Structured Matrix Computations

1. Basic Structures for One-Dimensional Problems
   - the role of boundary conditions
2. BCCB Matrices
   - periodic boundary conditions
3. Symmetric Toeplitz-plus-Hankel Matrices
   - reflexive boundary conditions
4. Kronecker Product Matrices
   - when the variables separate in the PSF
5. Summary of Fast Algorithms
6. Creating Realistic Test Data

The Linear Deblurring Model

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

- Given:
  a blurred and noisy image
  $\mathbf{b} = \text{vec}(\mathbf{B})$
  and a BIG blurring matrix $\mathbf{A}$.

- Goal:
  Compute an *approximation*
  of the true image $\mathbf{x} = \text{vec}(\mathbf{X})$.

Useful Matrix Factorizations

Singular Value Decomposition (SVD)

$$\mathbf{A} = \mathbf{U}\,\boldsymbol{\Sigma}\,\mathbf{V}^T$$

where all matrices are *real*

- $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N)\,, \qquad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_N \geq 0$
- $\mathbf{U} = [\,\mathbf{u}_1\,,\,\mathbf{u}_2\,,\,\ldots\,,\,\mathbf{u}_N\,], \qquad \mathbf{V} = [\,\mathbf{v}_1\,,\,\mathbf{v}_2\,,\,\ldots\,,\,\mathbf{v}_N\,]$
- $\mathbf{U}^T\mathbf{U} = \mathbf{I}\,, \quad \mathbf{V}^T\mathbf{V} = \mathbf{I}$

Spectral Decomposition

$$\mathbf{A} = \widetilde{\mathbf{U}}\,\boldsymbol{\Lambda}\,\widetilde{\mathbf{U}}^*$$

where the matrices are usually *complex*

- $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$ – no ordering
- $\widetilde{\mathbf{U}} = [\,\tilde{\mathbf{u}}_1\,,\,\tilde{\mathbf{u}}_2\,,\,\ldots\,,\,\tilde{\mathbf{u}}_N\,]$
- $\widetilde{\mathbf{U}}^*\,\widetilde{\mathbf{U}} = \mathbf{I}$, where $\widetilde{\mathbf{U}}^* =$ complex conjugate of $\widetilde{\mathbf{U}}^T$

Chapter Goal

The SVD and Spectral Decomposition can be used to:

- Investigate sensitivity of image deblurring problem
  $\rightarrow$ Chapter 5.
- Construct image deblurring algorithms
  $\rightarrow$ Chapter 6.

To compute these decompositions efficiently for large matrices, we must *exploit structure* $\rightarrow$ this chapter.

Question: What is the matrix $\mathbf{A}$ and how do we get it?

Basic Structures: 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

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

then

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

The weights come from the PSF.

An example, $\mathbf{p}$ = PSF array, $\mathbf{b} = \mathbf{A}\,\mathbf{x}$ = sum of weighted PSF's:

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

$$\mathbf{b} = \begin{bmatrix} p_3 \\ p_4 \\ p_5 \\ 0 \\ 0 \end{bmatrix} x_1 + \begin{bmatrix} p_2 \\ p_3 \\ p_4 \\ p_5 \\ 0 \end{bmatrix} x_2 + \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix} x_3 + \begin{bmatrix} 0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} x_4 + \begin{bmatrix} 0 \\ 0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} x_5$$

1. "Rotate" the PSF $\mathbf{p}$ 180 degrees about center:

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

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

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

3. Multiply corresponding components and sum them:

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

to obtain

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

This is *one-dimensional convolution.*

Same idea when **x** is longer than **p**:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \color{red}{x_5} \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} \begin{bmatrix} p_5 \\ p_4 \\ \color{red}{p_3} \\ p_2 \\ p_1 \end{bmatrix} \qquad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \color{red}{b_5} \\ b_6 \\ b_6 \\ b_8 \\ b_9 \end{bmatrix}$$

where we obtain from the convolution

$$b_5 = p_5 x_3 + p_4 x_4 + p_3 x_5 + p_2 x_6 + p_1 x_7$$

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

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

Impose boundary conditions:

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

then

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

Impose boundary conditions, such as **zero**

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

then

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

Impose boundary conditions, such as **periodic**

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

then

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

Impose boundary conditions, such as **reflexive**

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

then

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

In General, We Can Write

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

- "empty element" denotes 0
- 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 $\Rightarrow w_1 = x_2,\ w_2 = x_1,\ y_1 = x_5,\ y_2 = x_4$

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*.

Note that

- the middle column is identical to $\mathbf{p}$, and
- the middle row $[\, p_5 \;\; p_4 \;\; p_3 \;\; p_2 \;\; p_1 \,]$ consists of the elements of $\mathbf{p}$ in reverse order.

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*.

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 $\mathbf{A}$ is a *Toeplitz-plus-Hankel matrix*.

Two-Dimensional Problems

As with one-dimensional problems, to compute pixel b_{ij}:

1. Rotate the PSF $\mathbf{P}$ by 180 degrees.
2. Locate it at the desired position.
3. Match coefficients of rotated PSF and $\mathbf{X}$.
4. Multiply corresponding components and sum them.

For example, to compute b_{22}

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & \color{red}{x_{22}} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \quad \begin{bmatrix} \color{blue}{p_{11}} & \color{blue}{p_{12}} & \color{blue}{p_{13}} \\ \color{blue}{p_{21}} & \color{blue}{p_{22}} & \color{blue}{p_{23}} \\ \color{blue}{p_{31}} & \color{blue}{p_{32}} & \color{blue}{p_{33}} \end{bmatrix} \quad \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & \color{red}{b_{22}} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

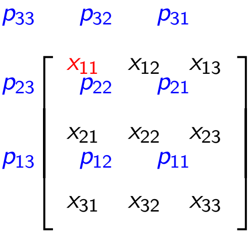
$$\mathbf{X} \qquad\qquad \mathbf{P} \qquad\qquad \mathbf{B}$$

Rotate, multiply, and sum:

$$\begin{bmatrix} x_{11} \; {\color{blue}p_{33}} & x_{12} \; {\color{blue}p_{32}} & x_{13} \; {\color{blue}p_{31}} \\ x_{21} \; {\color{blue}p_{23}} & {\color{red}x_{22}} \; {\color{blue}p_{22}} & x_{23} \; {\color{blue}p_{21}} \\ x_{31} \; {\color{blue}p_{13}} & x_{32} \; {\color{blue}p_{12}} & x_{33} \; {\color{blue}p_{11}} \end{bmatrix} \qquad \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & {\color{red}b_{22}} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$\begin{aligned} b_{22} &= p_{33}x_{11} + p_{32}x_{12} + p_{31}x_{13} + \\ &\quad p_{23}x_{21} + p_{22}x_{22} + p_{21}x_{23} + \\ &\quad p_{13}x_{31} + p_{12}x_{32} + p_{11}x_{33} \end{aligned}$$

Consider what happens at the edges:



$$\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$\begin{aligned} b_{11} \quad = \quad & p_{33}\underline{\ ?\ } + p_{32}\underline{\ ?\ } + p_{31}\underline{\ ?\ } + \\ & p_{23}\underline{\ ?\ } + p_{22}x_{11} + p_{21}x_{12} + \\ & p_{13}\underline{\ ?\ } + p_{12}x_{21} + p_{11}x_{22} \end{aligned}$$

Again, we need to impose boundary conditions:

$$
\begin{array}{cccc}
p_{33} & p_{32} & p_{31} & \\
p_{23} & x_{11} \;\; p_{22} & x_{12} \;\; p_{21} & x_{13} \\
p_{13} & x_{21} \;\; p_{12} & x_{22} \;\; p_{11} & x_{23} \\
 & x_{31} & x_{32} & x_{33}
\end{array}
\qquad\qquad
\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}
$$

Zero:

$$
\begin{aligned}
b_{11} \quad = \quad & p_{33}\underline{\;0\;} + p_{32}\underline{\;0\;} + p_{31}\underline{\;0\;} + \\
& p_{23}\underline{\;0\;} + p_{22}x_{11} + p_{21}x_{12} + \\
& p_{13}\underline{\;0\;} + p_{12}x_{21} + p_{11}x_{22}
\end{aligned}
$$

Again, we need to impose boundary conditions:

$$
\begin{array}{cccc}
p_{33} & p_{32} & p_{31} & \\
 & & & \\
p_{23} & \left[\begin{array}{ccc} \color{red}{x_{11}} & x_{12} & x_{13} \\ \color{blue}{p_{22}} & \color{blue}{p_{21}} & \\ x_{21} & x_{22} & x_{23} \\ \color{blue}{p_{12}} & \color{blue}{p_{11}} & \\ x_{31} & x_{32} & x_{33} \end{array}\right] & & \\
p_{13} & & &
\end{array}
\qquad
\begin{bmatrix} \color{red}{b_{11}} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}
$$

Periodic:

$$
\begin{aligned}
b_{11} \quad = \quad & p_{33}\underline{\,x_{33}\,} + p_{32}\underline{\,x_{31}\,} + p_{31}\underline{\,x_{32}\,} + \\
& p_{23}\underline{\,x_{13}\,} + p_{22}x_{11} + p_{21}x_{12} + \\
& p_{13}\underline{\,x_{23}\,} + p_{12}x_{21} + p_{11}x_{22}
\end{aligned}
$$

Again, we need to impose boundary conditions:

$$
\begin{array}{cccc}
p_{33} & p_{32} & p_{31} & \\
\end{array}
$$

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

Reflexive:

$$
\begin{aligned}
b_{11} \quad = \quad & p_{33}\underline{\ x_{11}\ } + p_{32}\underline{\ x_{11}\ } + p_{31}\underline{\ x_{12}\ } + \\
& p_{23}\underline{\ x_{11}\ } + p_{22}x_{11} \ + p_{21}x_{12} \ + \\
& p_{13}\underline{\ x_{21}\ } + p_{12}x_{21} \ + p_{11}x_{22}
\end{aligned}
$$

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, and BHHB (notation: $\mathrm{B}_{\mathrm{HT}}^{\mathrm{TT}}{+}_{\mathrm{HH}}^{\mathrm{TH}}\mathrm{B}$?)

Legend:

BTTB: Block Toeplitz with Toeplitz blocks

BCCB: Block circulant with circulant blocks

BTHB: Block Toeplitz with Hankel blocks

BHTB: Block Hankel with Toeplitz blocks

BHHB: Block Hankel with Hankel blocks

Zero Boundary Conditions $\Rightarrow$ BTTB matrix

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

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

Periodic Boundary Conditions ⇒ BCCB matrix

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

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

Reflecive Boundary Conditions $\Rightarrow \ldots$

With **reflexive** boundary conditions **A** is much more complicated.
For example, the first row of **A** is:

$$[\, p_{22} + p_{23} + p_{32} + p_{33},\ p_{12} + p_{13},\ 0,\ p_{21} + p_{31}, p_{11},\ 0,\ 0,\ 0,\ 0\,]$$

It can be shown that **A** is a sum of four structured matrices, with BTTB, BTHB, BHTB, and BHHB structure, respectively.

Separable Blur

Horizontal and vertical components separate.

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

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

Separable Blur

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 Blur

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

Separable Blur

We see that for this special PSF we obtain (with zero BC):

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

Here $\otimes$ denotes the *Kronecker product*.

The Wonderful World of Kronecker Products

Definition:

$$\mathbf{A}_\mathrm{r} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \Rightarrow \quad \mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c} = \begin{bmatrix} a_{11}\mathbf{A}_\mathrm{c} & a_{12}\mathbf{A}_\mathrm{c} \\ a_{21}\mathbf{A}_\mathrm{c} & a_{22}\mathbf{A}_\mathrm{c} \end{bmatrix}$$

Transposition and inversion:

$$\begin{aligned} (\mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c})^T &= \mathbf{A}_\mathrm{r}^T \otimes \mathbf{A}_\mathrm{c}^T \\ (\mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c})^{-1} &= \mathbf{A}_\mathrm{r}^{-1} \otimes \mathbf{A}_\mathrm{c}^{-1} \end{aligned}$$

SVD:

$$(\mathbf{U}_\mathrm{r}\boldsymbol{\Sigma}_\mathrm{r}\mathbf{V}_\mathrm{r}^T) \otimes (\mathbf{U}_\mathrm{c}\boldsymbol{\Sigma}_\mathrm{c}\mathbf{V}_\mathrm{c}^T) = (\mathbf{U}_\mathrm{r} \otimes \mathbf{U}_\mathrm{c})(\boldsymbol{\Sigma}_\mathrm{r} \otimes \boldsymbol{\Sigma}_\mathrm{c})(\mathbf{V}_\mathrm{r} \otimes \mathbf{V}_\mathrm{c})^T$$

Matrix-vector product:

$$(\mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c})\,\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{A}_\mathrm{c}\,\mathbf{X}\,\mathbf{A}_\mathrm{r}^T)$$

Separable Blur with Boundary Conditions

Similar structures occur for other boundary conditions:

$$\mathbf{A} = \mathbf{A}_r \otimes \mathbf{A}_c$$

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

Summary of Matrix Structures

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

Computations with BCCB Matrices

Recall that with periodic boundary conditions **A** is a BCCB matrix:

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

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

The One-Dimensional Discrete Fourier Transform

If $\mathbf{x} \in \mathbb{R}^n$ then $\hat{\mathbf{x}} = DFT(\mathbf{x}) \in \mathbb{C}^n$ is defined by

$$\hat{x}_k = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} x_j \, \exp(-2\pi\hat{\imath}(j-1)(k-1)/n) \;, \quad \hat{\imath} = \sqrt{-1} \;.$$

There exists a unitary matrix $\mathbf{F}_n \in \mathbb{C}^{n \times n}$ such that

$$\hat{\mathbf{x}} = \sqrt{n}\, \mathbf{F}_n \mathbf{x} \qquad \Leftrightarrow \qquad \mathbf{x} = \frac{1}{\sqrt{n}} \mathbf{F}_n^* \hat{\mathbf{x}}$$

in which $\mathbf{F}_n^* = \text{conj}(\mathbf{F}_n)^T$.

The Two-Dimensional DFT

If $\mathbf{X} \in \mathbb{C}^{m \times n}$ then the 2-D DFT is defined by

$$\widehat{\mathbf{X}} = (\sqrt{m}\,\mathbf{F}_m)\,\mathbf{X}\,(\sqrt{n}\,\mathbf{F}_n)^* = \sqrt{N}\,\mathbf{F}_m\,\mathbf{X}\,\mathbf{F}_n^*$$

with $N = m\,n$ (1-D DFTs along the columns and rows of $\mathbf{X}$).

From the Kronecker product relations we get

$$\mathrm{vec}(\widehat{\mathbf{X}}) = \sqrt{N}\,\big(\mathrm{conj}(\mathbf{F}_n) \otimes \mathbf{F}_m\big)\mathrm{vec}(\mathbf{X}) = \sqrt{N}\,\mathbf{F}\,\mathrm{vec}(\mathbf{X})\ ,$$

where $\mathbf{F} = \mathrm{conj}(\mathbf{F}_n) \otimes \mathbf{F}_m$.

Important BCCB Matrix Property

- Every BCCB matrix has the same set of eigenvectors:

$$\mathbf{A} = \mathbf{F}^* \boldsymbol{\Lambda} \, \mathbf{F} \quad \left(= \mathbf{F}^* \boldsymbol{\Lambda} \, (\mathbf{F}^*)^*\right)$$

  where

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

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

  $\mathbf{F}$ times a vector requires $O(N \log N)$ flops

  using the 2-D Fast Fourier Transform (FFT) algorithm.

FFT Computations

In Matlab, if $\mathbf{A} = \mathbf{F}^* \mathbf{\Lambda} \, \mathbf{F}$ is $N \times N$, then:

- `fft2` $\leftrightarrow \sqrt{N}\, \mathbf{F}$ and `ifft2` $\leftrightarrow \frac{1}{\sqrt{N}}\, \mathbf{F}^*$
- Specifically, the following operations are equivalent:
    - $\sqrt{N}\, \mathbf{F}\, \mathbf{x} \Leftrightarrow \texttt{fft2}(\mathbf{X})$
    - $\frac{1}{\sqrt{N}}\, \mathbf{F}^*\, \mathbf{x} \Leftrightarrow \texttt{ifft2}(\mathbf{X})$

  where $\mathbf{x} = \text{vec}(\mathbf{X})$, and $\mathbf{X}$ is $m \times n$ with $N = mn$.

Eigenvalues of BCCB Matrix

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

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

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

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

Computing the Eigenvalues of BCCB Matrix

Thus, to compute eigenvalues of $\mathbf{A}$, we need to:

- Multiply the matrix $\sqrt{N}\mathbf{F}$ by the first column of $\mathbf{A}$.
- Or, equivalently, apply `fft2` to a two-dimensional array containing the elements of the first column of $\mathbf{A}$.
- Can get this array from the PSF:

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

$\mathbf{P}$ first column of $\mathbf{A}$

`circshift(`$\mathbf{P}$`, 1-[2,2])`

Efficient BCCB Computations

Thus, for zero boundary conditions, we have a BCCB matrix defined by:

- the PSF array $\mathbf{P}$
- the center of PSF = `[row, col]`

To compute eigenvalues (spectral values) in this case:

```
S = fft2( circshift(P, 1-center) );
```

Note that `S` is an array, not a vector, and eigenvalues are not sorted.

Additional BCCB Computations

If **A** is the BCCB matrix defined by the PSF array **P**, and

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

then to compute **b** use

```
S = fft2( circshift(P, 1 - center) );
B = ifft2(S .* fft2(X));
B = real(B);
```

where

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

Small PSF Arrays. If the PSF array **P** is *smaller* than the **B** and **X** images, then use our Matlab function `padPSF` to embed the $p \times q$ array **P** in a larger array of size $m \times n$.

Additional BCCB Computations

If **A** is the BCCB matrix defined by the PSF array **P**, and

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

then to compute **x** use

```
S = fft2( circshift(P, 1 - center) );
X = ifft2(fft2(B) ./ S);
X = real(X);
```

where

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

BTTB+BTHB+BHTB+BHHB Matrices

With reflexive boundary conditions **A** is a

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

matrix defined by the PSF.

Double 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 (k, k)
- $\widetilde{\mathbf{P}} = \texttt{fliplr}(\widetilde{\mathbf{P}}) = \texttt{flipud}(\widetilde{\mathbf{P}})$

BTTB+BTHB+BHTB+BHHB Matrix Properties

If the PSF satisfies the double 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

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

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

DCT Computations

With Matlab's the image processing toolbox, if $\mathbf{A} = \mathbf{C}^T \boldsymbol{\Lambda}\, \mathbf{C}$ is $N \times N$, then:

- `dct2` $\leftrightarrow \mathbf{C}$ and `idct2` $\leftrightarrow \mathbf{C}^T$
- Specifically, the following operations are equivalent:
  - $\mathbf{C}\,\mathbf{x} \Leftrightarrow$ `dct2`$(\mathbf{X})$
  - $\mathbf{C}^T\mathbf{x} \Leftrightarrow$ `idct2`$(\mathbf{X})$

  where $\mathbf{x} = \mathrm{vec}(\mathbf{X})$, and $\mathbf{X}$ is $m \times n$ with $N = mn$.

Without the image processing toolbox, use our codes:

- `dct2` $\rightarrow$ `dcts2` and `idct2` $\rightarrow$ `idcts2`.

DCT Relations and Eigenvalues

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

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

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

More DCT Relations

Thus, to compute eigenvalues of $\mathbf{A}$, we need to:

- Multiply the matrix $\mathbf{C}$ to the first column of $\mathbf{A}$.
- Or, equivalently, apply `dct2` to a two-dimensional array containing the elements of the first column of $\mathbf{A}$.
- Can get this array by adding four shifted PSFs, which we have implemented as:

```
dctshift(P, center)
```

- We also need the first column of $\mathbf{C}$, i.e., $\mathbf{c}_1 = \mathbf{C}\,\mathbf{e}_1$.
- Note that $\mathbf{e}_1 = \text{vec}(\texttt{e1})$ with

```
e1 = zeros(m,n); e1(1,1) = 1;
```

- Thus we get the desired column via `dct2(e1)`.

Efficient Computations with DCT

Thus, for reflexive boundary conditions, with

- doubly symmetric PSF **P**
- center of PSF = `[row, col]`

To compute eigenvalues (spectral values) in this case:

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

Additional DCT Computations

If $\mathbf{A}$ is defined by a doubly 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 $\mathbf{b}$ use

```
e1 = zeros(size(P)); e1(1,1) = 1;
S = dct2( dctshift(P, center) ) ./ 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 **A** is defined by a doubly symmetric PSF with reflexive boundary conditions, and

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

then to compute **x** use

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

where

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

Separable PSFs and Kronecker Products

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.

Exploiting Kronecker Product Properties

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';
```

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';
```

We can compute SVD of small matrices:

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

Then

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

Note: Do not need to explicitly form the big matrices

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

To compute inverse solution from SVD of small matrices:

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

is equivalent to

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

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';
```

Getting $\mathbf{A}_r$ and $\mathbf{A}_c$ from the PSF Array

To construct $\mathbf{A}_r$ and $\mathbf{A}_c$ we must find $\mathbf{r}$ and $\mathbf{c}$ such that

$$\mathbf{P} = \mathbf{c}\mathbf{r}^T$$

- Compute the SVD: $\mathbf{P} = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \mathbf{u}_i \sigma_i \mathbf{v}_i^T$
- If $\mathbf{P}$ has rank $= 1$, then $\sigma_2 = \sigma_3 = \cdots = 0$, and

$$\mathbf{c} = \sqrt{\sigma_1}\mathbf{u}_1 \quad \mathbf{r} = \sqrt{\sigma_1}\mathbf{v}_1$$

- If $\mathbf{P}$ has rank $\neq 1$, then

$$\mathbf{c} = \sqrt{\sigma_1}\mathbf{u}_1 \quad \mathbf{r} = \sqrt{\sigma_1}\mathbf{v}_1$$

  give the approximations

$$\mathbf{P} \approx \mathbf{c}\,\mathbf{r}^T \quad \text{and} \quad \mathbf{A} \approx \mathbf{A}_r \otimes \mathbf{A}_c$$

Some Comments to the Matlab Code

- The singular vectors **r** and **c** can be computed using the `svd` or `svds` functions.
- Since we need at most two singular values, `svds` is convenient:

```
[U, S, V] = svds(P, 2);
```

  If `P` is $m \times n$ then `U` is $m \times 2$, `V` is $n \times 2$ and `S` is 2×2.

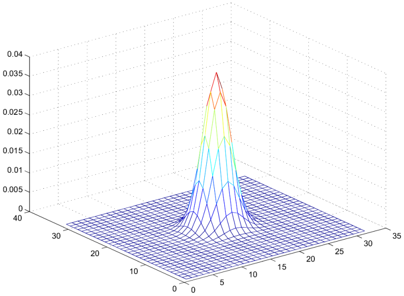
- Check to see if **P** is separable. For example, if

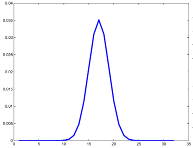
```
S(2,2)/S(1,1) > small_tol
```

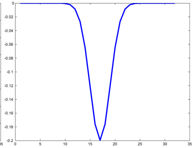
  then **P** is not separable.

A Matlab Example

- `P = psfGauss(32);`
- `mesh(P)`
- `plot(P(:,16)`
- `[U, S, V] = svds(P,2);`
- `c = sqrt(S(1,1))*U(:,1);`
- `r = sqrt(S(1,1))*V(:,1);`
- `mesh(c*r')`
- `plot(c)`







Check sign of singular vectors and change if necessary.

Construct $\mathbf{A}_r$ and $\mathbf{A}_c$

Given $\mathbf{r}$ and $\mathbf{c}$:

- zero BC: build Toeplitz $\mathbf{A}_r$, $\mathbf{A}_c$
- periodic BC: build circulant $\mathbf{A}_r$, $\mathbf{A}_c$
- reflexive BC: build Toeplitz-plus-Hankel $\mathbf{A}_r$, $\mathbf{A}_c$

Construct $\mathbf{A}_\mathrm{r}$ and $\mathbf{A}_\mathrm{c}$ for Zero BC

Suppose

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}, \quad \texttt{center} = [2, 3]$$

Then, with zero BC

$$\mathbf{A} = \mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c} = \begin{bmatrix} r_3 & r_2 & r_1 & \\ r_4 & r_3 & r_2 & r_1 \\ & r_4 & r_3 & r_2 \\ & & r_4 & r_3 \end{bmatrix} \otimes \begin{bmatrix} c_2 & c_1 & \\ c_3 & c_2 & c_1 \\ & c_3 & c_2 \end{bmatrix}$$

If $k = 3 =$ `center(2)`, then $\mathbf{A}_\mathrm{r} =$ `toeplitz(col,row)`, where

$$\begin{aligned} \texttt{col} &= \begin{bmatrix} r_k & r_{k+1} & \cdots & r_n & 0 & \cdots & 0 \end{bmatrix} \\ \texttt{row} &= \begin{bmatrix} r_k & r_{k-1} & \cdots & r_1 & 0 & \cdots & 0 \end{bmatrix} \end{aligned}$$

If $k = 2 =$ `center(1)`, then $\mathbf{A}_\mathrm{c} =$`toeplitz(col,row)`, where

$$\begin{aligned} \texttt{col} &= \begin{bmatrix} c_k & c_{k+1} & \cdots & c_n & 0 & \cdots & 0 \end{bmatrix} \\ \texttt{row} &= \begin{bmatrix} c_k & c_{k-1} & \cdots & c_1 & 0 & \cdots & 0 \end{bmatrix} \end{aligned}$$

$$\mathbf{A}_\mathrm{r} = \begin{bmatrix} r_3 & r_2 & {\color{blue}r_1} & \\ r_4 & r_3 & {\color{blue}r_2} & r_1 \\ & r_4 & {\color{blue}r_3} & r_2 \\ & & {\color{blue}r_4} & r_3 \end{bmatrix} \qquad \mathbf{A}_\mathrm{c} = \begin{bmatrix} c_2 & {\color{blue}c_1} & \\ c_3 & {\color{blue}c_2} & c_1 \\ & {\color{blue}c_3} & c_2 \end{bmatrix}$$

Construct $\mathbf{A}_r$ and $\mathbf{A}_c$ for Zero BC in Matlab

Matlab function to build Toeplitz $\mathbf{A}_r$, $\mathbf{A}_c$, given

- middle column defining entries of matrix: c $=$ $\mathbf{r}$ or $\mathbf{c}$
- loc. of center (diagonal) entry: k $=$ center(1) or center(2)

```
function T = buildToep(c, k)
  n = length(c);
  col = zeros(n,1); row = col;
  col(1:n-k+1) = c(k:n);
  row(1:k) = c(k:-1:1);
  T = toeplitz(col, row);
end
```

Then, given P = c*r' and center of P,

```
Ac = buildToep(c, center(1));
Ar = buildToep(r, center(2));
```

Construct $\mathbf{A}_\mathrm{r}$ and $\mathbf{A}_\mathrm{c}$ for Periodic BC

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}, \quad \texttt{center} = [2, 3]$$

Then, with periodic BC

$$\mathbf{A} = \mathbf{A}_\mathrm{r} \otimes \mathbf{A}_\mathrm{c} = \begin{bmatrix} r_3 & r_2 & r_1 & r_4 \\ r_4 & r_3 & r_2 & r_1 \\ r_1 & r_4 & r_3 & r_2 \\ r_2 & r_1 & r_4 & r_3 \end{bmatrix} \otimes \begin{bmatrix} c_2 & c_1 & c_3 \\ c_3 & c_2 & c_1 \\ c_1 & c_3 & c_2 \end{bmatrix}$$

If $k = 3 =$ `center(2)`, then $\mathbf{A}_\mathrm{r} =$ `toeplitz(col,row)`, where

$$\begin{aligned} \texttt{col} &= \begin{bmatrix} r_k & r_{k+1} & \cdots & r_n & r_1 & \cdots & r_{k-1} \end{bmatrix} \\ \texttt{row} &= \begin{bmatrix} r_k & r_{k-1} & \cdots & r_1 & r_n & \cdots & r_{k+1} \end{bmatrix} \end{aligned}$$

If $k = 2 =$ `center(1)`, then $\mathbf{A}_\mathrm{c} =$ `toeplitz(col,row)`, where

$$\begin{aligned} \texttt{col} &= \begin{bmatrix} c_k & c_{k+1} & \cdots & c_n & c_1 & \cdots & c_{k-1} \end{bmatrix} \\ \texttt{row} &= \begin{bmatrix} c_k & c_{k-1} & \cdots & c_1 & c_n & \cdots & c_{k+1} \end{bmatrix} \end{aligned}$$

$$\mathbf{A}_\mathrm{r} = \begin{bmatrix} r_3 & r_2 & r_1 & r_4 \\ r_4 & r_3 & r_2 & r_1 \\ r_1 & r_4 & r_3 & r_2 \\ r_2 & r_1 & r_4 & r_3 \end{bmatrix} \qquad \mathbf{A}_\mathrm{c} = \begin{bmatrix} c_2 & c_1 & c_3 \\ c_3 & c_2 & c_1 \\ c_1 & c_3 & c_2 \end{bmatrix}$$

Construct $\mathbf{A}_r$ and $\mathbf{A}_c$ for Periodic BC in Matlab

Matlab function to build circulant $\mathbf{A}_r$, $\mathbf{A}_c$, given

- middle column defining entries of matrix: $c = \mathbf{r}$ or $\mathbf{c}$
- loc. of center (diagonal) entry: `k` = `center(1)` or `center(2)`

```
function T = buildCirc(c, k)
  n = length(c);
  col = [c(k:n); c(1:k-1)];
  row = [c(k:-1:1); c(n:-1:k+1)];
  T = toeplitz(col, row);
end
```

Then, given `P = c*r'` and center of `P`

```
Ac = buildCirc(c, center(1));
Ar = buildCirc(r, center(2));
```

Construct $\mathbf{A}_\mathrm{r}$ and $\mathbf{A}_\mathrm{c}$ for Reflexive BC

In this case

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

where

- $\mathbf{A}_\mathrm{r}$ = Toeplitz + Hankel
- $\mathbf{A}_\mathrm{c}$ = Toeplitz + Hankel
- Use `buildToep` for Toeplitz parts.
- How to get Hankel parts?

Recall Reflexive BC for 1-D Problems

$$
\left[\begin{array}{cc|ccccc|cc}
c_5 & c_4 & c_3 & c_2 & c_1 & & & & \\
 & c_5 & c_4 & c_3 & c_2 & c_1 & & & \\
 & & c_5 & c_4 & c_3 & c_2 & c_1 & & \\
 & & & c_5 & c_4 & c_3 & c_2 & c_1 & \\
 & & & & c_5 & c_4 & c_3 & c_2 & c_1
\end{array}\right]
\left[\begin{array}{c}
x_2 \\ x_1 \\ \hline x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \hline x_5 \\ x_4
\end{array}\right]
$$

$$
= \left(
\begin{bmatrix}
c_3 & c_2 & c_1 & & \\
c_4 & c_3 & c_2 & c_1 & \\
c_5 & c_4 & c_3 & c_2 & c_1 \\
 & c_5 & c_4 & c_3 & c_2 \\
 & & c_5 & c_4 & c_3
\end{bmatrix}
+
\begin{bmatrix}
c_4 & c_5 & & & \\
c_5 & & & & \\
 & & & & \\
 & & & & c_1 \\
 & & & c_1 & c_2
\end{bmatrix}
\right)
\begin{bmatrix}
x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5
\end{bmatrix}
$$

Construct $\mathbf{A}_r$ and $\mathbf{A}_c$ for Reflexive BC in Matlab

Matlab function to build Hankel part for $\mathbf{A}_r$, $\mathbf{A}_c$, given

- middle column defining entries of matrix: c = **r** or **c**
- loc. of center (diagonal) entry: `k = center(1)` or `center(2)`

```
function T = buildHank(c, k)
  n = length(c);
  col = zeros(n,1); row = col;
  col(1:n-k) = c(k+1:n);
  row(n-k+2:n) = c(1:k-1);
  T = hankel(col, row);
end
```

Then, given `P = c*r'` and center of P

```
Ac = buildToep(c, center(1)) + buildHank(c, center(1));
Ar = buildToep(r, center(2)) + buildHank(r, center(2));
```

Construct $\mathbf{A}_r$ and $\mathbf{A}_c$ for All Three BC

```
[U, S, V] = svds(P, 2);
c = sqrt(S(1,1))*U(:,1);
r = sqrt(S(1,1))*V(:,1);
switch BC
  case 'zero'
   Ac = buildToep(c, center(1));
   Ar = buildToep(r, center(2));
  case 'reflexive'
   Ac = buildToep(c, center(1)) + buildHank(c, center(1));
   Ar = buildToep(r, center(2)) + buildHank(r, center(2));
  case 'periodic'
   Ac = buildCirc(c, center(1));
   Ar = buildCirc(r, center(2));
end
```

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-dim FFT
Doubly sym.	Reflexive	BTTB+BTHB +BHTB+BHHB	2-dim DCT
Separable	Arbitrary	Kronecker product	2 small SVDs

Creating Realistic Test Data

Issues that must be considered:

- Small PSF.
- Creating blurred image without imposing artificial boundary conditions.
- Additive noise.

PSF Sizes

- It is often the case that

  `size(P) < size(B)` and `size(X)`

- In this case we should "pad" `P` with zeros to increase size.
- Since we do this a lot, it is useful to have a function:

```
function Ppad = padPSF(P, size)
  Ppad = zeros(size);
  Ppad(1:size(P,1), 1:size(P,2)) = P;
end
```

- With this padding, center of `Ppad` = center of `P`

Creating Blurred Image

If we are *given* blurred image data:

- We try to make a best guess at what boundary condition is most realistic.
- Construct **A** using this boundary condition.

If we are *creating* blurred image data:

- We should simulate actual boundaries of an infinite scene.
- This can be done by blurring a large “true” image scene.
- Then extract the central part of the image.

Creating Blurred Image: Matlab example

```
Xbig = double(imread('iograyBorder.tif'));
[P, center] = psfGauss([512,512], s);
Pbig = padPSF(P, size(Xbig));
Sbig = fft2(circshift(Pbig, 1-center));
Bbig = real(ifft2(Sbig .* fft2(Xbig)));
X = Xbig(51:562,51:562);
B = Bbig(51:562,51:562);
```

Use `P`, `center`, `B`, `X` as realistic (noise free) test data.

Additive Noise

Additive noise $\Rightarrow$ add random perturbations to blurred image.

- Use Matlab `randn` function.
- Scale perturbations to data.
- Add to blurred image.

For example,

```
E = randn(size(B));
E = E / norm(E(:));
B = B + 0.01*norm(B(:))*E;
```

generates "1% noise."