DOE/NV/03624--0457

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

Dissertation defense

Jesse Adams

April 15, 2019

This document 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--0457

NEVADA NATIONAL NNSS SECURITY SITE

THE UNIVERSITY OF ARIZONA

This work was done in part 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.

Outline

Outline

X-ray Radiography at the NNSS

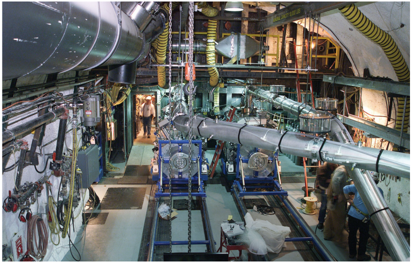
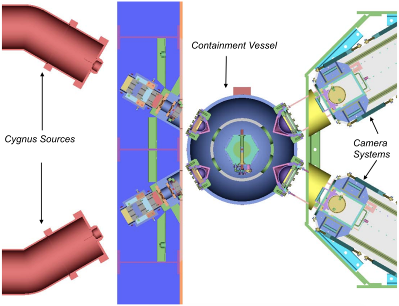


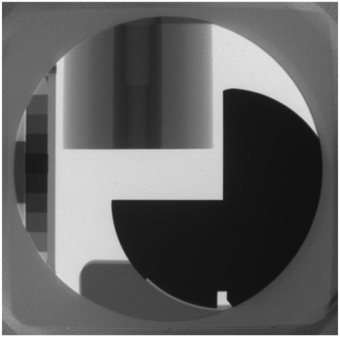
Photo courtesy of www.nnsa.energy.gov/cygnus

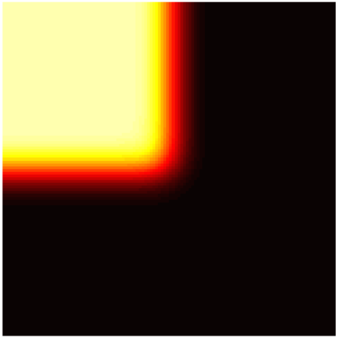


Schematic courtesy of [9]

- Dual-axis, 2.25 MeV X-ray source
- Used to image dynamic and static material studies

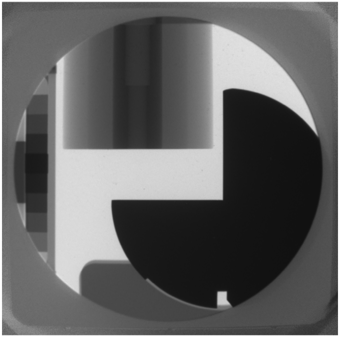
Calibration Imagery

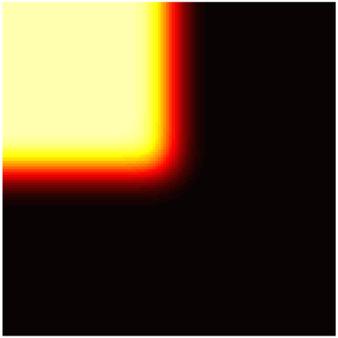




Radiographs of a set of test objects called the “Luttman Target” from the Cygnus Dual Beam Radiography Facility at the NNSS in North Las Vegas.

Calibration Imagery





Radiographs of a set of test objects called the “Luttman Target” from the Cygnus Dual Beam Radiography Facility at the NNSS in North Las Vegas.

Objectives

Quantitative imaging objectives:

- Reduce image blur
- Quantify uncertainties
- Deblur large images in a reasonable amount of time

Solution:

- Use a Bayesian approach to solve the inverse problem
- Produce a mean reconstruction and pixel-wise standard deviation via Markov chain Monte Carlo (MCMC)

Issues:

- Computational tractability is an issue with large images (high dimension)
- Number of samples required for MCMC tends to increase with dimension (i.e. image size) [12]

My contribution:

- MCMC sampler that produces $O(100)$ samples of a full size (4096×4096 pixel) Cygnus image in a day

Objectives

Quantitative imaging objectives:

- Reduce image blur
- Quantify uncertainties
- Deblur large images in a reasonable amount of time

Solution:

- Use a Bayesian approach to solve the inverse problem
- Produce a mean reconstruction and pixel-wise standard deviation via Markov chain Monte Carlo (MCMC)

Issues:

- Computational tractability is an issue with large images (high dimension)
- Number of samples required for MCMC tends to increase with dimension (i.e. image size) [12]

My contribution:

- MCMC sampler that produces $O(100)$ samples of a full size (4096×4096 pixel) Cygnus image in a day

Objectives

Quantitative imaging objectives:

- Reduce image blur
- Quantify uncertainties
- Deblur large images in a reasonable amount of time

Solution:

- Use a Bayesian approach to solve the inverse problem
- Produce a mean reconstruction and pixel-wise standard deviation via Markov chain Monte Carlo (MCMC)

Issues:

- Computational tractability is an issue with large images (high dimension)
- Number of samples required for MCMC tends to increase with dimension (i.e. image size) [12]

My contribution:

- MCMC sampler that produces $O(100)$ samples of a full size (4096×4096 pixel) Cygnus image in a day

Objectives

Quantitative imaging objectives:

- Reduce image blur
- Quantify uncertainties
- Deblur large images in a reasonable amount of time

Solution:

- Use a Bayesian approach to solve the inverse problem
- Produce a mean reconstruction and pixel-wise standard deviation via Markov chain Monte Carlo (MCMC)

Issues:

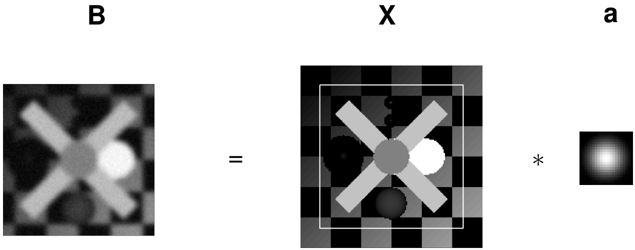
- Computational tractability is an issue with large images (high dimension)
- Number of samples required for MCMC tends to increase with dimension (i.e. image size) [12]

My contribution:

- MCMC sampler that produces $O(100)$ samples of a full size (4096×4096 pixel) Cygnus image in a day

Outline

Discrete 2D Convolution

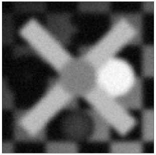
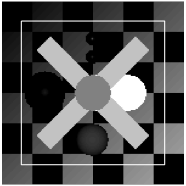
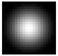


Assuming $\mathbf{a} \in \mathbb{R}^{m_a \times n_a}$ is smaller than $\mathbf{X} \in \mathbb{R}^{m_x \times n_x}$,

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

$\mathbf{B} = \mathbf{a} * \mathbf{X}$ can be written as $\mathbf{b} = \mathbf{A}\mathbf{x}$ by column stacking images [6]

Discrete 2D Convolution

B  = **X**  $*$ **a** 

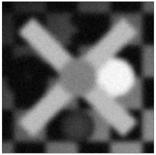
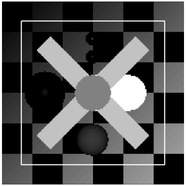
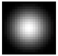
Assuming $\mathbf{a} \in \mathbb{R}^{m_a \times n_a}$ is smaller than $\mathbf{X} \in \mathbb{R}^{m_x \times n_x}$,

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

$\mathbf{B} = \mathbf{a} * \mathbf{X}$ can be written as $\mathbf{b} = \mathbf{A}\mathbf{x}$ by column stacking images [6]

Discrete 2D Convolution

B **X** **a**

 =  $*$ 

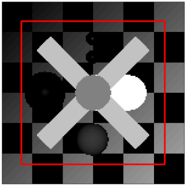
Assuming $\mathbf{a} \in \mathbb{R}^{m_a \times n_a}$ is smaller than $\mathbf{X} \in \mathbb{R}^{m_x \times n_x}$,

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

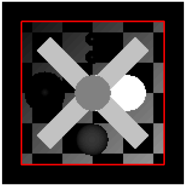
$\mathbf{B} = \mathbf{a} * \mathbf{X}$ can be written as $\mathbf{b} = \mathbf{A}\mathbf{x}$ by column stacking images [6]

Boundary Conditions

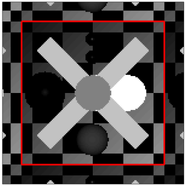
Deconvolution requires boundary conditions (BCs)



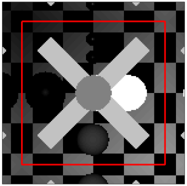
Original Image



Zero BCs



Periodic BCs

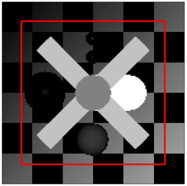


Reflecting BCs

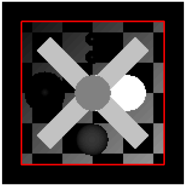
Deconvolution with traditional BCs allow for fast solution via spectral methods [6]

Boundary Conditions

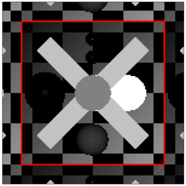
Deconvolution requires boundary conditions (BCs)



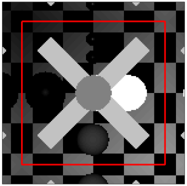
Original Image



Zero BCs



Periodic BCs

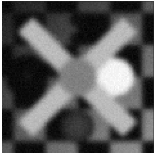


Reflecting BCs

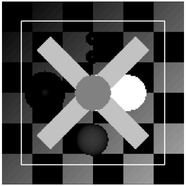
Deconvolution with traditional BCs allow for fast solution via spectral methods [6]

Dealing with Boundary Artifacts

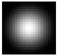
Blurred Image | Reconstructed Image | Kernel



=



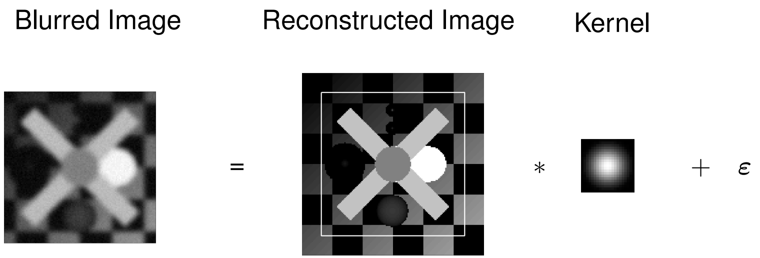
$*$



$+ \quad \varepsilon$

- Assume extended boundary on the reconstruction **x**
- Boundary conditions on extended boundary have small effect on field of view (FOV) [1, 3]
- Underdetermined system

Dealing with Boundary Artifacts



- Assume extended boundary on the reconstruction $\mathbf{x}$
- Boundary conditions on extended boundary have small effect on field of view (FOV) [1, 3]
- Underdetermined system

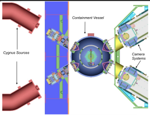
Bayesian Formulation

b

- Measured noisy data, column stacked
- Discretized model form: $\mathbf{b} = \mathbf{A}\mathbf{x} + \varepsilon$
- Assume Gaussian noise: $\varepsilon \sim \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$
- Parameter $\lambda \in \mathbb{R}_{>0}$
- Likelihood: $\pi(\mathbf{b}|\mathbf{x}, \lambda) \sim \mathcal{N}\left(\mathbf{A}\mathbf{x}, \lambda^{-1}\mathbf{I}\right)$

A

- Discretization of imaging system model
- Ill-conditioned [13]



x

- Unknown, reconstructed image, column stacked
- Impose prior $\pi(\mathbf{x}|\delta) \sim \mathcal{N}\left(\mathbf{0}, (\delta\mathbf{L})^{-1}\right)$ [11]
- $\mathbf{L}$: regularization matrix (e.g. Laplacian)
- $\delta \in \mathbb{R}_{>0}$ is a scaling parameter

Bayesian Formulation

Density of **x**:

$$\mathbf{x}|\mathbf{b},\lambda,\delta \sim \mathcal{N}\left(\left(\lambda\mathbf{A}^\top\mathbf{A}+\delta\mathbf{L}\right)^{-1}\lambda\mathbf{A}^\top\mathbf{b},\left(\lambda\mathbf{A}^\top\mathbf{A}+\delta\mathbf{L}\right)^{-1}\right)$$

In the context of the 2D Deconvolution problem:

- $\mathbf{x} \overset{\text{def}}{=} \mathbf{X}(:)$, i.e. a column stack of an image $\mathbf{X} \in \mathbb{R}^{m_x \times n_x}$
- $\mathbf{b} \overset{\text{def}}{=} \mathbf{B}(:)$, i.e. a column stack of an image $\mathbf{B} \in \mathbb{R}^{m_b \times n_b}$
- $\mathbf{H} = \lambda\mathbf{A}^\top\mathbf{A} + \delta\mathbf{L}$ is sparse and large
  - λ: likelihood precision
  - δ: prior precision
  - $\mathbf{A}$: convolution matrix
  - $\mathbf{L}$: regularization matrix
- Want to generate samples from this joint density for **x**

Bayesian Formulation

Density of **x**:

$$\mathbf{x}|\mathbf{b},\lambda,\delta \sim \mathcal{N}\left(\left(\lambda\mathbf{A}^\top\mathbf{A}+\delta\mathbf{L}\right)^{-1}\lambda\mathbf{A}^\top\mathbf{b},\left(\lambda\mathbf{A}^\top\mathbf{A}+\delta\mathbf{L}\right)^{-1}\right)$$

In the context of the 2D Deconvolution problem:

- $\mathbf{x} \stackrel{\text{def}}{=} \mathbf{X}(:)$, i.e. a column stack of an image $\mathbf{X} \in \mathbb{R}^{m_x \times n_x}$
- $\mathbf{b} \stackrel{\text{def}}{=} \mathbf{B}(:)$, i.e. a column stack of an image $\mathbf{B} \in \mathbb{R}^{m_b \times n_b}$
- $\mathbf{H} = \lambda\mathbf{A}^\top\mathbf{A}+\delta\mathbf{L}$ is sparse and large
  - λ: likelihood precision
  - δ: prior precision
  - **A**: convolution matrix
  - **L**: regularization matrix
- Want to generate samples from this joint density for **x**

Gibbs Sampler Refresher

- MCMC algorithm for drawing samples from a joint distribution $p(\mathbf{x}) = p([\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n])$
- Iterative, used to sample large sets of variables
- Useful when the conditional distribution is easy to sample from, but the joint distribution is not
- Given samples $\mathbf{x}^{(k)} = \left[\mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_i^{(k)}, \cdots, \mathbf{x}_n^{(k)}\right]$, then the i^{th} variable (or block of variables) is sampled from the conditional distribution

$$\mathbf{x}_i^{(k)} \sim p\left(\mathbf{x}_i \middle| \mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_{i-1}^{(k)}, \mathbf{x}_{i+1}^{(k-1)}, \cdots \mathbf{x}_n^{(k-1)}\right)$$

- Produces correlated samples with stationary distribution $p(\mathbf{x})$ [5]

Gibbs Sampler Refresher

- MCMC algorithm for drawing samples from a joint distribution $p(\mathbf{x}) = p([\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n])$
- Iterative, used to sample large sets of variables
- Useful when the conditional distribution is easy to sample from, but the joint distribution is not
- Given samples $\mathbf{x}^{(k)} = \left[\mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_i^{(k)}, \cdots, \mathbf{x}_n^{(k)}\right]$, then the i^{th} variable (or block of variables) is sampled from the conditional distribution

$$\mathbf{x}_i^{(k)} \sim p\left(\mathbf{x}_i \middle| \mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_{i-1}^{(k)}, \mathbf{x}_{i+1}^{(k-1)}, \cdots \mathbf{x}_n^{(k-1)}\right)$$

- Produces correlated samples with stationary distribution $p(\mathbf{x})$ [5]

Integrated Autocorrelation Time (IACT)

- Can compute τ_{int} to assess the efficiency of an MCMC algorithm
- IACT tends to increase with dimension: $\tau_{\text{int}} \propto n^p$, $p > 0$
  - e.g. $p = 1$ for random walk Metropolis, $p = 1/2$ for Hamiltonian MCMC, and $p = 1/4$ for Metropolis-adjusted Langevin Algorithm [10]
- Given a sample size, N_e, the effective sample size is

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

  Statistics from N_e correlated samples are similar to those from N_{eff} independent samples [15]
- $N_e = 2\tau_{\text{int}} N_{\text{eff}} \propto n^p$

Integrated Autocorrelation Time (IACT)

- Can compute τ_{int} to assess the efficiency of an MCMC algorithm
- IACT tends to increase with dimension: $\tau_{\text{int}} \propto n^p$, $p > 0$
  - e.g. $p = 1$ for random walk Metropolis, $p = 1/2$ for Hamiltonian MCMC, and $p = 1/4$ for Metropolis-adjusted Langevin Algorithm [10]
- Given a sample size, N_e, the effective sample size is

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

  Statistics from N_e correlated samples are similar to those from N_{eff} independent samples [15]
- $N_e = 2\tau_{\text{int}} N_{\text{eff}} \propto n^p$

Outline

Issues and Objectives

Issues:

- Data-driven BCs: cannot deconvolve via FFT
- Large image size: cannot sample directly
- MCMC expected to converge slowly: $N_e \propto \tau_{\text{int}} \propto n^p$

Objective:

- A sampler with IACT independent of image size, using data-driven BCs

Outline

Gibbs Sampling Example

Consider a small example where $\mathbf{x}$ has four components, $\mathbf{x} = [x_1, x_2, x_3, x_4]$.

- Independent samples
  1. $\left[x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right] \sim p(\mathbf{x})$
- Standard Gibbs
  1. $x_1^{(k)} \sim p\left(x_1 \,\middle|\, x_2^{(k-1)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  2. $x_2^{(k)} \sim p\left(x_2 \,\middle|\, x_1^{(k)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  3. $x_3^{(k)} \sim p\left(x_3 \,\middle|\, x_1^{(k)}, x_2^{(k)}, x_4^{(k-1)}\right)$
  4. $x_4^{(k)} \sim p\left(x_4 \,\middle|\, x_1^{(k)}, x_2^{(k)}, x_3^{(k)}\right)$

  Correlated samples

Gibbs Sampling Example

Consider a small example where $\mathbf{x}$ has four components, $\mathbf{x} = [x_1, x_2, x_3, x_4]$.

- Independent samples
  1. $\left[x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right] \sim p(\mathbf{x})$
- Standard Gibbs
  1. $x_1^{(k)} \sim p\left(x_1 \,\middle|\, x_2^{(k-1)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  2. $x_2^{(k)} \sim p\left(x_2 \,\middle|\, x_1^{(k)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  3. $x_3^{(k)} \sim p\left(x_3 \,\middle|\, x_1^{(k)}, x_2^{(k)}, x_4^{(k-1)}\right)$
  4. $x_4^{(k)} \sim p\left(x_4 \,\middle|\, x_1^{(k)}, x_2^{(k)}, x_3^{(k)}\right)$

  Correlated samples

Gibbs Sampling Example

Consider a small example where $\mathbf{x}$ has four components, $\mathbf{x} = [x_1, x_2, x_3, x_4]$.

- Independent samples
  1. $\left[x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right] \sim p(\mathbf{x})$
- Standard Gibbs
  1. $x_1^{(k)} \sim p\left(x_1 \,\middle|\, x_2^{(k-1)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  2. $x_2^{(k)} \sim p\left(x_2 \,\middle|\, x_1^{(k)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  3. $x_3^{(k)} \sim p\left(x_3 \,\middle|\, x_1^{(k)}, x_2^{(k)}, x_4^{(k-1)}\right)$
  4. $x_4^{(k)} \sim p\left(x_4 \,\middle|\, x_1^{(k)}, x_2^{(k)}, x_3^{(k)}\right)$

  Correlated samples

Gibbs Sampling Example

Block Gibbs sampling:

- One Block of 2
  1. $x_1^{(k)} \sim p\left(x_1 \,\middle|\, x_2^{(k-1)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  2. $x_2^{(k)} \sim p\left(x_2 \,\middle|\, x_1^{(k)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  3. $\left[x_3^{(k)}, x_4^{(k)}\right] \sim p\left([x_3, x_4] \,\middle|\, x_1^{(k)}, x_2^{(k)}\right)$
- Two blocks of 2
  1. $\left[x_1^{(k)}, x_2^{(k)}\right] \sim p\left([x_1, x_2] \,\middle|\, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  2. $\left[x_3^{(k)}, x_4^{(k)}\right] \sim p\left([x_3, x_4] \,\middle|\, x_1^{(k)}, x_2^{(k)}\right)$
- One block of 3
  1. $x_1^{(k)} \sim p\left(x_1 \,\middle|\, x_2^{(k-1)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  2. $\left[x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right] \sim p\left([x_2, x_3, x_4] \,\middle|\, x_1^{(k)}\right)$

The stationary distribution in all cases is $p(\mathbf{x})$

Gibbs Sampling Example

Block Gibbs sampling:

- One Block of 2
  1. $x_1^{(k)} \sim p\left(x_1 \,\middle|\, x_2^{(k-1)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  2. $x_2^{(k)} \sim p\left(x_2 \,\middle|\, x_1^{(k)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  3. $\left[x_3^{(k)}, x_4^{(k)}\right] \sim p\left([x_3, x_4] \,\middle|\, x_1^{(k)}, x_2^{(k)}\right)$
- If x_1 is independent of x_3, x_4
  1. $x_1^{(k)} \sim p\left(x_1 \,\middle|\, x_2^{(k-1)}\right) = p\left(x_1 \,\middle|\, x_2^{(k-1)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  2. $x_2^{(k)} \sim p\left(x_2 \,\middle|\, x_1^{(k)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  3. $\left[x_3^{(k)}, x_4^{(k)}\right] \sim p\left([x_3, x_4] \,\middle|\, x_2^{(k)}\right) = p\left([x_3, x_4] \,\middle|\, x_1^{(k)}, x_2^{(k)}\right)$
- This can improve efficiency (w.r.t. IACT) [10]

Gibbs Sampling Example

Block Gibbs sampling:

- One Block of 2
  1. $x_1^{(k)} \sim p\left(x_1 \,\middle|\, x_2^{(k-1)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  2. $x_2^{(k)} \sim p\left(x_2 \,\middle|\, x_1^{(k)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  3. $\left[x_3^{(k)}, x_4^{(k)}\right] \sim p\left([x_3, x_4] \,\middle|\, x_1^{(k)}, x_2^{(k)}\right)$
- If x_1 is independent of x_3, x_4
  1. $x_1^{(k)} \sim p\left(x_1 \,\middle|\, x_2^{(k-1)}\right) = p\left(x_1 \,\middle|\, x_2^{(k-1)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  2. $x_2^{(k)} \sim p\left(x_2 \,\middle|\, x_1^{(k)}, x_3^{(k-1)}, x_4^{(k-1)}\right)$
  3. $\left[x_3^{(k)}, x_4^{(k)}\right] \sim p\left([x_3, x_4] \,\middle|\, x_2^{(k)}\right) = p\left([x_3, x_4] \,\middle|\, x_1^{(k)}, x_2^{(k)}\right)$
- This can improve efficiency (w.r.t. IACT) [10]

In the context of imaging, $\mathbf{x}$ consists of all the pixels x_i.

Gibbs Sampling in images

- Want to generate samples $\mathbf{x}^{(k)} \,\middle|\, \mathbf{b}, \lambda, \delta \sim \mathcal{N}\left(\mathbf{m}, \mathbf{H}^{-1}\right)$ by sampling

$$\mathbf{x}_i^{(k)} \,\middle|\, \mathbf{b}, \lambda, \delta \sim p\left(\mathbf{x}_i \,\middle|\, \mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_{i-1}^{(k)}, \mathbf{x}_{i+1}^{(k-1)}, \cdots \mathbf{x}_n^{(k-1)}, \mathbf{b}, \lambda, \delta\right)$$

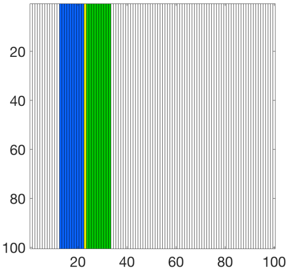
- How should $\mathbf{X}$ be broken up into smaller sub-images $\mathbf{X}_i$?

Gibbs Sampling in images

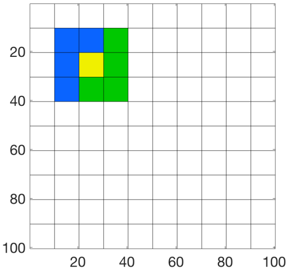
- Want to generate samples $\mathbf{x}^{(k)} \,\big|\, \mathbf{b}, \lambda, \delta \sim \mathcal{N}\left(\mathbf{m}, \mathbf{H}^{-1}\right)$ by sampling

$$\mathbf{x}_i^{(k)} \,\Big|\, \mathbf{b}, \lambda, \delta \sim p\left(\mathbf{x}_i \Big| \mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_{i-1}^{(k)}, \mathbf{x}_{i+1}^{(k-1)}, \cdots \mathbf{x}_n^{(k-1)}, \mathbf{b}, \lambda, \delta\right)$$

- How should $\mathbf{X}$ be broken up into smaller sub-images $\mathbf{X}_i$?



(a) Column based components



(b) Sub-image components

Gibbs Sampling in images

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

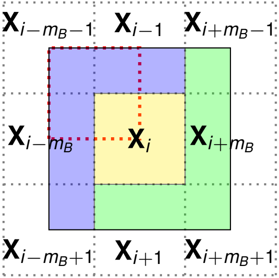
- Partition $\mathbf{X}$ into a grid of $m_B \times n_B$ sub-images
- Generate samples of $\mathbf{X}_i$ conditioned only on 8 neighbors:

$$\begin{aligned} \mathbf{x}_i^{(k)} &\sim p\left(\mathbf{x}_i \,\middle|\, \left\{\mathbf{x}_j^{(k)} \,\middle|\, j \in \mathcal{S}_{\text{pre}}\right\}, \left\{\mathbf{x}_j^{(k-1)} \,\middle|\, j \in \mathcal{S}_{\text{post}}\right\}, \mathbf{b}\right) \\ &= p\left(\mathbf{x}_i \,\middle|\, \left\{\mathbf{x}_j^{(k)} \,\middle|\, j < i\right\}, \left\{\mathbf{x}_j^{(k-1)} \,\middle|\, j > i\right\}, \mathbf{b}\right) \end{aligned}$$

where $\mathbf{x}_i = \mathbf{X}(:)$ is marked in yellow, $\mathcal{S}_{\text{pre}}$ corresponds to blue, and $\mathcal{S}_{\text{post}}$ corresponds to green

Gibbs Sampling in images

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



- Partition $\mathbf{X}$ into a grid of $m_B \times n_B$ sub-images
- Generate samples of $\mathbf{X}_i$ conditioned only on 8 neighbors:

$$\begin{aligned} \mathbf{x}_i^{(k)} &\sim p\left(\mathbf{x}_i \,\middle|\, \left\{\mathbf{x}_j^{(k)} \,\middle|\, j \in \mathcal{S}_{\text{pre}}\right\}, \left\{\mathbf{x}_j^{(k-1)} \,\middle|\, j \in \mathcal{S}_{\text{post}}\right\}, \mathbf{b}\right) \\ &= p\left(\mathbf{x}_i \,\middle|\, \left\{\mathbf{x}_j^{(k)} \,\middle|\, j < i\right\}, \left\{\mathbf{x}_j^{(k-1)} \,\middle|\, j > i\right\}, \mathbf{b}\right) \end{aligned}$$

where $\mathbf{x}_i = \mathbf{X}(:)$ is marked in yellow, $\mathcal{S}_{\text{pre}}$ corresponds to blue, and $\mathcal{S}_{\text{post}}$ corresponds to green

Block Gibbs Algorithm

Inputs:

- Precision $\mathbf{H} = \lambda \mathbf{A}^\top \mathbf{A} + \delta \mathbf{L}$
- Mean $\mathbf{m}$
- Initial state $\mathbf{x}_0$
- # sub-images $\mathbf{n}_B = [m_B, n_B]$
- # samples N_e

Sampling:

$$\mathbf{x}_i^{(k)} \,\Big|\, \left\{ \mathbf{x}_j^{(k)} \,\Big|\, j \in \mathcal{S}_{\text{pre}} \right\}, \left\{ \mathbf{x}_j^{(k-1)} \,\Big|\, j \in \mathcal{S}_{\text{post}} \right\}, \mathbf{b}, \lambda, \delta$$
$$\sim \mathcal{N}\Bigg(\mathbf{m}_i - \Bigg(\sum_{j \in \mathcal{S}_{\text{post}}} \mathbf{H}_{ij} \left(\mathbf{x}_j^{(k-1)} - \mathbf{m}_j \right) + \sum_{j \in \mathcal{S}_{\text{pre}}} \mathbf{H}_{ij} \left(\mathbf{x}_j^{(k)} - \mathbf{m}_j \right) \Bigg), \mathbf{H}_{ii}^{-1} \Bigg)$$

Output: Samples $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N_e)}$ distributed $\mathcal{N}(\mathbf{m}, \mathbf{H}^{-1})$

Image $\mathbf{X}$ with sub-images $\mathbf{X}_i$ (yellow), $\mathbf{X}_j$ with $j \in \mathcal{S}_{\text{pre}}$ (blue), and $\mathbf{X}_j$ with $j \in \mathcal{S}_{\text{post}}$ (green) highlighted.

Outline

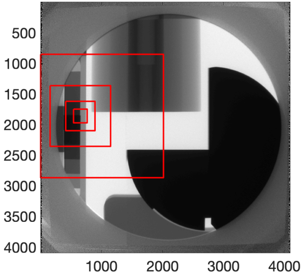
Efficiency and Scalability

- How does τ_{int} change with image size?
- Is there an optimal sub-image size to minimize computation time per sample?

IACT Experiment

Reminder: $N_e = 2\tau_{\text{int}} N_{\text{eff}}$, $\tau_{\text{int}} \propto n^p$

- Fix sub-image size of 128×128
- Consider full image sizes of $m_x \times n_x = 256 \times 256, 512 \times 512, 1024 \times 1024, 2048 \times 2048$
- Generate 1000 samples at each size, and calculate mean sample IACT, $\overline{\widehat{\tau}}_{\text{int}}$



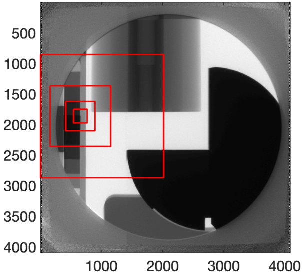
Data **B** outlined at various sizes.

Image dimension ($m_x = n_x$)	256	512	1024	2048
$2\overline{\widehat{\tau}}_{\text{int}}$	1.0225	1.0372	1.0372	0.9904

IACT Experiment

Reminder: $N_e = 2\tau_{\text{int}} N_{\text{eff}}$, $\tau_{\text{int}} \propto n^p$

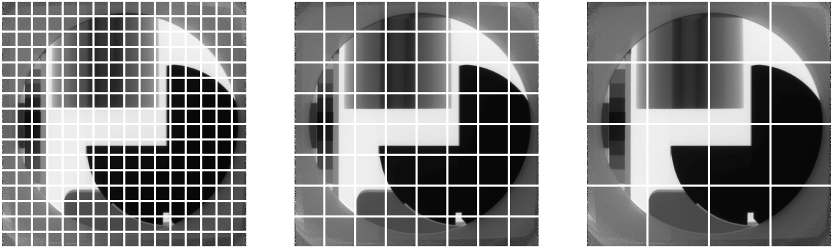
- Fix sub-image size of 128×128
- Consider full image sizes of $m_x \times n_x = 256 \times 256, 512 \times 512, 1024 \times 1024, 2048 \times 2048$
- Generate 1000 samples at each size, and calculate mean sample IACT, $\widehat{\overline{\tau}}_{\text{int}}$



Data **B** outlined at various sizes.

Image dimension ($m_x = n_x$)	256	512	1024	2048
$2\widehat{\overline{\tau}}_{\text{int}}$	1.0225	1.0372	1.0372	0.9904

Optimal Sub-image Size

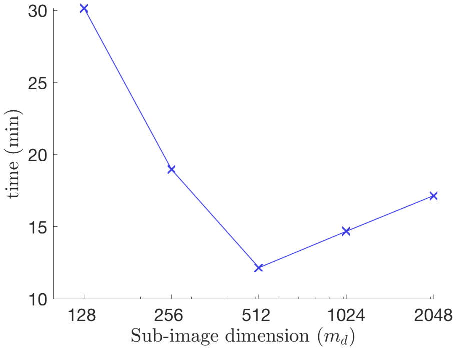


- Full image size: 4096×4096
- Sub-image sizes: 256×256 (left), 512×512 (center) and 1024×1024 (right)
- Regardless of sub-image size, still generating samples with stationary distribution $p(\mathbf{x}|\mathbf{b}, \lambda, \delta)$

Sub-image Computation Times

Experiment:

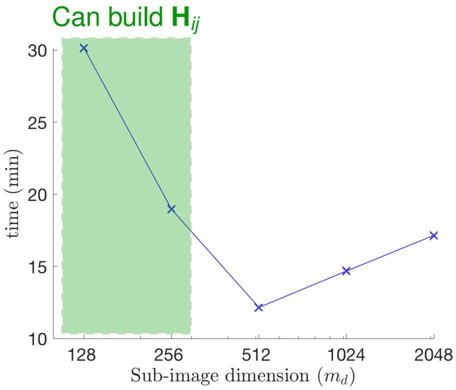
- Consider full image size $m_x \times n_x = 4096 \times 4096$
- Vary sub-image size, compute average computation times over five samples



Sub-image Computation Times

Experiment:

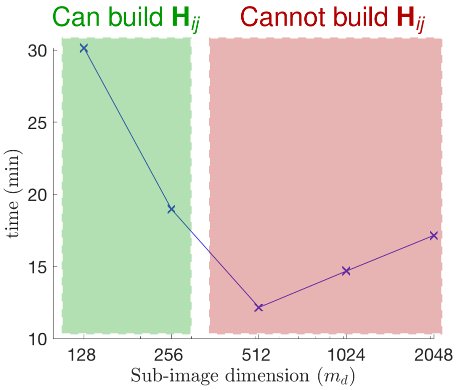
- Consider full image size $m_x \times n_x = 4096 \times 4096$
- Vary sub-image size, compute average computation times over five samples



Sub-image Computation Times

Experiment:

- Consider full image size $m_x \times n_x = 4096 \times 4096$
- Vary sub-image size, compute average computation times over five samples



Block Gibbs Computations

Sampling a sub-image requires sub-matrices $\mathbf{H}_{ij} \in \mathbb{R}^{m_d n_d \times m_d n_d}$:

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

which has three main components:

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

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

$$f_{\text{post}}(\mathbf{y}, i) = \sum_{j \in \mathcal{S}_{\text{post}}} \mathbf{H}_{ij} \mathbf{y}_j \quad , \quad f_{\text{pre}}(\mathbf{y}, i) = \sum_{j \in \mathcal{S}_{\text{pre}}} \mathbf{H}_{ij} \mathbf{y}_j \tag{3}$$

Block Gibbs Computations

Sampling a sub-image requires sub-matrices $\mathbf{H}_{ij} \in \mathbb{R}^{m_d n_d \times m_d n_d}$:

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

which has three main components:

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

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

$$f_{\text{post}}(\mathbf{y}, i) = \sum_{j \in \mathcal{S}_{\text{post}}} \mathbf{H}_{ij} \mathbf{y}_j \quad , \quad f_{\text{pre}}(\mathbf{y}, i) = \sum_{j \in \mathcal{S}_{\text{pre}}} \mathbf{H}_{ij} \mathbf{y}_j \tag{3}$$

Block Gibbs Computations (cont.)

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

Each of these components contains at least one sub-matrix $\mathbf{H}_{ij}$, which has the form

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

and provides the portion of the blur in the i^{th} output sub-image due to the j^{th} input sub-image.

In order to write the functions (1) – (3), it is necessary to also have functions $f_{\mathbf{A}_{:,j}}()$, $f_{(\mathbf{A}_{:,i})^{\top}}()$, and $f_{\mathbf{L}_{ij}}()$. Then

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

Block Gibbs Computations (cont.)

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

Each of these components contains at least one sub-matrix $\mathbf{H}_{ij}$, which has the form

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

and provides the portion of the blur in the i^{th} output sub-image due to the j^{th} input sub-image.

In order to write the functions (1) – (3), it is necessary to also have functions $f_{\mathbf{A}_{:,j}}()$, $f_{(\mathbf{A}_{:,i})^{\top}}()$, and $f_{\mathbf{L}_{ij}}()$. Then

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

Efficient Computation of f_{diag} and f_{sqrt}

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

- Can use the forward function $f_{\mathbf{H}_{ii}}(\mathbf{Y}_i)$ in combination with an iterative algorithm for solving $\mathbf{Ax} = \mathbf{b}$ such as Conjugate Gradient (CG) [8, 14]

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

- Additional assumption: can compute a square root of $\mathbf{L}_{ii} = (\mathbf{L}_{:,i})^{\top/2}(\mathbf{L}_{:,i})^{1/2}$
- Generate

$$\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_1)\ ,\ \mathbf{z}_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2)\ ,$$

  where the size of $\mathbf{I}_1$, $\mathbf{I}_2$ depend on the sub-image
- Then

$$\sqrt{\lambda} f_{(\mathbf{A}_{:,i})^\top}(\mathbf{Z}_1) + \sqrt{\delta} f_{(\mathbf{L}_{:,i})^{\top/2}}(\mathbf{Z}_2) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{H}_{ii}\right)$$

Efficient Computation of f_{diag} and f_{sqrt}

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

- Can use the forward function $f_{\mathbf{H}_{ii}}(\mathbf{Y}_i)$ in combination with an iterative algorithm for solving $\mathbf{Ax} = \mathbf{b}$ such as Conjugate Gradient (CG) [8, 14]

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

- Additional assumption: can compute a square root of $\mathbf{L}_{ii} = (\mathbf{L}_{:,i})^{\top/2}(\mathbf{L}_{:,i})^{1/2}$
- Generate

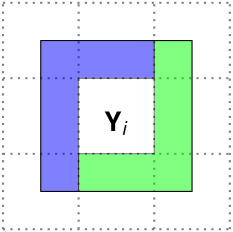
$$\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_1)\ ,\ \mathbf{z}_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2)\ ,$$

  where the size of $\mathbf{I}_1$, $\mathbf{I}_2$ depend on the sub-image
- Then

$$\sqrt{\lambda} f_{(\mathbf{A}_{:,i})^\top}(\mathbf{Z}_1) + \sqrt{\delta} f_{(\mathbf{L}_{:,i})^{\top/2}}(\mathbf{Z}_2) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{H}_{ii}\right)$$

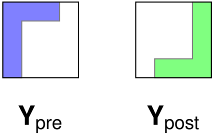
Efficient Computation of the pre- and post-sums

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



Assuming that $\mathbf{L}_{i,:}\mathbf{y}$ can be calculated with $f_{\mathbf{L}}()$ similar to $\mathbf{A}_{i,:}\mathbf{y}$, then

$$f_{\text{pre}}(\mathbf{y}, i) = \lambda f_{(\mathbf{A}_{:,i})^\top}\left(f_{\mathbf{A}}\left(\mathbf{Y}_{\text{pre}}\right)\right) + \delta f_{\mathbf{L}}(\mathbf{Y}_{\text{pre}})$$



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

Outline

Problem

Density of $\mathbf{x}$:

$$\mathbf{x}|\mathbf{b}, \lambda, \delta \sim \mathcal{N}\left(\left(\lambda \mathbf{A}^{\top}\mathbf{A} + \delta \mathbf{L}\right)^{-1} \lambda \mathbf{A}^{\top}\mathbf{b}, \left(\lambda \mathbf{A}^{\top}\mathbf{A} + \delta \mathbf{L}\right)^{-1}\right)$$

Need to define the kernel and precision parameters:

- $\mathbf{a}$: convolution kernel
- λ: likelihood precision
- δ: prior precision

Problem

Density of **x**:

$$\mathbf{x}|\mathbf{b}, \lambda, \delta \sim \mathcal{N}\left(\left(\lambda \mathbf{A}^\top \mathbf{A} + \delta \mathbf{L}\right)^{-1} \lambda \mathbf{A}^\top \mathbf{b}, \left(\lambda \mathbf{A}^\top \mathbf{A} + \delta \mathbf{L}\right)^{-1}\right)$$

Need to define the kernel and precision parameters:

- **a**: convolution kernel
- λ: likelihood precision
- δ: prior precision

Kernel

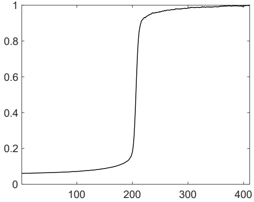
Idea:

- Blur mostly due to X-ray intensity profile [4]
- Use a model that gives the kernel from an edge in the image [7]

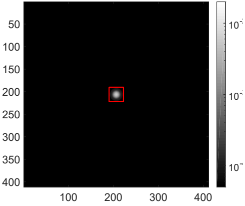
Kernel

Idea:

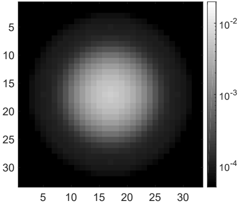
- Blur mostly due to X-ray intensity profile [4]
- Use a model that gives the kernel from an edge in the image [7]



(a) 1D data



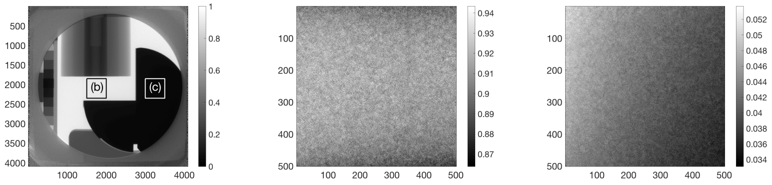
(b) Full kernel



(c) Cropped kernel

(a) Line out from L-rolled edge in Luttman target
(b) Mean reconstruction of X-ray intensity profile over 1000 samples
(c) Cropped portion of (b) in the red box, used as the kernel **a**

Parameter Selection



(a) Full image

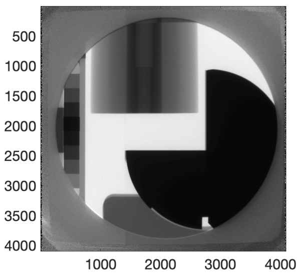
(b) Pixel-wise var: $\approx 1.1 \cdot 10^{-4}$

(c) Pixel-wise var: $\approx 1.1 \cdot 10^{-5}$

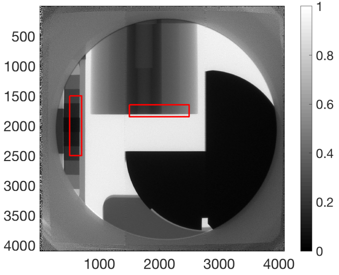
Data from a processed image from Cygnus is provided in (a), with two subsections identifying the sub-images shown in (b) and (c). Each figure has a different colorbar

- $\lambda \approx 9000$ defined by variance in the image
- $\delta \approx 22$ chosen by a parameter sweep over smaller portions of the image, and parameter estimation techniques [6, 13]

Deconvolution Results



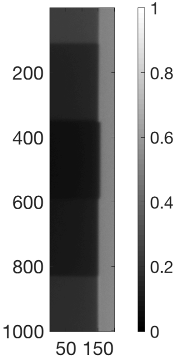
(a) Blurred data



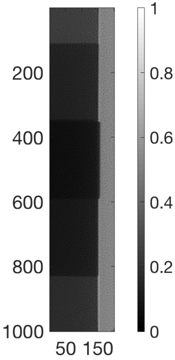
(b) Mean reconstruction

- Kernel size: 33×33
- $\lambda \approx 9000$, $\delta \approx 22$
- $m_d \times n_d = 512 \times 512$ pixels ($8 \times 8 = 64$ sub-images)
- 526 samples

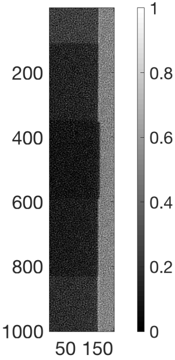
Deconvolution Results Step Wedge



(a) Blurred image

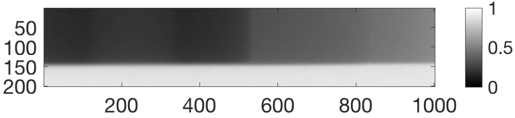


(b) Mean reconstruction

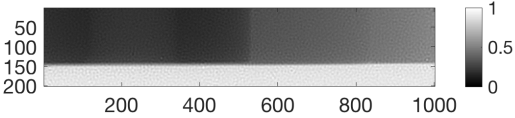


(c) Sample # 410

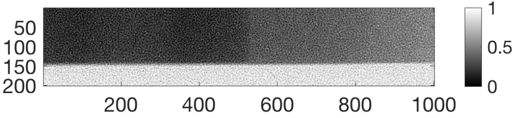
Deconvolution Results Abel Cylinder



(a) Blurred image



(b) Mean reconstruction



(c) Sample # 410

Conclusion

- No published MCMC algorithms in the literature using images and kernels of this size, even with traditional BCs
- Presented a block Gibbs sampler which:
  - is efficient w.r.t. IACT
  - is scalable to images of size 4096×4096
- Can generate $O(100)$ deblurred images per day on a reasonable desktop (or with a factor of 2 time increase with a 2017 Macbook Pro: 2.3 GHz Intel Core i5, 8 GB 2133 MHz DDR3)

Thank you for coming!

Questions?

In collaboration with Dr. Matthias Morzfeld, Dr. Aaron Luttman, and Dr. Kevin Joyce.

References

[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.

[3] 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.

[4] 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.

[5] 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.

[6] P. C. Hansen, J. Nagey, and D. P. O'Leary.
Deblurring Images: Matrices, Spectra, and Filtering.
SIAM Philadelphia, 2006.

[7] 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.

[8] D. Kincaid and W. Cheney.
Numerical Analysis: Mathematics of Scientific Computing.
Brooks/Cole, 3 edition, 2002.

[9] 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.

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

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

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

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

[14] D. S. Watkins.
Fundamentals of Matrix Computations.
John Wiley and Sons, 3 edition, 2010.

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

Conjugate Gradient

- Iterative method for solving $\mathbf{Rx} = \mathbf{d}$ where $\mathbf{R}$ is SPD
- Equivalent to minimizing the quadratic $Q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Rx} - \mathbf{x}^\top \mathbf{d}$, which has gradient $(\mathbf{Rx} - \mathbf{d})$
- In our case, $\mathbf{R} = \mathbf{H}_{ii}$
- The method:
  - Start with an initial guess (commonly $\mathbf{x}_0 = \mathbf{0}$), and build a set of basis vectors $\mathbf{v}_i$ which are conjugate, i.e. $\mathbf{v}_i^\top \mathbf{Rv}_j = 0$ for $i \neq j$
  - Similar to gradient descent, which chooses the search direction $\mathbf{r}_j = \mathbf{d} - \mathbf{Rx}_j$ (negative gradient)
  - Conjugacy imposed similar to Gram-Schmidt:

$$\mathbf{v}_j = \mathbf{r}_j - \sum_{i<j} \frac{\mathbf{v}_i^\top \mathbf{Rr}_j}{\mathbf{v}_i^\top \mathbf{Rv}_i} \mathbf{v}_i$$

  - Update step: $\mathbf{x}_{j+1} = \mathbf{x}_j + t_j \mathbf{v}_j, \qquad t_j = \frac{\mathbf{v}_j^\top \mathbf{r}_j}{\mathbf{v}_j^\top \mathbf{Rv}_j}$

This algorithm will produce an exact solution (assuming no numerical error) in n iterations, where $\mathbf{x} \in \mathbb{R}^n$ [8, 14].