

---

# Numerical Linear Algebra and Image Restoration

---

Maui High Performance Computing Center  
Wednesday, October 8, 2003

James G. Nagy  
Emory University  
Atlanta, GA

Thanks to:

- AFOSR, Dave Tyler, Stuart Jefferies, Sudhaker Prasad
- NSF (MPS/Computational Math)
- Emory University
- Julie Kamm (Raytheon), Lisa Perrone (Emory)

## Basic Problem

Linear system of equations

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

where

- $A$ ,  $\mathbf{b}$  are known
  - $A$  is large, structured, **ill-conditioned**
  - Goal: Compute an approximation of  $\mathbf{x}$
- 

**Example:** Image restoration

- $\mathbf{b}$  = observed image
- $A$  = matrix defined by point spread function (PSF)
- $\mathbf{e}$  = noise
- $\mathbf{x}$  = true image

## Computational Considerations

- Ill-conditioned  $A$  implies

$$\hat{\mathbf{x}} = A^{-1}\mathbf{b} = \mathbf{x} + A^{-1}\mathbf{e}$$

is corrupted with noise.

- Stabilize numerical scheme using *regularization*.
- 

Examples of regularization:

1. Filtering

Pseudo inverse, Wiener, Tikhonov, ...

2. Iterative

Richardson-Lucy, EM, Conjugate gradients, ...

## Computational Considerations

Efficient implementation of **filters** depends on structure of  $A$ . We consider

- Circulant
- Toeplitz
- Toeplitz+Hankel
- Kronecker product

**Iterative** methods can be accelerated **using filters** as “pre-conditioners” .

---

## Course Aims

**Wednesday:** Background, notation, tools.

**Thursday:** Kronecker products in image restoration.

**Friday:** Spatially variant blurs and preconditioning.

---

### Wednesday

1. Tools (matrix decompositions) for building filters.
2. Some filtering methods.
3. Structured matrices in image restoration.
4. Matrix decompositions for structured matrices.

## Tools for Building Filters

### Singular value decomposition (SVD):

Any matrix  $A$  can be written as

$$A = U\Sigma V^T$$

where

- $U^T U = I$  (orthogonal)
- $V^T V = I$
- $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_N)$ ,  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_N \geq 0$

---

Note: For image restoration,

$$\sigma_1 \approx 1, \quad \sigma_N \approx 0$$

$$\Rightarrow \text{condition}(A) = \frac{\sigma_1}{\sigma_N} \gg 1$$

## Tools for Building Filters

### Singular value decomposition:

In Matlab:

```
>> [U, S, V] = svd(A);
```

Computational cost for an  $n \times n$  matrix  $A$

- FLOPS:  $O(n^3)$
- Storage:  $O(n^2)$

**Remark:** Fast algorithms are possible for sparse matrices, but storage is difficult to reduce.

In Matlab:

```
>> [U, S, V] = svds(A);
```

## Tools for Building Filters

### Spectral decomposition:

Some matrices  $A$  can be written as

$$A = V^* \Lambda V$$

where

- $V^*$  = complex conjugate transpose
- $V^*V = I$  (unitary)
- $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_N)$ ,

$\lambda_i$  = eigenvalues,

$i$ -th column of  $V = \mathbf{v}_i$  = eigenvectors

## Tools for Building Filters

### Spectral decomposition:

In Matlab:

```
>> [V, E] = eig(A);
```

Computational cost for an  $n \times n$  matrix  $A$

- FLOPS:  $O(n^3)$
- Storage:  $O(n^2)$

**Remark:** Fast algorithms are possible for sparse matrices, but storage is difficult to reduce.

In Matlab:

```
>> [V, E] = eigs(A);
```

## Filtering Methods

### Inverse solution using SVD:

If  $A = U\Sigma V^T$  (or  $A = V^*\Lambda V$ ), then

$$\begin{aligned} \mathbf{x} &= A^{-1}\mathbf{b} \\ &= V\Sigma^{-1}U^T\mathbf{b} \\ &= \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix} \begin{bmatrix} 1/\sigma_1 & & \\ & \cdots & \\ & & 1/\sigma_n \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_n^T \end{bmatrix} \mathbf{b} \\ &= \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^T \mathbf{b} / \sigma_1 \\ \vdots \\ \mathbf{u}_n^T \mathbf{b} / \sigma_n \end{bmatrix} \\ &= \sum_{i=1}^n \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i \end{aligned}$$

## Filtering Methods

Inverse solution for noisy, ill-posed problems:

If  $A = U\Sigma V^T$ , then

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

## Filtering Methods

**Filtering Goal:** Do not let error dominate computed solution.

$$\begin{aligned} \mathbf{x}_{\text{filt}} &= \sum_{i=1}^n \phi_i \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i \\ &= V \begin{bmatrix} \frac{\phi_1}{\sigma_1} & & \\ & \dots & \\ & & \frac{\phi_n}{\sigma_n} \end{bmatrix} U^T \mathbf{b} \end{aligned}$$

where the "filter factors" satisfy

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

## Filtering Methods

Frequency space representation of filter:

If  $U^T = V^T =$  Fourier transform, then

$$\begin{aligned}\mathbf{x}_{\text{filt}} &= V \begin{bmatrix} \frac{\phi_1}{\sigma_1} & & \\ & \dots & \\ & & \frac{\phi_n}{\sigma_n} \end{bmatrix} U^T \mathbf{b} \\ V^T \mathbf{x}_{\text{filt}} &= \begin{bmatrix} \frac{\phi_1}{\sigma_1} & & \\ & \dots & \\ & & \frac{\phi_n}{\sigma_n} \end{bmatrix} U^T \mathbf{b} \\ \hat{\mathbf{x}}_{\text{filt}} &= \begin{bmatrix} \frac{\phi_1}{\sigma_1} & & \\ & \dots & \\ & & \frac{\phi_n}{\sigma_n} \end{bmatrix} \hat{\mathbf{b}}\end{aligned}$$

is the frequency space representation of the filter.

## Filtering Methods

Some examples of filters:

1. Pseudo-inverse (truncated SVD)

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

2. Tikhonov

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

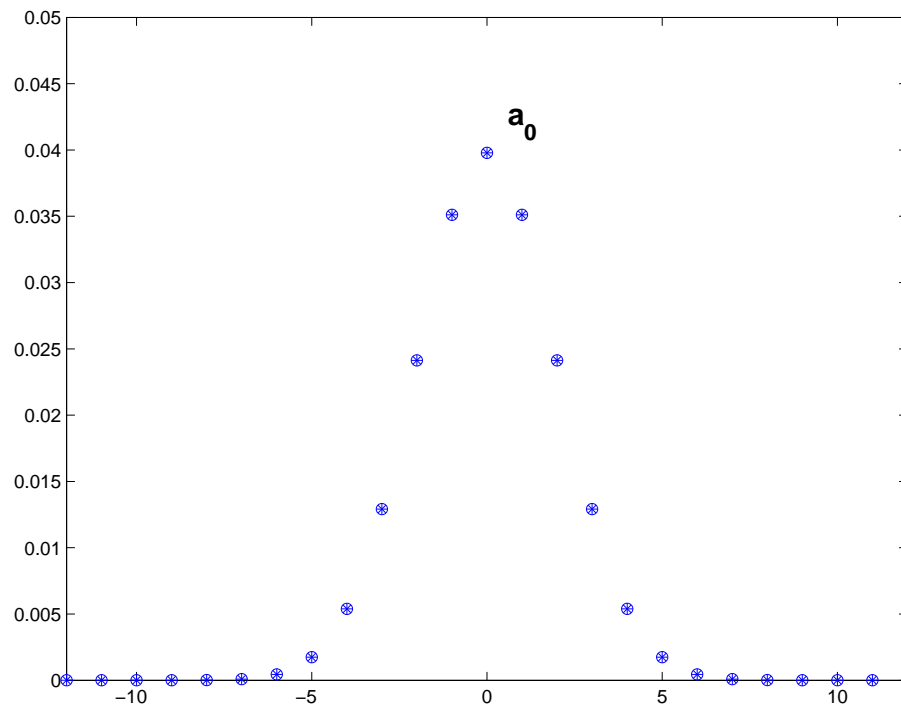
3. Wiener

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

# Structured Matrices in Image Restoration

## Toeplitz Matrix

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & 0 & 0 \\ a_1 & a_0 & a_{-1} & a_{-2} & 0 \\ a_2 & a_1 & a_0 & a_{-1} & a_{-2} \\ 0 & a_2 & a_1 & a_0 & a_{-1} \\ 0 & 0 & a_2 & a_1 & a_0 \end{bmatrix}$$



# Structured Matrices in Image Restoration

Circulant Matrix (Toeplitz + periodic)

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & a_2 & a_1 \\ a_1 & a_0 & a_{-1} & a_{-2} & a_2 \\ a_2 & a_1 & a_0 & a_{-1} & a_{-2} \\ a_{-2} & a_2 & a_1 & a_0 & a_{-1} \\ a_{-1} & a_{-2} & a_2 & a_1 & a_0 \end{bmatrix}$$

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & 0 & 0 \\ a_1 & a_0 & a_{-1} & a_{-2} & 0 \\ a_2 & a_1 & a_0 & a_{-1} & a_{-2} \\ 0 & a_2 & a_1 & a_0 & a_{-1} \\ 0 & 0 & a_2 & a_1 & a_0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & a_2 & a_1 \\ 0 & 0 & 0 & 0 & a_2 \\ 0 & 0 & 0 & 0 & 0 \\ a_{-2} & 0 & 0 & 0 & 0 \\ a_{-1} & a_{-2} & 0 & 0 & 0 \end{bmatrix}$$

# Structured Matrices in Image Restoration

## Hankel Matrix

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 \\ a_1 & a_2 & a_3 & a_4 & a_5 \\ a_2 & a_3 & a_4 & a_5 & a_6 \\ a_3 & a_4 & a_5 & a_6 & a_7 \\ a_4 & a_5 & a_6 & a_7 & a_8 \end{bmatrix}$$

## Toeplitz + Hankel (Toeplitz + reflection)

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & 0 & 0 \\ a_1 & a_0 & a_{-1} & a_{-2} & 0 \\ a_2 & a_1 & a_0 & a_{-1} & a_{-2} \\ 0 & a_2 & a_1 & a_0 & a_{-1} \\ 0 & 0 & a_2 & a_1 & a_0 \end{bmatrix} + \begin{bmatrix} a_1 & a_2 & 0 & 0 & 0 \\ a_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{-2} \\ 0 & 0 & 0 & a_{-2} & a_{-1} \end{bmatrix}$$

# Structured Matrices in Image Restoration

## Hankel Matrix

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 \\ a_1 & a_2 & a_3 & a_4 & a_5 \\ a_2 & a_3 & a_4 & a_5 & a_6 \\ a_3 & a_4 & a_5 & a_6 & a_7 \\ a_4 & a_5 & a_6 & a_7 & a_8 \end{bmatrix}$$

## "Symmetric" Toeplitz + Hankel

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & 0 & 0 \\ a_1 & a_0 & a_1 & a_2 & 0 \\ a_2 & a_1 & a_0 & a_1 & a_2 \\ 0 & a_2 & a_1 & a_0 & a_1 \\ 0 & 0 & a_2 & a_1 & a_0 \end{bmatrix} + \begin{bmatrix} a_1 & a_2 & 0 & 0 & 0 \\ a_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_2 \\ 0 & 0 & 0 & a_2 & a_1 \end{bmatrix}$$



symmetric about the center

# Structured Matrices in Image Restoration

## Two-dimensional analogs:

e.g., block Toeplitz with Toeplitz blocks (BTTB)

$$A = \begin{bmatrix} A_0 & A_{-1} & A_{-2} & 0 & 0 \\ A_1 & A_0 & A_{-1} & A_{-2} & 0 \\ A_2 & A_1 & A_0 & A_{-1} & A_{-2} \\ 0 & A_2 & A_1 & A_0 & A_{-1} \\ 0 & 0 & A_2 & A_1 & A_0 \end{bmatrix}, \quad \text{each } A_i \text{ is Toeplitz}$$

We consider: BTTB, BCCB, BHHB, BTHB, BHTB

B  $\leftrightarrow$  block

T  $\leftrightarrow$  Toeplitz

C  $\leftrightarrow$  circulant

H  $\leftrightarrow$  Hankel

# Structured Matrices in Image Restoration

## Kronecker Products

$$A = C \otimes D = \begin{bmatrix} c_{11}D & c_{12}D & \cdots & c_{1n}D \\ c_{21}D & c_{22}D & \cdots & c_{2n}D \\ \vdots & \vdots & \cdots & \vdots \\ c_{n1}D & c_{n2}D & \cdots & c_{nn}D \end{bmatrix}$$

Note:

- If  $C$  and  $D$  are  $n \times n$  matrices, then  $A$  is  $n^2 \times n^2$
- $C$  and  $D$  could be structured (e.g., Toeplitz).

# Structured Matrices in Image Restoration

## Summary:

- **Problem:** Solve  $\mathbf{b} = A\mathbf{x} + \mathbf{e}$
- **Solution:** Compute a filtered solution:

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

- **Difficulty:** Need to compute either

Singular value decomp.:  $A = U\Sigma V^T$

Spectral decomposition:  $A = V^* \Lambda V$

How to do this efficiently?

Can structure of  $A$  be exploited?

## Decompositions for Structured Matrices

### Circulant and BCCB matrices:

Can use FFT to get spectral decomposition:

$$A = \mathcal{F}^* \Lambda \mathcal{F}$$

where

- $\mathcal{F}$  = unitary discrete Fourier transform matrix
- $\Lambda \mathbf{1} = \sqrt{n} \mathcal{F} \mathbf{a}$ ,  $\mathbf{a}$  = 1st column of  $A$

Computational cost using FFTs:

- FLOPs =  $O(n \log n)$
- Storage =  $O(n)$

## Decompositions for Structured Matrices

### Circulant and BCCB matrices:

In Matlab, if  $a$  is the 1st column of circulant  $A$ :

- To compute eigenvalues:

```
>> Eigs = fft(a);
```

- To solve  $Ax = b$ :

```
>> x = ifft( fft(b) ./ Eigs );
```

- If  $f$  is a vector containing filter factors,  $\phi_i$ :

```
>> x_filt = ifft( fft(b) .* (f ./ Eigs) );
```

## Decompositions for Structured Matrices

### Symmetric Toeplitz + Hankel Matrices:

Can use discrete cosine transform (DCT):

$$A = c^T \Lambda c$$

where

- $C$  = orthogonal discrete cosine transform matrix
- $\Lambda_1 = C a ./ c$ ,  $a$  = 1st col of  $A$ ,  $c$  = 1st col of  $C$

Computational cost using fast DCTs:

- FLOPs =  $O(n \log n)$
- Storage =  $O(n)$

## Decompositions for Structured Matrices

### Symmetric Toeplitz + Hankel Matrices:

In Matlab, if  $a$  is the 1st column of  $A$ :

- To compute eigenvalues:

```
>> Eigs = dct(a ./ dct(eye(length(a),1)));
```

- To solve  $Ax = b$ :

```
>> x = idct( dct(b) ./ Eigs );
```

- If  $f$  is a vector containing filter factors,  $\phi_i$ :

```
>> x_filt = idct( dct(b) .* (f ./ Eigs) );
```

## Decompositions for Structured Matrices

**Kronecker Products:**  $A = C \otimes D$

Some simple properties:

- $(C \otimes D)^T = C^T \otimes D^T$
- $(C \otimes D)^{-1} = C^{-1} \otimes D^{-1}$
- $(C_1 \otimes D_1)(C_2 \otimes D_2) = C_1 C_2 \otimes D_1 D_2$

A useful property:

- $A\mathbf{x} = \mathbf{b} \quad \Rightarrow \quad (C \otimes D)\mathbf{x} = \mathbf{b}$   
 $\quad \quad \quad \Rightarrow \quad DXC^T = B$

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

## Decompositions for Structured Matrices

**Kronecker Products:**  $A = C \otimes D$

Using these properties, can compute SVD efficiently:

- If  $C = U_c \Sigma_c V_c^T$  and  $D = U_d \Sigma_d V_d^T$ , then

$$\begin{aligned} A = C \otimes D &= (U_c \Sigma_c V_c^T) \otimes (U_d \Sigma_d V_d^T) \\ &= (U_c \otimes U_d) (\Sigma_c \otimes \Sigma_d) (V_c \otimes V_d)^T \\ &= U \Sigma V^T \\ &= \text{SVD of } A \end{aligned}$$

- Computational cost for  $N \times N = n^2 \times n^2$  matrix:

$$\text{FLOPs} = O(n^3) = O(N^{3/2}) > O(N \log N)$$

$$\text{Storage} = O(N)$$

## Decompositions for Structured Matrices

**Kronecker Products:**  $A = C \otimes D$

In Matlab, if

- $F_c$  = diagonal matrix of filter factors for  $C$
- $F_d$  = diagonal matrix of filter factors for  $D$

then a filtered solution can be computed as:

```
>> [Uc, Sc, Vc] = svd(C);  
>> [Ud, Sd, Vd] = svd(D);  
>> X_filt = Vd * (Sd \ Fd) * (Ud'*B*Uc) * (Sc \ Fc) * Vd';
```

## Example

- Gaussian PSF:  $e^{-\alpha^2(x^2+y^2)} = e^{-\alpha^2x^2}e^{-\alpha^2y^2}$
- Discretizing this, we get:

$$A = C \otimes D$$

where  $C$  and  $D$  are Toeplitz matrices.

- Simulated data:

$$B = D * X * C^T + \text{Noise}$$

---

## Summary

- **Problem:**  $\mathbf{b} = A\mathbf{x} + \mathbf{e}$
- **Solution:** Compute a filtered solution:

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

- **Efficiency:** Exploiting matrix structure, can use:

$$A = V^* \Lambda V \text{ for circulant}$$

$$A = V^* \Lambda V \text{ for symmetric Toeplitz } + \text{ Hankel}$$

$$A = U \Sigma V^T \text{ for Kronecker products}$$

## Preview for Thursday

- Determine which matrix structure is most appropriate.  
Depends on:
  - Properties of PSF, and
  - boundary conditions.
- Exploiting Kronecker product structure for non-separable PSFs.
- Implementation details for realistic data.