Basics of Image Deblurring

Math 561

Fall, 2006

Outline

Image Restoration: Simple Example

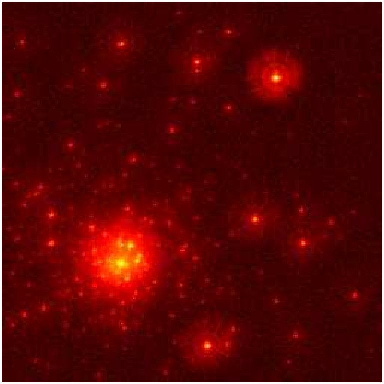
- Given blurred image, and some information about the blurring.
- Goal: Compute approximation of true image.

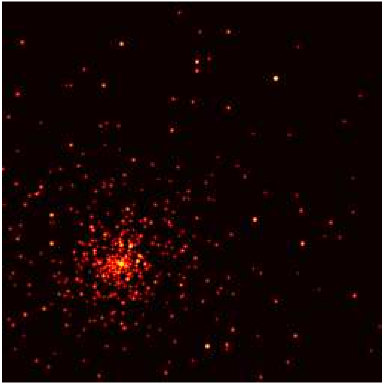
Image Restoration: Simple Example

- Given blurred image, and some information about the blurring.
- Goal: Compute approximation of true image.

Applications: Astronomy

Viewing distant star fields using ground based telescopes.

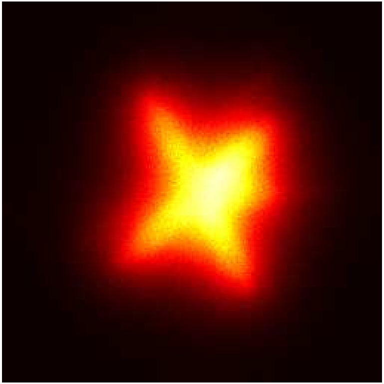




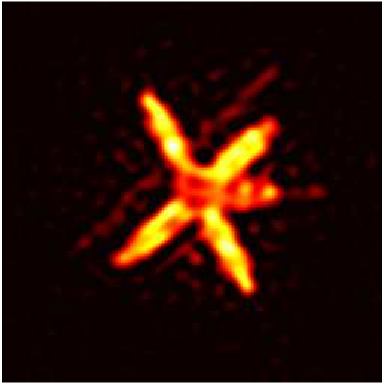
Applications: Space Observations

Viewing space vehicles, satellites and other space junk.

Observed data

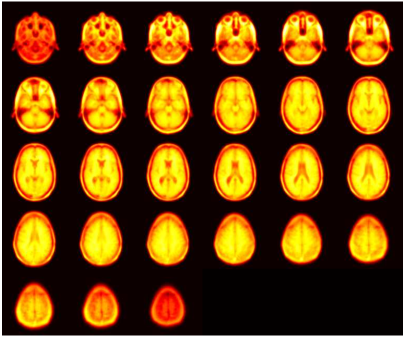


Restored data

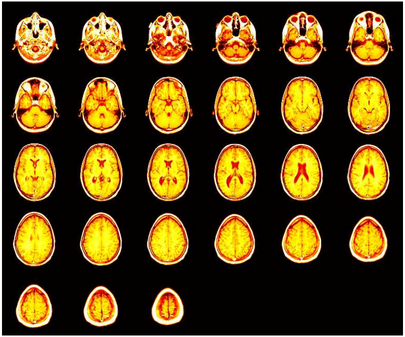


Applications: Medical Imaging

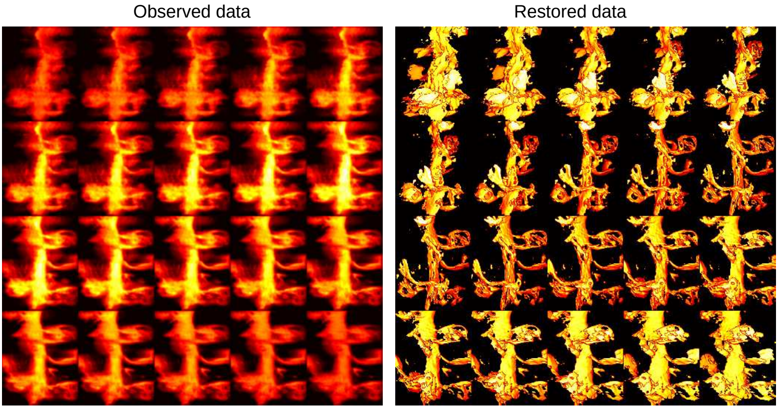
Observed data



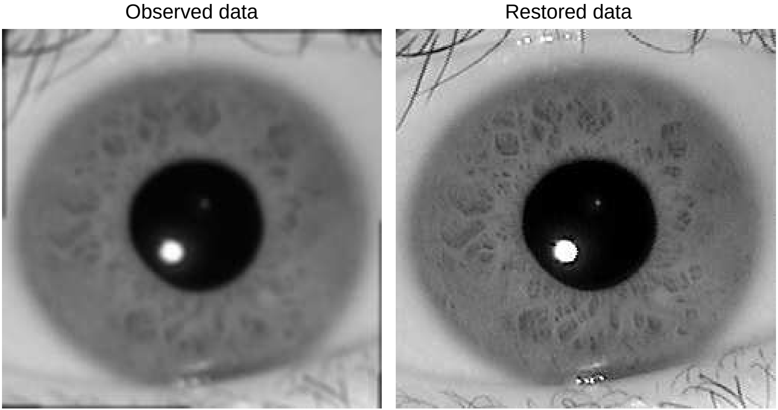
Restored data



Applications: Microscopy



Applications: Iris Recognition



Mathematical Model of Image Formation

$$b(u, v) = \int\int a(u, s, v, t)x(s, t)ds\, dt + e(u, v)$$

"Convolution" implies shift invariance

$$b(u,v) = \int\int a(u-s, v-t)x(s,t)ds\,dt + e(u,v)$$

Some remarks

- The mathematical model:

$$b(u,v) = \int\int a(u,s,v,t)x(s,t)ds\,dt + e(u,v)$$

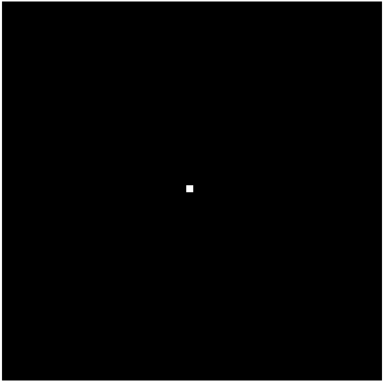
is an example of an ill-posed inverse problem.

Small changes in $e \Rightarrow$ large changes in x.

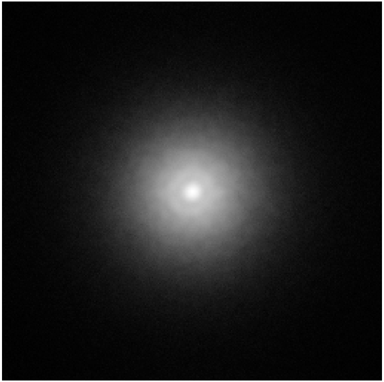
- Images are usually discrete pixel values, not functions!
  - Can approximate by matrix-vector equation: $\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}$
  - $\mathbf{A}$ is defined by the "point spread function" $a(u,s,v,t)$.
  - If the PSF is not known, it can be estimated by imaging "point source" objects.

Generating Experimental Point Spread Function

Point Source Object



PSF: Picture of Point Source Object



The Computational Problem

From the matrix-vector equation

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}$$

- Given $\mathbf{b}$ and $\mathbf{A}$ (or the PSF), compute an approximation of $\mathbf{x}$
- Regarding the noise, $\mathbf{e}$:
  - It is usually not known.
  - However, some statistical information may be known.
  - It is usually small, but it cannot be ignored!
    That is, solving the linear algebra problem:

    $$\mathbf{A}\mathbf{x} = \mathbf{b} \quad \Rightarrow \quad \mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

    usually does not work.

SVD Analysis

An important linear algebra tool: Singular Value Decomposition

Let $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ where

- $\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)\,, \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$
- $\mathbf{U}^T\mathbf{U} = \mathbf{I}\,, \quad \mathbf{V}^T\mathbf{V} = \mathbf{I}$

SVD Analysis

An important linear algebra tool: Singular Value Decomposition

Let $\quad \mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad$ where

- $\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)\,, \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$
- $\mathbf{U}^T\mathbf{U} = \mathbf{I}\,, \quad \mathbf{V}^T\mathbf{V} = \mathbf{I}$

For image restoration problems,

- $\sigma_1 \approx 1, \quad$ small singular values cluster at 0
- small singular values $\Rightarrow$ oscillating singular vectors

SVD Analysis

The naïve inverse solution can then be represented as:

$$\begin{aligned} \mathbf{x} &= \mathbf{A}^{-1}\mathbf{b} \\ &= \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T\mathbf{b} \\ &= \sum_{i=1}^{n} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i \end{aligned}$$

SVD Analysis

The naïve inverse solution can then be represented as:

$$\begin{aligned}\hat{\mathbf{x}} &= \mathbf{A}^{-1}(\mathbf{b}+\mathbf{e}) \\ &= \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T(\mathbf{b}+\mathbf{e}) \\ &= \sum_{i=1}^{n} \frac{\mathbf{u}_i^T(\mathbf{b}+\mathbf{e})}{\sigma_i}\mathbf{v}_i\end{aligned}$$

SVD Analysis

The naïve inverse solution can then be represented as:

$$
\begin{aligned}
\hat{\mathbf{x}} &= \mathbf{A}^{-1}(\mathbf{b} + \mathbf{e}) \\
&= \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T(\mathbf{b} + \mathbf{e}) \\
&= \sum_{i=1}^{n} \frac{\mathbf{u}_i^T(\mathbf{b} + \mathbf{e})}{\sigma_i}\mathbf{v}_i \\
&= \sum_{i=1}^{n} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i + \sum_{i=1}^{n} \frac{\mathbf{u}_i^T\mathbf{e}}{\sigma_i}\mathbf{v}_i \\
&= \mathbf{x} + \text{error}
\end{aligned}
$$

The Computational Problem

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}$$

- Given **b** and **A**
- Goal: Compute approximation of true image, **x**
- Naïve inverse solution

$$\hat{\mathbf{x}} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i + \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{e}}{\sigma_i} \mathbf{v}_i$$
$$= \mathbf{x} + \text{error}$$

is corrupted with noise!

The Computational Problem

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}$$

- Given **b** and **A**
- Goal: Compute approximation of true image, **x**
- Naïve inverse solution

$$\hat{\mathbf{x}} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i + \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{e}}{\sigma_i} \mathbf{v}_i = \mathbf{x} + \text{error}$$

is corrupted with noise!

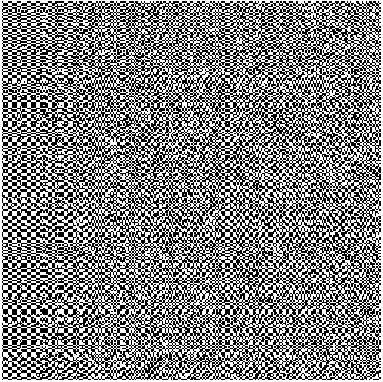
The Computational Problem

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}$$

- Given $\mathbf{b}$ and $\mathbf{A}$
- Goal: Compute approximation of true image, $\mathbf{x}$
- Naïve inverse solution

$$\begin{aligned}\hat{\mathbf{x}} &= \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}\mathbf{v}_i + \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{e}}{\sigma_i}\mathbf{v}_i \\ &= \mathbf{x} + \text{error}\end{aligned}$$

is corrupted with noise!



Basic Idea of Filtering

Basic Idea: Filter out effects of small singular values.

$$\mathbf{x}_{\text{reg}} = \sum_{i=1}^{n} \phi_i \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i$$

where the "filter factors" satisfy

$$\phi_i \approx \begin{cases} 1 & \text{if } \sigma_i \text{ is large} \\ 0 & \text{if } \sigma_i \text{ is small} \end{cases}$$

Examples of Filtering Methods

1. Truncated SVD

$$\mathbf{x}_{\text{tsvd}} = \sum_{i=1}^{k} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i$$

2. Tikhonov

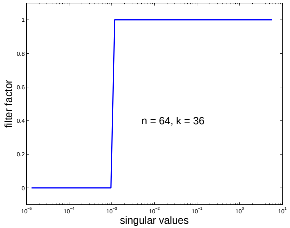
$$\mathbf{x}_{\text{tik}} = \sum_{i=1}^{n} \frac{\sigma_i^2}{\sigma_i^2 + \alpha^2} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i$$

3. Iterative (more in next lecture)

Examples of Filtering Methods

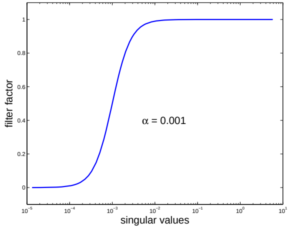
1. Truncated SVD

$$\mathbf{x}_{\text{tsvd}} = \sum_{i=1}^{k} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i$$



2. Tikhonov

$$\mathbf{x}_{\text{tik}} = \sum_{i=1}^{n} \frac{\sigma_i^2}{\sigma_i^2 + \alpha^2} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i$$



3. Iterative (more in next lecture)

Choosing Regularization Parameters

Lots of choices: Generalized Cross Validation (GCV), L-curve, discrepancy principle, ...

Choosing Regularization Parameters

Lots of choices: Generalized Cross Validation (GCV), L-curve, discrepancy principle, ...

GCV and Tikhonov: Choose λ to minimize

$$\mathrm{GCV}(\lambda) = \frac{n \displaystyle\sum_{i=1}^{n} \left(\frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i^2 + \lambda^2} \right)^2}{\left(\displaystyle\sum_{i=1}^{n} \frac{1}{\sigma_i^2 + \lambda^2} \right)^2}$$

Choosing Regularization Parameters

Lots of choices: Generalized Cross Validation (GCV), L-curve, discrepancy principle, ...

GCV and Tikhonov: Choose λ to minimize

$$\mathrm{GCV}(\lambda) = \frac{n \displaystyle\sum_{i=1}^{n} \left(\frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i^2 + \lambda^2} \right)^2}{\left(\displaystyle\sum_{i=1}^{n} \frac{1}{\sigma_i^2 + \lambda^2} \right)^2}$$

For L-curve, see G. Rodriguez.

Remarks on Computational Methods

- SVD filtering can be computationally expensive.
- Further simplifying approximations are often used to obtain more efficient algorithms:
  - Spatial invariance and periodic boundary conditions:
    - $\mathbf{A}$ is circulant.
    - Can replace SVD with fast Fourier transforms (FFT).
  - Other fast transforms (e.g., DCT) can sometimes be used.
  - Separable blur $\Rightarrow$ $\mathbf{A}$ can be decomposed using Kronecker products:

$$\mathbf{A} = \mathbf{A}_{\mathrm{r}} \otimes \mathbf{A}_{\mathrm{c}}$$

One-Dimensional Problems

Recall:

Each blurred pixel is a weighted sum of the corresponding pixel and its neighbors in the true image.

For example, if

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

then

$$b_3 = \square\, x_1 + \square\, x_2 + \square\, x_3 + \square\, x_4 + \square\, x_5$$

One-Dimensional Problems

The weights come from the PSF:

For example, if

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

then

One-Dimensional Problems

The weights come from the PSF:

For example, if

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}, \qquad \begin{bmatrix} p_5 \\ p_4 \\ p_3 \\ p_2 \\ p_1 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

then

1. Rotate the PSF, **p**, by 180 degrees about center.

One-Dimensional Problems

The weights come from the PSF:

For example, if

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \begin{bmatrix} p_5 \\ p_4 \\ p_3 \\ p_2 \\ p_1 \end{bmatrix} \qquad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

then

2. Match coefficients of rotated PSF and $\mathbf{x}$

One-Dimensional Problems

The weights come from the PSF:

For example, if

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \begin{bmatrix} p_5 \\ p_4 \\ p_3 \\ p_2 \\ p_1 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

then

3. Multiply corresponding components and sum.

One-Dimensional Problems

The weights come from the PSF:

For example, if

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \begin{bmatrix} p_5 \\ p_4 \\ p_3 \\ p_2 \\ p_1 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

then

$$b_3 = p_5 x_1 + p_4 x_2 + p_3 x_3 + p_2 x_4 + p_1 x_5$$

One-Dimensional Problems

If the weights fall outside the true image scene

$$\begin{matrix} ? \\ \begin{bmatrix} x_1 \\ \color{red}{x_2} \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \end{matrix} \begin{bmatrix} p_5 \\ p_4 \\ \color{red}{p_3} \\ p_2 \\ p_1 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ \color{red}{b_2} \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

then

$$b_2 = p_5 \underline{\ ?\ } + p_4 x_1 + p_3 x_2 + p_2 x_3 + p_1 x_4$$

One-Dimensional Problems

If the weights fall outside the true image scene
impose boundary conditions

$$\begin{matrix} w \\ \left[\begin{matrix} x_1 \\ \color{red}{x_2} \\ x_3 \\ x_4 \\ x_5 \end{matrix}\right] \end{matrix} \begin{matrix} \left[\begin{matrix} p_5 \\ p_4 \\ \color{red}{p_3} \\ p_2 \\ p_1 \end{matrix}\right] \\ \end{matrix} \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ \color{red}{b_2} \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

then

$$b_2 = p_5 \underline{\ w\ } + p_4 x_1 + p_3 x_2 + p_2 x_3 + p_1 x_4$$

One-Dimensional Problems

If the weights fall outside the true image scene impose boundary conditions, such as zero

$$\begin{matrix} 0 \\ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \end{matrix} \begin{bmatrix} p_5 \\ p_4 \\ p_3 \\ p_2 \\ p_1 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

then

$$b_2 = p_4 x_1 + p_3 x_2 + p_2 x_3 + p_1 x_4$$

One-Dimensional Problems

If the weights fall outside the true image scene impose boundary conditions, such as periodic

$$\begin{array}{c} x_5 \\ \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array}\right] \end{array} \left[\begin{array}{c} p_5 \\ p_4 \\ p_3 \\ p_2 \\ p_1 \end{array}\right] \qquad \mathbf{b} = \left[\begin{array}{c} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{array}\right]$$

then

$$b_2 = p_5 x_5 + p_4 x_1 + p_3 x_2 + p_2 x_3 + p_1 x_4$$

One-Dimensional Problems

If the weights fall outside the true image scene impose boundary conditions, such as reflexive

$$\begin{matrix} x_1 \\ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \end{matrix} \begin{bmatrix} p_5 \\ p_4 \\ p_3 \\ p_2 \\ p_1 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

then

$$b_2 = p_5 x_1 + p_4 x_1 + p_3 x_2 + p_2 x_3 + p_1 x_4$$

One-Dimensional Problems

In general, we can write

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} p_5 & p_4 & p_3 & p_2 & p_1 & & & & \\ & p_5 & p_4 & p_3 & p_2 & p_1 & & & \\ & & p_5 & p_4 & p_3 & p_2 & p_1 & & \\ & & & p_5 & p_4 & p_3 & p_2 & p_1 & \\ & & & & p_5 & p_4 & p_3 & p_2 & p_1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \hline x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \hline y_1 \\ y_2 \end{bmatrix}$$

where

- zero BC $\Rightarrow w_i = y_i = 0$
- periodic BC $\Rightarrow w_1 = x_4,\ w_2 = x_5,\ y_1 = x_1,\ y_2 = x_2$
- reflexive BC $\Rightarrow w_1 = x_2,\ w_2 = x_1,\ y_1 = x_5,\ y_2 = x_4$

One-Dimensional Problems

Therefore, for zero boundary conditions we get:

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} p_3 & p_2 & p_1 & & \\ p_4 & p_3 & p_2 & p_1 & \\ p_5 & p_4 & p_3 & p_2 & p_1 \\ & p_5 & p_4 & p_3 & p_2 \\ & & p_5 & p_4 & p_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

Here $\mathbf{A}$ is a Toeplitz matrix

One-Dimensional Problems

For periodic boundary conditions we get:

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} p_3 & p_2 & p_1 & p_5 & p_4 \\ p_4 & p_3 & p_2 & p_1 & p_5 \\ p_5 & p_4 & p_3 & p_2 & p_1 \\ p_1 & p_5 & p_4 & p_3 & p_2 \\ p_2 & p_1 & p_5 & p_4 & p_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

Here **A** is a circulant matrix

One-Dimensional Problems

For reflexive boundary conditions we get:

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \left(\begin{bmatrix} p_3 & p_2 & p_1 & & \\ p_4 & p_3 & p_2 & p_1 & \\ p_5 & p_4 & p_3 & p_2 & p_1 \\ & p_5 & p_4 & p_3 & p_2 \\ & & p_5 & p_4 & p_3 \end{bmatrix} + \begin{bmatrix} p_4 & p_5 & & & \\ p_5 & & & & \\ & & & & \\ & & & & p_1 \\ & & & p_1 & p_2 \end{bmatrix} \right) \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

Here **A** is a Toeplitz-plus-Hankel

Two-Dimensional Problems

With zero boundary conditions we obtain BTTB matrix:

$$
\left[\begin{array}{c} b_{11} \\ b_{21} \\ b_{31} \\ \hline b_{12} \\ b_{22} \\ b_{32} \\ \hline b_{13} \\ b_{23} \\ b_{33} \end{array}\right]
=
\left[\begin{array}{ccc|ccc|ccc}
p_{22} & p_{12} & & p_{21} & p_{11} & & & & \\
p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & & & \\
 & p_{32} & p_{22} & & p_{31} & p_{21} & & & \\ \hline
p_{23} & p_{13} & & p_{22} & p_{12} & & p_{21} & p_{11} & \\
p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} \\
 & p_{33} & p_{23} & & p_{32} & p_{22} & & p_{31} & p_{21} \\ \hline
 & & & p_{23} & p_{13} & & p_{22} & p_{12} & \\
 & & & p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} \\
 & & & & p_{33} & p_{23} & & p_{32} & p_{22}
\end{array}\right]
\left[\begin{array}{c} x_{11} \\ x_{21} \\ x_{31} \\ \hline x_{12} \\ x_{22} \\ x_{32} \\ \hline x_{13} \\ x_{23} \\ x_{33} \end{array}\right]
$$

$\mathbf{b} = \mathrm{vec}(\mathbf{B})$, $\mathbf{p} = \mathrm{vec}(\mathbf{P})$, $\mathbf{x} = \mathrm{vec}(\mathbf{X})$

Two-Dimensional Problems

Matrix structures:

- Zero boundary conditions $\Rightarrow$ **A** is BTTB
- Periodic boundary conditions $\Rightarrow$ **A** is BCCB
- Reflexive boundary conditions $\Rightarrow$ **A** is sum of BTTB, BTHB, BHTB, BHHB

Legend:

BTTB: Block Toeplitz with Toeplitz blocks

BCCB: Block circulant with circulant blocks

BHHB: Block Hankel with Hankel blocks

BTHB: Block Toeplitz with Hankel blocks

BHTB: Block Hankel with Toeplitz blocks

Remark on Boundary Conditions

- Many other choices for boundary conditions.
- For example: Anti-reflective
  (Aricó, Donatelli, Serra-Cappizano)
  - Preserve continuity of the image at boundaries.
  - Preserve continuity of the normal derivative at the boundary.
  - Can help to reduce ringing artifacts.

Separable Two-Dimensional Blurs

Separable Blur ⇒ Horizontal and vertical components separate.

In this case, the PSF array has rank $= 1$:

$$\mathbf{P} = \mathbf{r}\mathbf{c}^T = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix} = \begin{bmatrix} c_1 r_1 & c_1 r_2 & c_1 r_3 \\ c_2 r_1 & c_2 r_2 & c_2 r_3 \\ c_3 r_1 & c_3 r_2 & c_3 r_3 \end{bmatrix}$$

Separable Two-Dimensional Blurs

Separable Blur ⇒ Horizontal and vertical components separate.

Forming the matrix with this special PSF we obtain (zero BC):

$$\mathbf{A} = \left[\begin{array}{ccc|ccc|ccc}
c_2 r_2 & c_1 r_2 & & c_2 r_1 & c_1 r_1 & & & & \\
c_3 r_2 & c_2 r_2 & c_1 r_2 & c_3 r_1 & c_2 r_1 & c_1 r_1 & & & \\
 & c_3 r_2 & c_2 r_2 & & c_3 r_1 & c_2 r_1 & & & \\
\hline
c_2 r_3 & c_1 r_3 & & c_2 r_2 & c_1 r_2 & & c_2 r_1 & c_1 r_1 & \\
c_3 r_3 & c_2 r_3 & c_1 r_3 & c_3 r_2 & c_2 r_2 & c_1 r_2 & c_3 r_1 & c_2 r_1 & c_1 r_1 \\
 & c_3 r_3 & c_2 r_3 & & c_3 r_2 & c_2 r_2 & & c_3 r_1 & c_2 r_1 \\
\hline
 & & & c_2 r_3 & c_1 r_3 & & c_2 r_2 & c_1 r_2 & \\
 & & & c_3 r_3 & c_2 r_3 & c_1 r_3 & c_3 r_2 & c_2 r_2 & c_1 r_2 \\
 & & & & c_3 r_3 & c_2 r_3 & & c_3 r_2 & c_2 r_2
\end{array}\right]$$

Separable Two-Dimensional Blurs

Separable Blur ⇒ Horizontal and vertical components separate.

Forming the matrix with this special PSF we obtain (zero BC):

$$\mathbf{A} = \left[\begin{array}{c|c|c} r_2 \begin{bmatrix} c_2 & c_1 & \\ c_3 & c_2 & c_1 \\ & c_3 & c_2 \end{bmatrix} & r_1 \begin{bmatrix} c_2 & c_1 & \\ c_3 & c_2 & c_1 \\ & c_3 & c_2 \end{bmatrix} & 0 \\ \hline r_3 \begin{bmatrix} c_2 & c_1 & \\ c_3 & c_2 & c_1 \\ & c_3 & c_2 \end{bmatrix} & r_2 \begin{bmatrix} c_2 & c_1 & \\ c_3 & c_2 & c_1 \\ & c_3 & c_2 \end{bmatrix} & r_1 \begin{bmatrix} c_2 & c_1 & \\ c_3 & c_2 & c_1 \\ & c_3 & c_2 \end{bmatrix} \\ \hline 0 & r_3 \begin{bmatrix} c_2 & c_1 & \\ c_3 & c_2 & c_1 \\ & c_3 & c_2 \end{bmatrix} & r_2 \begin{bmatrix} c_2 & c_1 & \\ c_3 & c_2 & c_1 \\ & c_3 & c_2 \end{bmatrix} \end{array}\right]$$

Separable Two-Dimensional Blurs

Separable Blur ⇒ Horizontal and vertical components separate.

Forming the matrix with this special PSF we obtain (zero BC):

$$\mathbf{A} = \mathbf{A}_{\mathrm{r}} \otimes \mathbf{A}_{\mathrm{c}} \begin{bmatrix} r_2 & r_1 & \\ r_3 & r_2 & r_1 \\ & r_3 & r_2 \end{bmatrix} \otimes \begin{bmatrix} c_2 & c_1 & \\ c_3 & c_2 & c_1 \\ & c_3 & c_2 \end{bmatrix}$$

Where $\otimes$ denotes Kronecker product.

Separable Two-Dimensional Blurs

Similar structures occur for other boundary conditions:

$$\mathbf{A} = \mathbf{A}_{\mathrm{r}} \otimes \mathbf{A}_{\mathrm{c}}$$

where

- Zero boundary conditions:
  - $\mathbf{A}_{\mathrm{r}}$ is Toeplitz, defined by $\mathbf{r}$
  - $\mathbf{A}_{\mathrm{c}}$ is Topelitz, defined by $\mathbf{c}$
- Periodic boundary conditions:
  - $\mathbf{A}_{\mathrm{r}}$ is circulant, defined by $\mathbf{r}$
  - $\mathbf{A}_{\mathrm{c}}$ is circulant, defined by $\mathbf{c}$
- Reflexive boundary conditions:
  - $\mathbf{A}_{\mathrm{r}}$ is Toeplitz-plus-Hankel, defined by $\mathbf{r}$
  - $\mathbf{A}_{\mathrm{c}}$ is Toeplitz-plus-Hankel, defined by $\mathbf{c}$

Summary of Matrix Structures

BC	Non-separable PSF	Separable PSF
zero	BTTB	Kronecker of Toeplitz matrices
periodic	BCCB	Kronecker of circulant matrices
reflexive	BTTB+BTHB +BHTB+BHHB	Kronecker of Toeplitz-plus-Hankel matrices

BCCB Matrices

With periodic boundary conditions **A** is a BCCB matrix:

$$\left[\begin{array}{c} b_{11} \\ b_{21} \\ b_{31} \\ \hline b_{12} \\ b_{22} \\ b_{32} \\ \hline b_{13} \\ b_{23} \\ b_{33} \end{array}\right] = \left[\begin{array}{ccc|ccc|ccc} p_{22} & p_{12} & p_{32} & p_{21} & p_{11} & p_{31} & p_{23} & p_{13} & p_{33} \\ p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & p_{33} & p_{23} & p_{13} \\ p_{12} & p_{32} & p_{22} & p_{11} & p_{31} & p_{21} & p_{13} & p_{33} & p_{23} \\ \hline p_{23} & p_{13} & p_{33} & p_{22} & p_{12} & p_{32} & p_{21} & p_{11} & p_{31} \\ p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} \\ p_{13} & p_{33} & p_{23} & p_{12} & p_{32} & p_{22} & p_{11} & p_{31} & p_{21} \\ \hline p_{21} & p_{11} & p_{31} & p_{23} & p_{13} & p_{33} & p_{22} & p_{12} & p_{32} \\ p_{31} & p_{21} & p_{11} & p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} \\ p_{11} & p_{31} & p_{21} & p_{13} & p_{33} & p_{23} & p_{12} & p_{32} & p_{22} \end{array}\right] \left[\begin{array}{c} x_{11} \\ x_{21} \\ x_{31} \\ \hline x_{12} \\ x_{22} \\ x_{32} \\ \hline x_{13} \\ x_{23} \\ x_{33} \end{array}\right]$$

$\mathbf{b} = \mathrm{vec}(\mathbf{B})$, $\mathbf{p} = \mathrm{vec}(\mathbf{P})$, $\mathbf{x} = \mathrm{vec}(\mathbf{X})$

Important BCCB Matrix Property

- Every BCCB matrix has the same set of eigenvectors:

$$\mathbf{A} = \mathbf{F}^* \boldsymbol{\Lambda} \mathbf{F}$$

where

  - $\mathbf{F}$ is the two-dimensional discrete Fourier transform matrix
  - $\mathbf{F}^*\mathbf{F} = \mathbf{F}\mathbf{F}^* = \mathbf{I}$
  - $\boldsymbol{\Lambda}$ = diagonal containing eigenvalues of $\mathbf{A}$

- Computations with $\mathbf{F}$ can be done very efficiently:

$$O(N \log N)$$

using Fast Fourier Transforms (FFT)s.

BCCB and FFT Relations

$$\mathbf{A} = \mathbf{F}^* \mathbf{\Lambda} \mathbf{F} \quad \Rightarrow \quad \mathbf{F}\mathbf{A} = \mathbf{\Lambda}\mathbf{F} \quad \Rightarrow \quad \mathbf{F}\mathbf{a}_1 = \mathbf{\Lambda}\mathbf{f}_1$$

where

- $\mathbf{a}_1$ = first column of $\mathbf{A}$
- $\mathbf{f}_1$ = first column of $\mathbf{F}$,

$$\mathbf{f}_1 = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

- Thus,

$$\mathbf{F}\mathbf{a}_1 = \mathbf{\Lambda}\mathbf{f}_1 = \frac{1}{\sqrt{N}} \boldsymbol{\lambda}$$

where $\boldsymbol{\lambda}$ is a vector containing the eigenvalues of $\mathbf{A}$.

Some BCCB Computations

If $\mathbf{A}$ is BCCB defined by PSF $\mathbf{P}$, and

$$\mathbf{b} = \mathbf{A}\mathbf{x} = \mathbf{F}^* \mathbf{\Lambda} \mathbf{F} \mathbf{x}$$

then to compute $\mathbf{b}$ use

```
S = fft2( circshift(P) );
B = ifft2(S .* fft2(X));
```

where

$$\mathbf{b} = \mathrm{vec}(\mathbf{B}) \quad \text{and} \quad \mathbf{x} = \mathrm{vec}(\mathbf{X})$$

Some BCCB Computations

If $\mathbf{A}$ is BCCB defined by PSF $\mathbf{P}$, and

$$\mathbf{x}^{\text{naive}} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{F}^{*}\mathbf{\Lambda}^{-1}\mathbf{F}\mathbf{b}$$

then to compute $\mathbf{x}^{\text{naive}}$ use

```
S = fft2( circshift(P) );
X = ifft2(fft2(B) ./ S);
```

where

$$\mathbf{b} = \text{vec}(\mathbf{B}) \quad \text{and} \quad \mathbf{x} = \text{vec}(\mathbf{X})$$

Some BCCB Computations

If $\mathbf{A}$ is BCCB defined by PSF $\mathbf{P}$, and $\mathbf{\Phi}$ contains filter factors,

$$\mathbf{x}^{\text{filt}} = \mathbf{F}^* \mathbf{\Phi} \mathbf{\Lambda}^{-1} \mathbf{F} \mathbf{b}$$

then to compute $\mathbf{x}^{\text{filt}}$ use

```
S = fft2( circshift(P) );
Sfilt = Phi ./ S;
X = ifft2(fft2(B) .* Sfilt);
```

where

$$\mathbf{b} = \text{vec}(\mathbf{B}) \quad \text{and} \quad \mathbf{x} = \text{vec}(\mathbf{X})$$

Summary of Matrix Structures

BC	Non-separable PSF	Separable PSF
zero	BTTB	Kronecker of Toeplitz matrices
periodic	BCCB	Kronecker of circulant matrices
reflexive	BTTB+BTHB +BHTB+BHHB	Kronecker of Toeplitz-plus-Hankel matrices

Toeplitz-plus-Hankel Matrices

With reflexive boundary conditions **A** is a

$$\text{BTTB} + \text{BTHB} + \text{BHTB} + \text{BHHB}$$

matrix defined by the PSF.

"Strong" symmetry condition: If

$$\mathbf{P} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{P}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

where

- $\widetilde{\mathbf{P}}$ is $(2k-1) \times (2k-1)$ with center located at the (k, k)
- $\widetilde{\mathbf{P}} = \texttt{fliplr}(\widetilde{\mathbf{P}}) = \texttt{flipud}(\widetilde{\mathbf{P}}) = \texttt{fliplr}(\texttt{flipud}(\widetilde{\mathbf{P}}))$

Toeplitz-plus-Hankel Matrices

With reflexive boundary conditions $\mathbf{A}$ is a

$$\text{BTTB} + \text{BTHB} + \text{BHTB} + \text{BHHB}$$

matrix defined by the PSF.

"Strong" symmetry condition: If

$$\mathbf{P} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{P}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

where

- $\widetilde{\mathbf{P}}$ is $(2k-1) \times (2k-1)$ with center located at the (k,k)
- $\widetilde{\mathbf{P}} = \texttt{fliplr}(\widetilde{\mathbf{P}}) = \texttt{flipud}(\widetilde{\mathbf{P}}) = \texttt{fliplr(flipud}(\widetilde{\mathbf{P}}))$

BTTB+BTHB+BHTB+BHHB Matrix Properties

If the PSF satisfies strong symmetry condition, then:

- $\mathbf{A}$ is symmetric
- $\mathbf{A}$ is block symmetric
- Each block in $\mathbf{A}$ is symmetric
- $\mathbf{A}$ has the spectral decomposition

$$\mathbf{A} = \mathbf{C}^T \mathbf{\Lambda} \mathbf{C}$$

  where $\mathbf{C}$ is the two-dimensional discrete cosine transform (DCT) matrix.
- As with FFTs, computations with $\mathbf{C}$ cost $O(N \log N)$.

Toeplitz-plus-Hankel and DCT Relations

$$\mathbf{A} = \mathbf{C}^T \mathbf{\Lambda} \mathbf{C} \quad \Rightarrow \quad \mathbf{C}\mathbf{A} = \mathbf{\Lambda}\mathbf{C} \quad \Rightarrow \quad \mathbf{C}\mathbf{a}_1 = \mathbf{\Lambda}\mathbf{c}_1$$

where

- $\mathbf{a}_1$ = first column of $\mathbf{A}$
- $\mathbf{c}_1$ = first column of $\mathbf{C}$,
- Thus, the eigenvalues of $\mathbf{C}$ are given by

$$\mathbf{C}\mathbf{a}_1 = \mathbf{\Lambda}\mathbf{c}_1 \quad \Rightarrow \quad \lambda_i = \frac{[\mathbf{C}\mathbf{a}_1]_i}{[\mathbf{c}_1]_i}$$

Additional DCT Computations

If **A** is defined by strongly symmetric PSF with reflexive boundary conditions, and

$$\mathbf{b} = \mathbf{A}\mathbf{x} = \mathbf{C}^T \mathbf{\Lambda} \mathbf{C}\mathbf{x}$$

then to compute **b** use

```
e1 = zeros(size(P));, e1(1,1) = 1;
S = dct2( dctshift(P) ) ./ dct2(e1);
B = idct2(S .* dct2(X));
```

where

$$\mathbf{b} = \text{vec}(\mathbf{B}) \quad \text{and} \quad \mathbf{x} = \text{vec}(\mathbf{X})$$

Additional DCT Computations

If $\mathbf{A}$ is defined by strongly symmetric PSF with reflexive boundary conditions, and

$$\mathbf{x}^{\text{naive}} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{C}^T \mathbf{\Lambda}^{-1} \mathbf{C}\mathbf{b}$$

then to compute $\mathbf{x}^{\text{naive}}$ use

```
e1 = zeros(size(P));, e1(1,1) = 1;
S = dct2( dctshift(P) ) ./ dct2(e1);
X = idct2(dct2(B) ./ S);
```

where

$$\mathbf{b} = \text{vec}(\mathbf{B}) \quad \text{and} \quad \mathbf{x} = \text{vec}(\mathbf{X})$$

Additional DCT Computations

If $\mathbf{A}$ is defined by strongly symmetric PSF with reflexive boundary conditions, and $\mathbf{\Phi}$ contains filter factors,

$$\mathbf{x}^{\text{filt}} = \mathbf{C}^T \mathbf{\Phi} \mathbf{\Lambda}^{-1} \mathbf{C} \mathbf{b}$$

then to compute $\mathbf{x}^{\text{filt}}$ use

```
e1 = zeros(size(P));, e1(1,1) = 1;
S = dct2( dctshift(P) ) ./ dct2(e1);
Sfilt = Phi ./ S;
X = idct2(dct2(B) .* Sfilt);
```

where

$$\mathbf{b} = \text{vec}(\mathbf{B}) \quad \text{and} \quad \mathbf{x} = \text{vec}(\mathbf{X})$$

Summary of Matrix Structures

BC	Non-separable PSF	Separable PSF
zero	BTTB	Kronecker of Toeplitz matrices
periodic	BCCB	Kronecker of circulant matrices
reflexive	BTTB+BTHB +BHTB+BHHB	Kronecker of Toeplitz-plus-Hankel matrices
reflexive strongly symmetric	BTTB+BTHB +BHTB+BHHB	Kronecker of symmetric Toeplitz-plus-Hankel matrices

Separable PSFs

Recall: If the PSF has rank = 1,

$$\mathbf{P} = \mathbf{c}\mathbf{r}^T = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} \begin{bmatrix} r_1 & r_2 & \cdots & r_n \end{bmatrix}$$

then the blurring matrix has the form

$$\mathbf{A} = \mathbf{A}_{\mathrm{r}} \otimes \mathbf{A}_{\mathrm{c}}$$

where $\mathbf{A}_{\mathrm{r}}$ is defined by $\mathbf{r}$ and $\mathbf{A}_{\mathrm{c}}$ is defined by $\mathbf{c}$.

Assume for now $\mathbf{A}_{\mathrm{r}}$ and $\mathbf{A}_{\mathrm{c}}$ are known.

Useful Kronecker Product Properties

- $\mathbf{b} = (\mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c})\mathbf{x} \quad \Leftrightarrow \quad \mathbf{B} = \mathbf{A}_\mathrm{c}\mathbf{X}\mathbf{A}_\mathrm{r}^T$
  where $\mathbf{b} = \mathrm{vec}(\mathbf{B})$ and $\mathbf{x} = \mathrm{vec}(\mathbf{X})$
- $(\mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c})^T = \mathbf{A}_\mathrm{r}^T \otimes \mathbf{A}_\mathrm{c}^T$
- $(\mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c})^{-1} = \mathbf{A}_\mathrm{r}^{-1} \otimes \mathbf{A}_\mathrm{c}^{-1}$
- $(\mathbf{A}_\mathrm{r}^{(1)} \otimes \mathbf{A}_\mathrm{c}^{(1)})(\mathbf{A}_\mathrm{r}^{(2)} \otimes \mathbf{A}_\mathrm{c}^{(2)}) = (\mathbf{A}_\mathrm{r}^{(1)}\mathbf{A}_\mathrm{r}^{(2)}) \otimes (\mathbf{A}_\mathrm{c}^{(1)}\mathbf{A}_\mathrm{c}^{(2)})$

Exploiting Kronecker Product Properties in MATLAB

Using the property:

$$\mathbf{b} = (\mathbf{A}_{\mathrm{r}} \otimes \mathbf{A}_{\mathrm{c}})\mathbf{x} \quad \Leftrightarrow \quad \mathbf{B} = \mathbf{A}_{\mathrm{c}}\mathbf{X}\mathbf{A}_{\mathrm{r}}^{T}$$

in MATLAB we can compute

```
B = Ac*X*Ar';
```

Exploiting Kronecker Product Properties in MATLAB

Using the property:

$$\mathbf{b} = (\mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c})\mathbf{x} \quad \Leftrightarrow \quad \mathbf{B} = \mathbf{A}_\mathrm{c}\mathbf{X}\mathbf{A}_\mathrm{r}^T$$

and if $\mathbf{A}_\mathrm{r}$ and $\mathbf{A}_\mathrm{c}$ are nonsingular,

$$(\mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c})^{-1} = \mathbf{A}_\mathrm{r}^{-1} \otimes \mathbf{A}_\mathrm{c}^{-1}$$

we obtain

$$\mathbf{X} = \mathbf{A}_\mathrm{c}^{-1}\mathbf{B}\mathbf{A}_\mathrm{r}^{-T}$$

In MATLAB we can compute

```
X = Ac \ B / Ar';
```

Exploiting Kronecker Product Properties in MATLAB

We can compute SVD of small matrices:

$$\mathbf{A}_\mathrm{r} = \mathbf{U}_\mathrm{r}\boldsymbol{\Sigma}_\mathrm{r}\mathbf{V}_\mathrm{r}^T \quad \text{and} \quad \mathbf{A}_\mathrm{c} = \mathbf{U}_\mathrm{c}\boldsymbol{\Sigma}_\mathrm{c}\mathbf{V}_\mathrm{c}^T$$

Then

$$\begin{aligned} \mathbf{A} &= \mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c} \\ &= (\mathbf{U}_\mathrm{r}\boldsymbol{\Sigma}_\mathrm{r}\mathbf{V}_\mathrm{r}^T) \otimes (\mathbf{U}_\mathrm{c}\boldsymbol{\Sigma}_\mathrm{c}\mathbf{V}_\mathrm{c}^T) \\ &= (\mathbf{U}_\mathrm{r} \otimes \mathbf{U}_\mathrm{c})(\boldsymbol{\Sigma}_\mathrm{r} \otimes \boldsymbol{\Sigma}_\mathrm{c})(\mathbf{V}_\mathrm{r} \otimes \mathbf{V}_\mathrm{c})^T \\ &= \text{SVD of big matrix } \mathbf{A} \end{aligned}$$

Note: Do not need to explicitly form big matrices

$$\mathbf{U}_\mathrm{r} \otimes \mathbf{U}_\mathrm{c}, \quad \boldsymbol{\Sigma}_\mathrm{r} \otimes \boldsymbol{\Sigma}_\mathrm{c}, \quad \mathbf{V}_\mathrm{r} \otimes \mathbf{V}_\mathrm{c}$$

Exploiting Kronecker Product Properties in MATLAB

To compute inverse solution from SVD of small matrices:

$$\mathbf{x}^{\text{naive}} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{b}$$

is equivalent to

$$\mathbf{X}^{\text{naive}} = \mathbf{A}_{\text{c}}^{-1}\mathbf{B}\mathbf{A}_{\text{r}}^{-T} = \mathbf{V}_{\text{c}}\mathbf{\Sigma}_{\text{c}}^{-1}\mathbf{U}_{\text{c}}^T\mathbf{B}\mathbf{U}_{\text{r}}\mathbf{\Sigma}_{\text{r}}^{-1}\mathbf{V}_{\text{r}}^T$$

A MATLAB implementation could be:

```
[Ur, Sr, Vr] = svd(Ar);
[Uc, Sc, Vc] = svd(Ac);
S = diag(Sc) * diag(Sr)';
X = Vc * ( (Uc' * B * Ur)./S ) * Vr';
```

Exploiting Kronecker Product Properties in MATLAB

If $\mathbf{\Phi}$ contains filter factors, the filtered solution

$$\mathbf{x}^{\mathrm{filt}} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{V}\mathbf{\Phi}\mathbf{\Sigma}^{-1}\mathbf{U}^{T}\mathbf{b}$$

can be computed as:

```
[Ur, Sr, Vr] = svd(Ar);
[Uc, Sc, Vc] = svd(Ac);
S = diag(Sc) * diag(Sr)';
Sfilt = Phi ./ S;
X = Vc * ( (Uc' * B * Ur) .* Sfilt ) * Vr';
```

Summary of Fast Algorithms

For spatially invariant PSFs, we have the following fast algorithms.

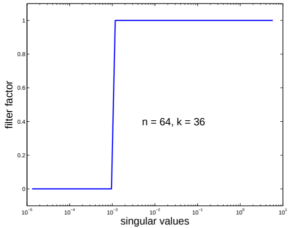
PSF	Boundary condition	Matrix structure	Fast algorithm
arbitrary	periodic	BCCB	2-d FFT
strongly symmetric	reflexive	sum of BXXB	2-d DCT
rank one	arbitrary	Kronecker product	2 SVDs

MATLAB Examples

Recall some examples of filtering methods:

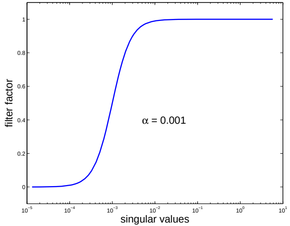
1. Truncated SVD

$$\mathbf{x}_{\text{tsvd}} = \sum_{i=1}^{k} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i$$



2. Tikhonov

$$\mathbf{x}_{\text{tik}} = \sum_{i=1}^{n} \frac{\sigma_i^2}{\sigma_i^2 + \alpha^2} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i$$



3. Iterative (more in next lecture)

The End

- Image deblurring examples arise in many applications.
- Fast algorithms can produce good reconstructions for many problems.
- Further details and software can be found in:

  Deblurring Images: Matrices, Spectra and Filtering
  P. C. Hansen, J. G. Nagy and D. P. O'Leary
  SIAM, 2006
  `http://www2.imm.dtu.dk/~pch/HNO/`
- Next time: Iterative methods for harder problems.