_{\mathbf{L}_{i,:}}$  $= f_{\mathbf{L}_{i,:}}$  $+ f_{\mathbf{L}_{i,:}}$  $+ f_{\mathbf{L}_{i,:}}$  , (3.2.35)

where the boundaries (in blue and green) represent a single pixel. As in the case of $\mathbf{A}$, the relevant sub-image blocks can be all considered at one time. The total output for the $\mathbf{L}$ portion of the pre- and post-sums can be computed by

$$\begin{aligned} f_{\mathbf{L},\text{pre}}(\mathbf{X}) &= \sum_{j<i} \mathbf{L}_{ij}\mathbf{x}_j \\ &= f_{\mathbf{L}}\left(\begin{bmatrix} 0 & \mathbf{X}_{i-1}(m_d,:) & 0 \\ \mathbf{X}_{i-m_B}(:,n_d) & \mathbf{0}_{m_d\times n_d} & \mathbf{0} \\ 0 & \mathbf{0} & 0 \end{bmatrix}_{(m_d+2)\times(n_d+2)}\right), \end{aligned} \tag{3.2.36}$$

$$\begin{aligned} f_{\mathbf{L},\text{post}}(\mathbf{X}) &= \sum_{j>i} \mathbf{L}_{ij}\mathbf{x}_j \\ &= f_{\mathbf{L}}\left(\begin{bmatrix} 0 & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{0}_{m_d\times n_d} & \mathbf{X}_{i+m_B}(:,n_d) \\ 0 & \mathbf{X}_{i+1}(m_d,:) & 0 \end{bmatrix}_{(m_d+2)\times(n_d+2)}\right), \end{aligned} \tag{3.2.37}$$

respectively. In both cases, the center $\mathbf{0}$ has dimensions $m_d \times n_d$, and the portions of the $\mathbf{X}_j$'s present are either a row or column. Combining the results from the pre-sum functions (3.2.17), (3.2.33) and (3.2.36), the pre-sum can be computed as

$$f_{\text{pre}}(\mathbf{X}, i) = \lambda f_{\tilde{\mathbf{A}}}\left(f_{\text{pad}}\left(f_{\mathbf{A}}(\mathbf{X}_{\text{pre}})\right)\right) + \delta f_{\mathbf{L},\text{pre}}(\mathbf{X}). \tag{3.2.38}$$

Similarly, combining the post-sum functions (3.2.17), (3.2.34) and (3.2.37), the post-sum can be computed as

$$f_{\text{post}}(\mathbf{X}, i) = \lambda f_{\tilde{\mathbf{A}}}\left(f_{\text{pad}}\left(f_{\mathbf{A}}(\mathbf{X}_{\text{post}})\right)\right) + \delta f_{\mathbf{L},\text{post}}(\mathbf{X}) \tag{3.2.39}$$

With these functions defined, the conditional distributions (3.2.1) are complete.

Chapter 4

Application to high-energy x-ray radiography

The US DOE owns several high energy X-ray imaging systems, one of which is located at the Nevada National Security Site (NNSS). Scientists perform expensive hydrodynamic materials studies experiments at the NNSS and use radiography as a primary means of data acquisition in these experiments. The digital images produced by these machines are high dimensional, so it is important to have a computational framework that can effectively deblur these large images. The desired implementation must be capable of producing $O(100)$ deblurred image samples of size 4096×4096 and provide uncertainty (via standard deviation), which the block Gibbs sampler from from Chapter 3 is capable. This chapter provides background on radiography at the NNSS, image specific kernel estimation and parameter selection methods required in the block Gibbs sampler. Results from debluring a full 4096×4096 image are provided, using the sub-image size that optimizes the time to produce a sample. The sampler is shown to be scalable and the process of finding the optimal sub-image size is detailed.

4.1 Data acquisition and parameter selection

4.1.1 Cygnus Dual Beam Radiography Facility

The Cygnus Dual Beam Radiographic Facility at the NNSS is a high energy X-ray imaging system designed for hydrodynamic materials experiments. Cygnus is located in a laboratory 100 miles northwest of Las Vegas, Nevada, and approximately 1000 feet underground, which provides security and isolation for experimentation. The facility contains two radiographic sources, called Cygnus 1 and Cygnus 2, each of which has a 2.25 MeV endpoint energy, 1 mm source diameter, and a 50 ns pulse length [39], and can be shot independently to produce radiographs at different times [30]. The machine itself is quite large, having a footprint of 22 feet by 92 feet [39]. This system differs considerably in scale and function from the more common medical X-ray machine, and requires a different means of image acquisition and analysis.

X-rays from the Cygnus sources pass through a collimation system in a lead bulkhead wall into the experiment room at a 60° angle [38]. The radiographic object is housed in a containment vessel and identical camera systems opposite the Cygnus sources collect high resolution radiographs of the object, shown in figure 4.1.1. The imaging system functions by using a scintillator, which reacts with X-rays to produce visible light, which is then reflected by an elliptical pellicle mirror through a set of lenses and captured by a CCD camera [25]. The lenses allow for magnification to varying fields of view, as well as adjustment for different scintillators [24]. This complex imaging system is difficult to model, and contributes to the blur in the images.

Since hydrodynamic materials experiments can be extremely costly and Cygnus is a primary means of data acquisition for such experiments at the NNSS, it is imperative to understand its imaging capabilities. In order to study the Cygnus radiographs, a calibration object known as the Luttman Target is used. The Luttman Target is comprised of three calibration objects: (1) a step wedge, (2) an L-rolled edge, and

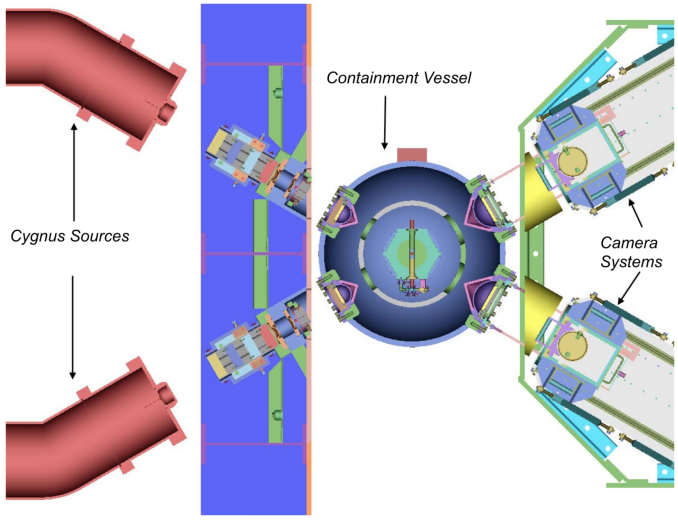


Figure 4.1.1: Top down view of Cygnus source and imaging system (Image from [27]).

(3) an Abel cylinder, shown in Figure 4.1.2. The tantalum step wedge contains ten different steps (two slightly outside the circular window) of varying thickness. The L-rolled edge has a rectangular notch cut out at a 90° angle, to provide a stark contrast between an area where X-rays are completely attenuated (the black region), and where X-rays pass freely (in the cut out). The Abel cylinder is a radially symmetric object comprised of three layered materials with three different bore widths, and a hollow center. Each of these objects can be used with different models to help characterize the imaging system [1, 12, 19, 21].

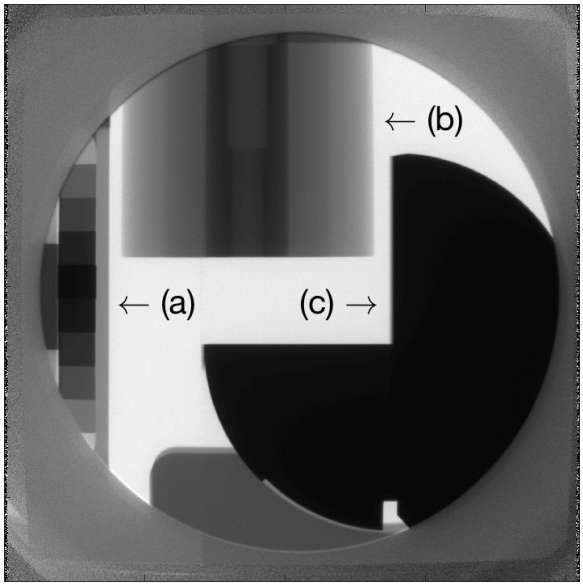
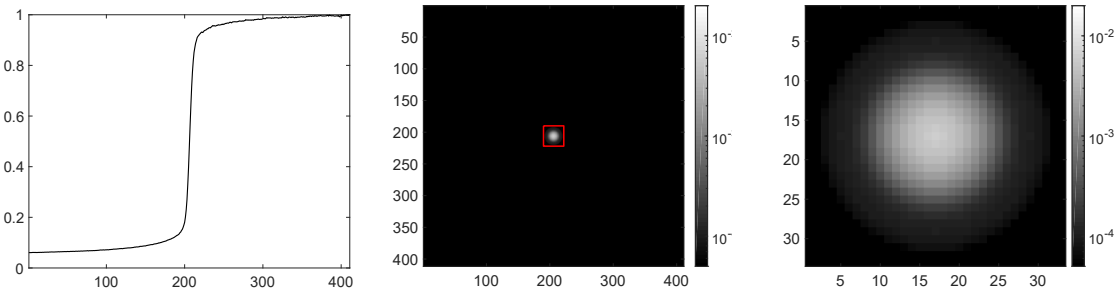


Figure 4.1.2: The Luttman Target consists of three calibration objects: (a) the step wedge, (b) the Abel cylinder, and (c) the L-rolled edge.

4.1.2 Kernel estimation and parameter selection

In order to use Algorithm 2 to solve the inverse problem, it is necessary to estimate the blurring kernel and select parameters that define the operator $\mathbf{H} = \lambda \mathbf{A}^\top \mathbf{A} + \delta \mathbf{L}$. The 2D convolution represented by $\mathbf{A}$ requires a blurring kernel, $\mathbf{a}$. $\mathbf{H}$ also includes the precision parameters λ and δ which provide a scaling between the model and the regularization.

A primary cause of blur in Cygnus radiographs is the shape and extent of the radiation source [12]. Therefore, a reconstruction of the X-ray source profile provides a reasonable approximation for the blurring kernel. Methods for reconstructing the source profile are based on imaging a known calibration object and solving an inverse problem to produce the reconstruction. In [21], the kernel is assumed to have radial symmetry, and utilizes a single line-out (cross-section) from an edge of the L-rolled edge in the Luttman target (see Figure 4.1.2) to compute the reconstruction via MCMC. Figure 4.1.3(a) provides an example line-out used to compute the kernel. Since the size of the kernel has a significant effect on computation time, it is preferable



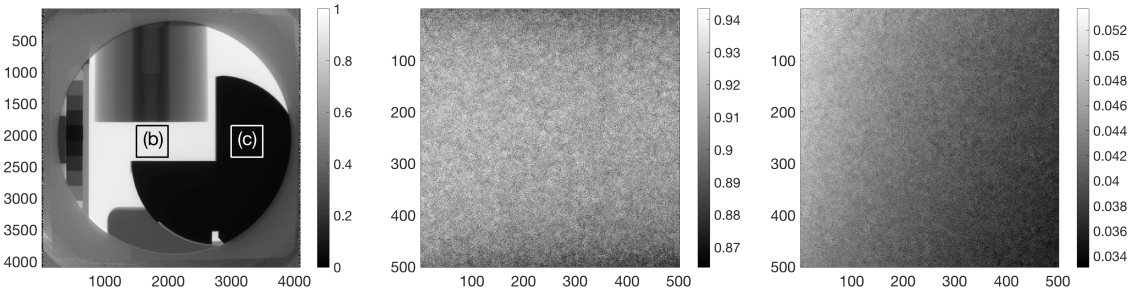
(a) 1D data (b) Full reconstruction (c) Cropped reconstruction

Figure 4.1.3: The line out of the L-rolled edge data used to reconstruct the radially symmetric X-ray intensity profile is shown in (a). The full sized mean reconstruction over 1000 samples of the intensity profile is shown in (b) on a log scale, and the cropped portion identified by the red box in (b) is shown in (c), which is used as the blurring kernel $\mathbf{a}$.

to use the smallest possible kernel. Due to the rapid decay of the kernel from the peak, a significant portion of the kernel reconstruction is essentially zero. Using a threshold of 1% of the peak value of the kernel leads to a kernel of size $m_a = n_a = 33$. Figure 4.1.3(b) shows the full sized mean reconstruction of the kernel from the line-out in Figure 4.1.3(a), and Figure 4.1.3(c) shows the 33×33 cropped portion of the full reconstruction that is used as the kernel $\mathbf{a}$ in the deblurred image reconstructions in Section 4.2.

Once the kernel has been reconstructed, the precision parameters λ and δ can be selected. In Section 2.2, λ^{-1} is introduced as the pixel-wise variance. Therefore, λ can be approximated by calculating a sample variance from the image. For the Luttman target, the sample standard deviations for two subsections of the image are $1.1 \cdot 10^{-2}$ and $3.4 \cdot 10^{-3}$, for white and black portions of the image, respectively, as seen in Figure 4.1.4. Taking a conservative estimate, i.e. maximizing the variance λ^{-1}, λ is set using the larger SD of $1.1 \cdot 10^{-2}$, which gives $\lambda \approx 9 \cdot 10^3$. This choice of λ is specific to this image, and λ varies depending on the image.

To obtain δ, several sub-images are used and a parameter sweep over a log space of the ratio δ/λ is tested. To decide on a practical δ, several regularization parameter



(a) Full image (b) Sample SD $1.1 \cdot 10^{-2}$ (c) Sample SD $3.4 \cdot 10^{-3}$

Figure 4.1.4: Data from a processed image from Cygnus is provided in (a), with two subsections identifying the sub-images shown in (b) and (c). Each figure has a different colorbar.

selection methods are implemented.The first is called the Discrepancy Principle (DP) estimator, which is a standard regularization parameter selection method [11, 18, 42], and attempts to set $||\mathbf{A}\mathbf{x}_\delta - \mathbf{b}||^2$ equal to the noise level, where $\mathbf{x}_\delta$ represents the solution for the corresponding choice of δ. That is, given the function

$$D(\delta) = ||\mathbf{A}\mathbf{x}_\delta - \mathbf{b}||^2 - \frac{m_b \cdot n_b}{\lambda}, \tag{4.1.1}$$

the estimator δ_{DP} satisfies $D(\delta_{\text{DP}}) = 0$. Equivalently, the estimator δ_{DP} can be found by minimizing $(D(\delta))^2$ over δ. The second is called the the Unbiased Predictive Risk Estimator (UPRE), which minimizes the predictive risk, $E\left(||\mathbf{A}\mathbf{x}_\delta - \mathbf{A}\mathbf{x}||^2\right)$, where $E(\cdot)$ represents expectation, and $\mathbf{x}$ represents the truth. Then the UPRE for this problem is given by

$$U(\delta) = ||\mathbf{A}\mathbf{x}_\delta - \mathbf{b}||^2 + \frac{2}{\lambda}\text{tr}\left(\mathbf{A}\left(\lambda\mathbf{A}^\top\mathbf{A} + \delta\mathbf{L}\right)^{-1}\mathbf{A}^\top\right) - \frac{m_b \cdot n_b}{\lambda}, \tag{4.1.2}$$

where $\text{tr}(\cdot)$ denotes the trace, and the estimator δ_{UPRE} is found by minimizing $U(\delta)$ over δ [2]. The trace is estimated via randomized trace estimation (see [2, 42]), which generates a Gaussian vector $\mathbf{v}$ with mean $\mathbf{0}$ and covariance $\mathbf{I}$, and approximates the trace as

$$\text{tr}\left(\mathbf{A}\left(\lambda\mathbf{A}^\top\mathbf{A} + \delta\mathbf{L}\right)^{-1}\mathbf{A}^\top\right) \approx \left(\mathbf{A}^\top\mathbf{v}\right)^\top\left(\lambda\mathbf{A}^\top\mathbf{A} + \delta\mathbf{L}\right)^{-1}\left(\mathbf{A}^\top\mathbf{v}\right). \tag{4.1.3}$$

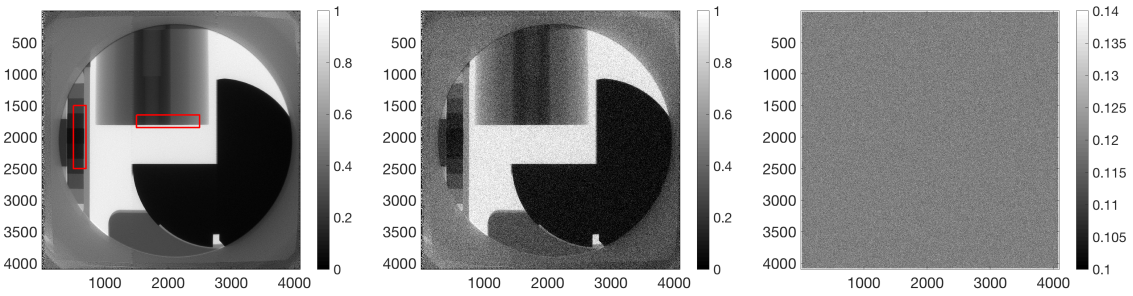
The optimal values δ_{DP} and δ_{UPRE} differed by approximately one order of magnitude, which is expected since both of these regularization parameter selection methods define optimality in different ways [2]. The values of δ_{DP} and δ_{UPRE} varied with the size and location of the sub-images of data used. Based on the calculated values for δ_{DP} and δ_{UPRE} from several sub-images of the data, the value of $\delta = 2.5 \cdot 10^{-3} \cdot \lambda$ was chosen.

4.2 Deblurred image reconstructions

Using the kernel $\mathbf{a}$ and parameters λ, δ from Section 4.1.2, full 4096×4096 Cygnus images were deblurred. A sub-image size of 512×512 was used, for a total of $8 \times 8 = 64$ sub-images (see the center image in Figure 4.4.1). The scalability of the block Gibbs algorithm implemented is shown in Section 4.3 and the choice of the sub-image size is detailed in Section 4.4.

Figures 4.2.1 – 4.2.3 show actual image reconstruction results on a full 4096×4096 Cygnus image. Figure 4.2.1 shows (a) the mean reconstruction over 526 samples, (b) the sample that minimized $||\mathbf{Ax} - \mathbf{b}||$, and (c) the pointwise sample standard deviation. The average standard deviation over all pixels in the reconstruction was found to be about 0.12, which is larger than the prior standard deviation, but features in the mean reconstruction are more distinct, which allow for higher precision on the locations of actual features in the image.

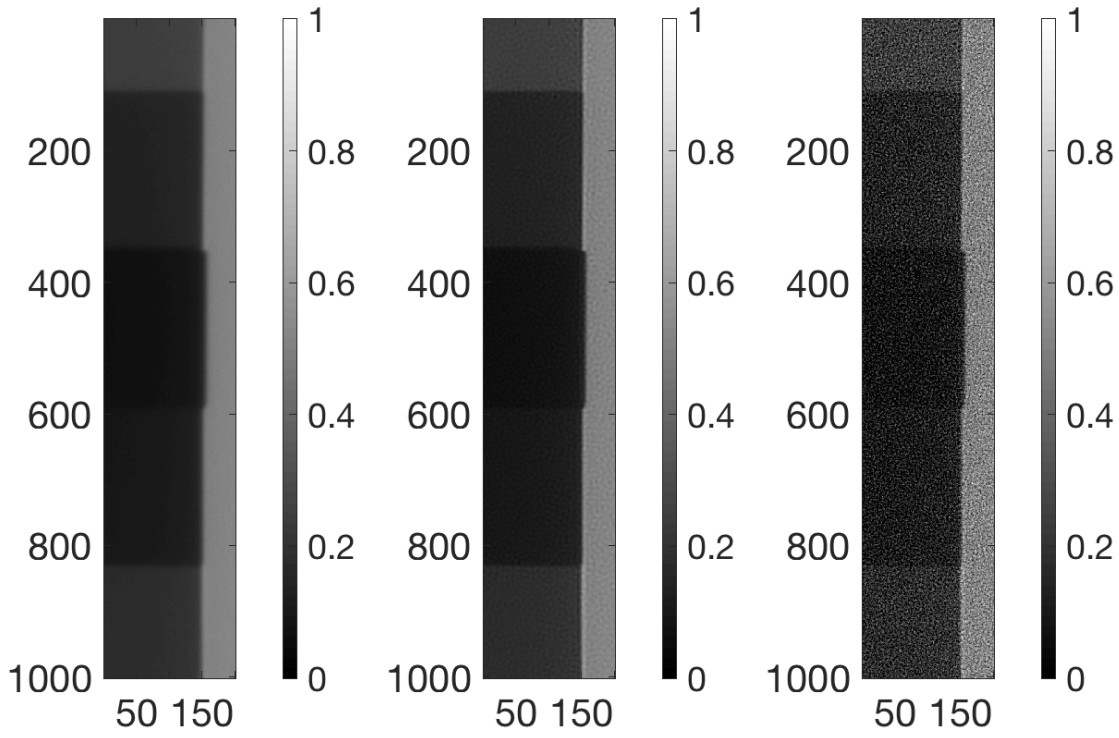
Two sections of the full mean image in Figure 4.2.1 are highlighted with red boxes and enlarged in Figures 4.2.2 and 4.2.3 for comparison between the original blurred image and the reconstructions. In both Figures 4.2.2 and 4.2.3, the (a) original blurred image is shown, with the (b) mean reconstruction and (c) sample that minimized $||\mathbf{Ax} - \mathbf{b}||$. Figure 4.2.2 contains a portion of the step wedge, and it is possible to see that the step edges are sharpened in the reconstructions. Figure



(a) Mean reconstruction (b) Sample # 410 (c) Sample SD

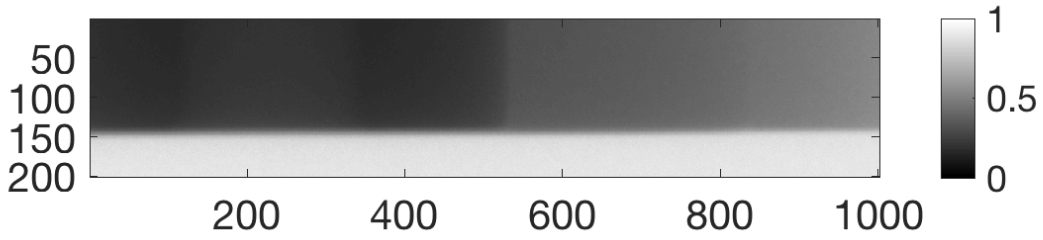
Figure 4.2.1: Results from deblurring a 4096×4096 Cygnus radiograph of the Luttman Target, using a sub-image size of 512×512. The mean reconstruction over 526 samples is shown in (a), the sample that minimizes $||\mathbf{Ax} - \mathbf{b}||$ is shown in (b), and the sample SD is shown in (c). The subsections of the image that are boxed in (a) are shown in more detail in Figures 4.2.2 and 4.2.3.

4.2.3 contains a portion of the Abel cylinder, and while the edge of the cylinder is sharper, the transitions within the cylinder remain smooth as expected and desired. Although the single sample has sharper features than the data, there is a considerable amount of noise in the reconstruction, which indicates room for improvement. Even still, the features of the image in the mean reconstruction show improvement over the data, allowing the locations of features in the image to be more precisely identified.

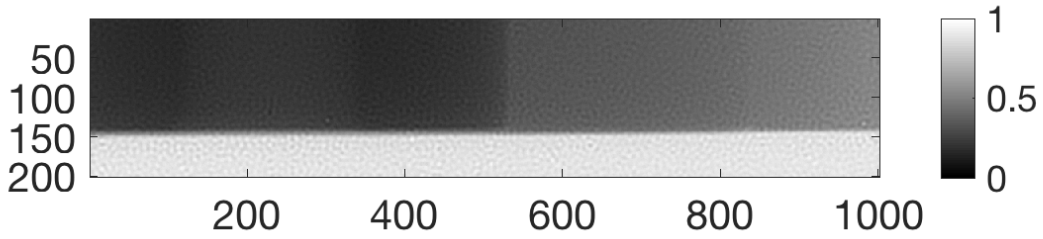


(a) Blurred image (b) Mean reconstruction (c) Sample # 410

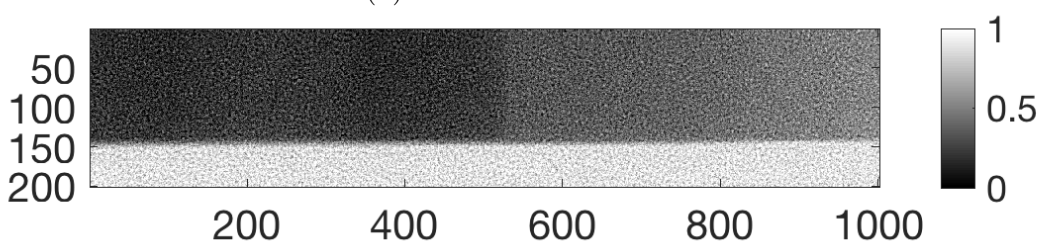
Figure 4.2.2: An enlarged subsection of the full image shown in Figure 4.2.1 showing details in the step wedge.



(a) Blurred image



(b) Mean reconstruction



(c) Sample # 410

Figure 4.2.3: An enlarged subsection of the full image shown in Figure 4.2.1 showing details in the Abel cylinder.

4.3 Efficiency of the sampler

Once the parameter selection process from Section 4.1.2 is complete, the block Gibbs sampler in Algorithm 2 can be used to produce samples from the posterior. In order to ensure that these reconstructions produce meaningful estimates on high dimensional data such as Cygnus radiographs, it is necessary to show that the Gibbs sampler scales well with image size. In general, MCMC methods, including Gibbs samplers, tend to scale poorly with dimension [7, 17, 40]. However, due to the structure and sparsity of the operator $\mathbf{H}$, the block Gibbs sampler is expected to scale well with dimension[28]. One means of assessing the convergence of the MCMC scheme is via the integrated auto-correlation time (IACT), τ_{int}, which provides a measure for the number of effective samples, N_{eff}, out of the total N_e samples, as in Equation (2.3.2). The definitions and numerical approximations found in [40, 44] are used to calculate an estimator, $\hat{\tau}_{\text{int}}$, for the IACT of each pixel.

In order to determine the scaling of the mean IACT with image dimension, a set of numerical experiment was performed. Ideally, an increase in dimension would have little effect on the IACT, indicating the method is appropriate for high dimensional problems. The experiment conducted is as follows. Fix a sub-image size, then generate 1000 samples via the block Gibbs sampler described in Chapter 3 at a specific full image size. Compute the mean IACT over all pixels at this image size, then repeat the experiment for larger full image sizes.

The fixed sub-image size chosen for the experiment is $m_d = n_d = 2^7 = 128$. The full image sizes used are $m_x = n_x = 2^8, \cdots, 2^{11}$, and the corresponding data size $m_b \times n_b$ with a kernel of size $m_a = n_a = 33$ are outlined in Figure 4.3.1, where $m_b = n_b = m_x - m_a + 1$. The positions of the smaller images were chosen to center on a corner of the step wedge, to ensure that an interesting feature existed to deblur in the data. The smallest image size ($m_x = 2^8 = 256$) results in only $m_B \times n_B = 2 \times 2 = 4$ sub-images, while the largest image size ($m_x = 2^{11} = 2048$) results in a total of

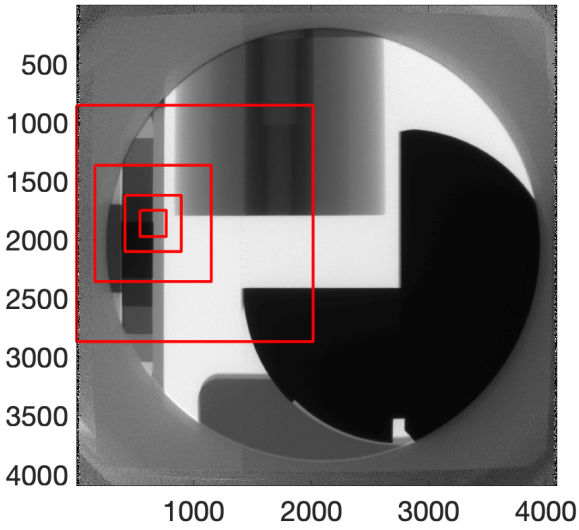


Figure 4.3.1: The image sizes $m_x = n_x = 2^8, \cdots 2^{12}$ were considered, and the corresponding data $\mathbf{B}$ are outlined, which have dimension $m_b = n_b = m_x - m_a + 1$. The smaller images are centered on a corner of the step wedge to ensure at least one interesting feature exists in the image.

$m_B \times n_B = 16 \times 16 = 256$ sub-images.

A related concern with respect to algorithmic efficiency is the burn in time for sampling. In many cases, it can take a significant number of samples before an MCMC chain reaches the stationary distribution [40]. The samples prior to reaching stationarity are generally discarded, and considered the burn in. Since the samples at the full target size are so expensive to compute, avoiding the removal of any samples is preferable. As the target distribution is Gaussian, and seeding the chain with the initial state as the mean, $\mathbf{x}^{(0)} = \mathbf{m}$, it is expected that in this case stationarity is reached effectively immediately. To provide a small check for this, several pixel chains are provided in Figure 4.3.3, which do not seem to exhibit any burn in. The locations of the pixels used are marked for reference in Figure 4.3.2, and the sub-image size was 512×512.

The mean IACTs for different full image sizes, all reconstructed using the fixed sub-image size of 128×128, are provided in Tables 4.3.1 and 4.3.2. For each pixel, an estimate $\hat{\tau}_{\text{int}}$ was calculated, and the global average $\bar{\hat{\tau}}_{\text{int}}$ and maximum $\max\{\hat{\tau}_{\text{int}}\}$

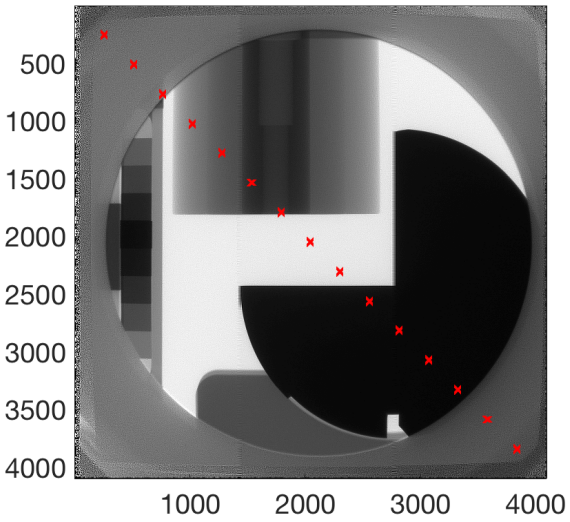


Figure 4.3.2: The location of the line outs of the pixel chains shown in Figure 4.3.3 are marked with red x's.

Image dimension ($m_x = n_x$)	256	512	1024	2048
$2\hat{\tau}_{\text{int}}$	1.0225	1.0372	1.0372	0.9904
$2 \max\{\hat{\tau}_{\text{int}}\}$	3.8977	3.7271	5.2049	10.2248
# sub-images ($m_B \times n_B$)	4	16	64	256

Table 4.3.1: Mean and max IACTs for several image sizes, with a sub-image size of 128×128, with $\hat{\tau}_{\text{int}}$ as in [40].

over all pixels in the image are provided in the table. Perfectly independent samples would expect to have $2\tau_{\text{int}} = 1$, which gives $N_{\text{eff}} = N_e$ (see Equation (2.3.2)). It is possible for $2\tau_{\text{int}} < 1$ which indicates anticorrelation, but in this case is likely simply due to the variability of the estimator $\hat{\tau}_{\text{int}}$ [40]. In all of the cases, twice the mean IACT for each image size is very near 1, indicating that the blocking scheme scales well with image dimension. There is a slight increase in the maximum Thus the block Gibbs sampler is a viable means of producing nearly uncorrelated samples, even with large image sizes.

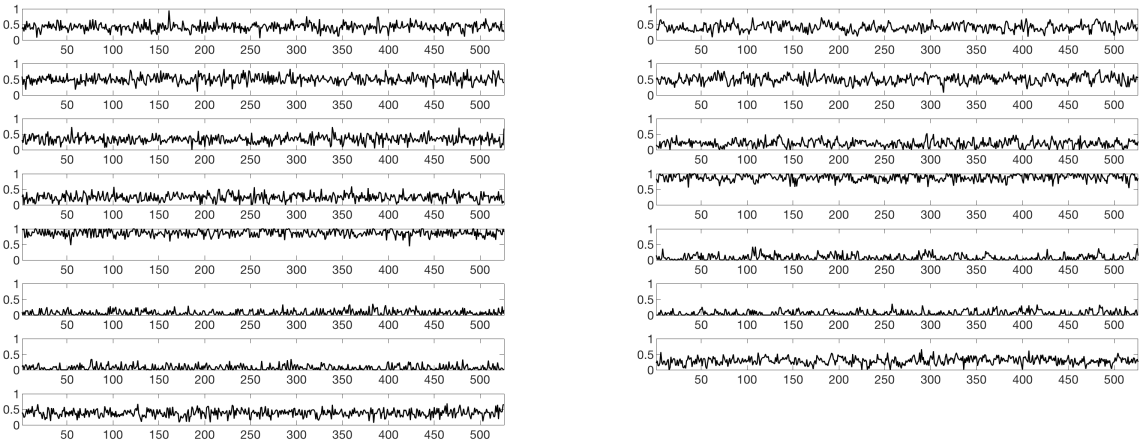


Figure 4.3.3: The line outs from 15 pixel chains are shown. In each case, there appears to be no significant burn in, as expected since the distribution is Gaussian. The locations of the pixels shown are provided in Figure 4.3.2.

Image dimension ($m_x = n_x$)	256	512	1024	2048
$2\bar{\hat{\tau}}_{\text{int}}$	1.0275	1.0136	1.0197	1.0136
$2\max\{\hat{\tau}_{\text{int}}\}$	3.6578	3.7744	4.2303	6.9804
# sub-images ($m_B \times n_B$)	4	16	64	256

Table 4.3.2: Mean and max IACTs for several image sizes, with a sub-image size of 128×128, with $\hat{\tau}_{\text{int}}$ as in [44].

4.4 Optimal sub-image size

The results in Section 4.3 indicate that, for a given sub-image size, the mean IACT over all pixels, $\bar{\hat{\tau}}_{\text{int}}$, scales well with the dimension of the full image. There are, however, various possible sub-image sizes for any full image size. A primary goal for the use of the block Gibbs sampler is to produce deblurred image reconstructions in a timely manner and, thus, it is necessary to find a sub-image size that optimizes the time required to generate a sample. Section 3.1 provides a description of the partitioning scheme used for the following results. The reconstructed image is assumed to have dimension $m_x \times n_x$, which is divisible into equally sized sub-images with dimension $m_d \times n_d$, as shown in Figure 4.4.1. Sub-image sizes that are powers of 2 are considered, because this provides a computational speed increase in the implementation of Algorithm 3.

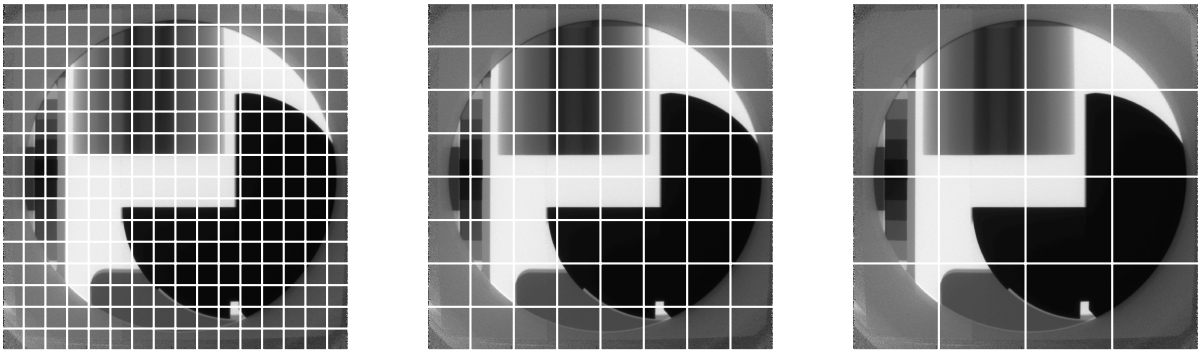
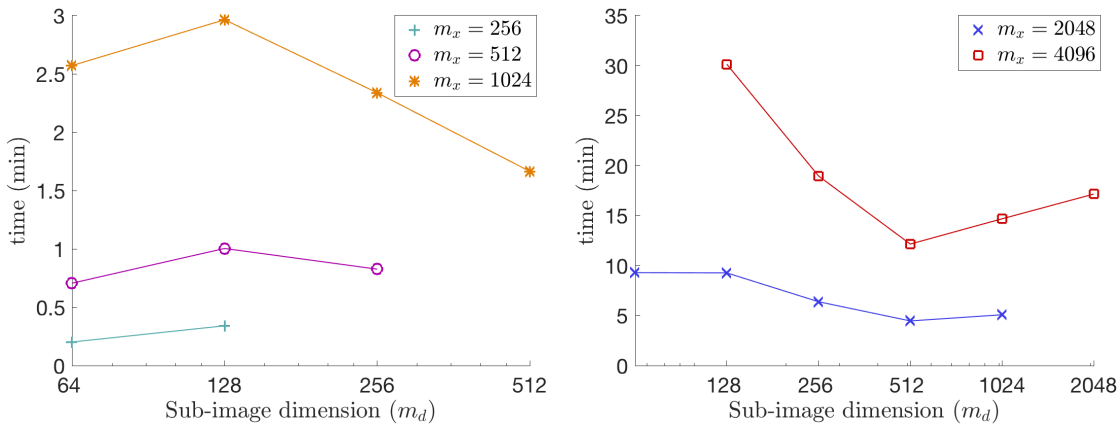


Figure 4.4.1: Three different sub-image partitions for an image of size 4096×4096. The sub-images have size 256×256 (left), 512×512 (center) and 1024×1024 (right).

Different sub-image sizes result in different computation times per sample, due to the iterative nature of the algorithm. Specifically, the inner loop in Algorithm 2 requires successive computations over each sub-image $\mathbf{X}_i^{(k)}$, where using CG to compute the backsolve with $\mathbf{H}_{ii}$ is the computational bottleneck at each iteration. The time required for CG depends on the number of iterations required to reach the desired tolerance and the data dimension. Figure 4.4.2 provides computation times for different image sizes, as well as different sub-image sizes used in the Gibbs sampler. At each full image size ($m_x \times n_x$), five deblurred image samples were produced using the block Gibbs sampler (Algorithm 2) with various sub-image sizes ($m_d \times n_d$), and the average time to produce a sample was calculated. In all cases, the CG tolerance was set as 10^{-10}, and the maximum number of CG iterations was set to $m_d \cdot n_d/16$. For small images ($256 \times 256 - 512 \times 512$), the sub-image size does not have a significant effect on the computation time per iteration, which is evident in the computation times shown in Figure 4.4.2(a). For larger images ($2048 \times 2048 - 4096 \times 4096$), the sub-image size begins to have an appreciable effect and a sub-image size of 512×512 emerges as an optimal choice. This optimum is dependent on the configuration of the CG solver, the kernel size, and the hardware constraints of the computer used to solve the problem. One node from the University of Arizona High Performance Computing (HPC) cluster, was used to produce results, which consists of a 2.3 GHz



(a) Small images (b) Large images

Figure 4.4.2: Average computation times to produce a full sample $\mathbf{X}^{(k)}$ for various image sizes ($m_x \times m_x$) and sub-image sizes ($m_d \times m_d$). The average time to reconstruct larger images (b) depends more heavily on the sub-image size than the time to reconstruct smaller images (a).

Dual 14-core processor with 192 GBs of memory available. Using a sub-image size of 512×512, it is possible to generate a sample $\mathbf{X}^{(k)}$ of size 4096×4096 on the order of 12 minutes. Assuming that $O(100)$ samples are required to estimate pixel-wise standard deviations, the algorithm can solve the inverse problem at this size in about 20 hours.

Chapter 5

Conclusion

In this thesis, a scalable Gibbs sampler for deconvolution-based image deblurring problems was derived, implemented, and tested on radiographic data from the Cygnus Dual Beam Radiographic Facility at the NNSS. The success of the method hinges on exploiting the local nature of deconvolution (which defines the sparsity pattern of the posterior precision) and developing matrix free algorithms. Defining the blocks in the Gibbs sampler as 2D sub-images that are large enough with respect to the blurring kernel results in a scheme where each block depends on at most eight other blocks. Optimizing the blocking structure allowed for faster generation of image reconstructions on Cygnus data.

The discrete 2D deconvolution matrix with a small blurring kernel relative to the image size is sparse and highly structured. A careful partitioning of the image into sub-images results in a 9-block-diagonal matrix. Using a block Gibbs sampler based on this blocking technique was shown to be scalable – as measured by the mean pixel-wise integrated autocorrelation time – to images up to size 4096×4096.

The algorithm was successfully implemented at this size by constructing functions that perform the actions of the required high dimensional matrices. A matrix free implementation was necessary due to the infeasibility of storing and construct-

ing matrices of the dimensions seen in the target application. Such functions were made possible by utilizing previously existing convolution functions and exploiting the linearity of the operator.

The optimal choice for the block size that minimizes solution time is problem-dependent and relies on factors such as the parameters chosen for CG (which affects the number of CG iterations required), the size of the blurring kernel, the statistics of the image, the prior chosen, and the hardware used to compute the solutions. For applications to Cygnus radiographs with a kernel of size 33×33, a sub-image size of 512×512 was found to be optimal, allowing for $O(100)$ full image reconstructions of size 4096×4096 to be produced in under a day with one node on the University of Arizona High Performance Computing cluster. The mean reconstructions display sharper features than the data, while still preserving smooth features in the image. This allows for the locations of features to be measured more precisely. The results shown on images of static calibration objects can be used on images of hydrodynamic materials experiments, providing better quantitative image analysis than currently available.

Bibliography

[1] J. Bardsley and A. Luttman. Dealing with boundary artifacts in MCMC-based deconvolution. *Linear Algebra and its Applications*, 473, 2015.

[2] J. M. Bardsley. *Computational Uncertainty Quantification for Inverse Problems*. SIAM Philadelphia, 2018. ISBN 9781611975376.

[3] P. Barone and A. Frigessi. Improving stochastic relaxation for Gaussian random fields. *Probability in the Engineering and Informational Sciences*, 4(3):369–389, 1990. doi: 10.1017/S0269964800001674.

[4] H. H. Barrett and K. J. Myers. *Foundations of Image Science*. John Wiley and Sons, 1 edition, 2003. ISBN 978-0-471-15300-9.

[5] M. Bertero and P. Boccacci. A simple method for the reduction of boundary effects in the Richardson-Lucy approach to image deconvolution. *Astronomy and Astrophysics*, 437:369–374, 2005. doi: 10.1051/0004-6361:20052717.

[6] A. Beskos, G. Roberts, and A. Stuart. Optimal scalings for local Metropolis-Hastings chains on nonproduct targets in high dimensions. *The Annals of Applied Probability*, 19:863–898, 2009.

[7] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, editors. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall, Oxford, 2011.

[8] A. Buccini, M. Donatelli, and L. Reichel. Iterated Tikhonov regularization with a general penalty term. *Numerical Linear Algebra Applications*, 24, 2017.

[9] H. Chen, C. Wang, Y. Song, and Z. Li. Split Bregmanized anisotropic total variation model for image deblurring. *Journal of Visual Communication and Image Representation*, 31:282–293, 2015.

[10] O. F. Christensen, G. O. Roberts, and J. S. Rosenthal. Scaling limits for the transient phase of local Metropolis-Hastings algorithms. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67:253–268, 2005.

[11] H. Engl, M. Hanke, and A. Neubauer. *Regularization for Inverse Problems*. Kluwer Academic Publishers, 1996. ISBN 0792341570.

[12] M. Fowler, M. Howard, A. Luttman, S. E. Mitchell, and T. J. Webb. A stochastic approach to quantifying the blur with uncertainty for high-energy X-ray imaging systems. *Inverse Problems in Science and Engineering*, 24, 2016.

[13] C. Fox and R. A. Norton. Fast sampling in a linear-Gaussian inverse problem. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):1191 – 1218, July 2016.

[14] C. Fox and A. Parker. Accelerated Gibbs sampling of normal distributions using matrix splittings and polynomials. *Bernoulli*, 23(4B):3711–3743, May 2017.

[15] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall CRC Press, 3 edition, 2013. ISBN 978-1439840955.

[16] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 11 1984.

[17] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Springer-Science+Business Media, 1996.

[18] P. C. Hansen, J. Nagey, and D. P. O'Leary. *Deblurring Images: Matrices, Spectra, and Filtering*. SIAM Philadelphia, 2006. ISBN 978-0-898716-18-4.

[19] M. Howard, M. Fowler, A. Luttman, S. E. Mitchell, and M. C. Hock. Bayesian Abel inversion in quantitative X-ray radiography. *SIAM Journal on Scientific Computing*, 38(3):B396–B413, 2016.

[20] Y. Jiao, Q. Jin, X. Lu, and W. Wang. Alternating direction method of multipliers for linear inverse problems. *SIAM Journal on Numerical Analysis*, 54(4):2114–2137, 2016.

[21] K. T. Joyce, J. M. Bardsley, and A. Luttman. Point spread function estimation in X-ray imaging with partially collapsed Gibbs sampling. *SIAM Journal on Scientific Computing*, 40(3):B766–B787, 2018.

[22] D. Kincaid and W. Cheney. *Numerical Analysis: Mathematics of Scientific Computing*. Brooks/Cole, 3 edition, 2002. ISBN 0-534-38905-8.

[23] L. Ma, L. Xu, and T. Zeng. Low rank prior and total variation regularization for image deblurring. *Journal of Scientific Computing*, 70:1336–1357, 2017.

[24] R. M. Malone, S. A. Baker, K. K. Brown, A. H. Curtis, D. L.Esquibel, D. K. Frayer, B. C. Frogget, M. R. Furlanetto, J. R. Garten, T. J. Haines, R. A. Howe, J. A. Huerta, M. I. Kaufman, N. S. P. King, S. S. Lutz, K. D. McGillivray, and A S. Smith. Design and assembly of a telecentric zoom lens for the Cygnus X-ray source. In *Proceedings of SPIE*, volume 8488 of *84880B*, October 2012. doi: 10.1117/12.929160.

[25] R. M. Malone, S. A. Baker, K. K. Brown, A. H. Curtis, D. L.Esquibel, D. K. Frayer, B. C. Frogget, K. G. Frogget, M. I. Kaufman, A S. Smith, A. Tibbitts, R. A. Howe, J. A. Huerta, K. D. McGillivray, D. W. Droemer, M. D. Crain, T. J. Haines, and N. S. P. King. Telecentric zoom lens designed for the Cygnus X-ray source. In *2013 Abstracts IEEE International Conference on Plasma Science (ICOPS)*, June 2013. doi: 10.1109/PLASMA.2013.6633437.

[26] B. T. McCuistian, D. Moir, E. Rose, H. Bender, C. Carlson, C. Hollabaugh, and R. Trainham. Temporal spot evolution of the DARHT first axis radiographic spot. In *11th European Partical Accelerator Conference*, pages 1206–1208, 01 2008.

[27] S. E. Mitchel. Radiography and image processing. Presentation, 2012.

[28] M. Morzfeld, X.T. Tong, and Y.M. Marzouk. Localization for MCMC: sampling high-dimensional posterior distributions with local structure. *Journal of Computational Physics*, 380:1–28, 2018.

[29] L. G. Multhauf. The LLNL flash X-ray induction linear accelerator (FXR). In *25th Conference on High Speed Photography and Photonics*, September 2002.

[30] D. Nelson, M. Burke, J. Chael, E. Ormond, S. Cordova, and I. Molina. Cygnus source emission. In *2009 IEEE Pulsed Power Conference*, pages 1272–1279, June 2009. doi: 10.1109/PPC.2009.5386413.

[31] M. K. Ng, R. H. Chan, and W. Tang. A fast algorithm for deblurring models with Neumann boundary conditions. *SIAM Journal on Scientific Computing*, 21:851–866, 1999.

[32] E. Ormond, M. Burke, J. Chael, D. Nelson, S. Cordova, I. Molina, and B. Oliver. Cygnus dose symmetry. In *2009 IEEE Pulsed Power Conference*, pages 17–22, June 2009. doi: 10.1109/PPC.2009.5386206.

[33] G. Roberts and J. Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 60:255–268, 1998.

[34] G. Roberts, A. Gelman, and W. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7: 110–120, 1997.

[35] G. O. Roberts and S. Sahu. Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler. *Journal of the Royal Statistical Society Series B*, 59(2):291–317, 1997.

[36] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method*. John Wiley and Sons, 3 edition, 2017. ISBN 9781118632208.

[37] H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Applications*. Chapman and Hall/CRC, 2005.

[38] J. Smith, R. Carlson, R. Fulton, J. Chavez, P. Ortega, R. O'Rear, R. Quicksilver, B. Anderson, D. Henderson, C. Mitton, R. Owens, S. Cordova, J. Maenchen, I. Molina, D. Nelson, and E. Ormond. Cygnus dual beam radiography source. In *Proceedings of the Pulsed Power Conference*, pages 334 – 337. IEEE, 07 2005. doi: 10.1109/PPC.2005.300626.

[39] J. Smith, D. Nelson, E. Ormond, S. Cordova, I. Molina, G. Corrow, M. Hansen, D. Henderson, S. Lutz, and C. Mitton. Cygnus performance in subcritical experiments. In *Proceedings of the 16th IEEE International Pulsed Power Conference*, pages 685–688. IEEE, 2007. doi: 10.1109/PPC.2005.300626.

[40] A. D. Sokal. Monte Carlo methods in statistical mechanics: foundations and new algorithms, 1998.

[41] S. Tao, W. Dong, Z. Xu, and Z. Tang. Fast total variation deconvolution for blurred image contaminated by Poisson noise. *Journal of Visual Communication and Image Representation*, 38:582–594, 2016.

[42] C. R. Vogel. *Computational Methods for Inverse Problems*. SIAM, Philadelphia, 2002.

[43] D. S. Watkins. *Fundamentals of Matrix Computations*. John Wiley and Sons, 3 edition, 2010. ISBN 978-0-470-52833-4.

[44] U. Wolff. Monte Carlo errors with less errors. *Computer Physics Communications*, 156(2):145–153, 2004.

[45] J. Xu, H. B. Chang, and J. Qin. Domain decomposition method for image deblurring. *Journal of Computational and Applied Mathematics*, 271:401–414, 2014.