Lecture 5 (double session)

SINGULAR VALUE DECOMPOSITION: COMPUTATIONAL METHODS (SVD DIY)

Harrison H. Barrett
University of Arizona

SVD is a beautiful theory, but is it practical?

Goal of this lecture is to develop several methods of SVD analysis applicable to real-world problems in imaging

Next week we will introduce the Moore-Penrose pseudoinverse and give some practical algorithms for computing it.

OUTLINE

- Review of SVD basics from lecture 4
- Matrix methods for SVD

  Life in the small domain
  DD Example: Matrix representation of LSIV systems
  CD Example: Mirror modes

- Fourier integral methods for CC LSIV systems
- Other Fourier methods
- Axial systems

  Example: Through-focus optical sectioning

Review of adjoints

The adjoint of a general linear operator is defined by

$$(\mathrm{g}_2, \boldsymbol{\mathcal{H}}\mathrm{f}_1)_{\mathbb{V}} = (\boldsymbol{\mathcal{H}}^{\dagger}\mathrm{g}_2, \mathrm{f}_1)_{\mathbb{U}} \tag{1.39}$$

For a matrix, adjoint is conjugate transpose.

For a CC operator, the forward and adjoint operators are:

$$[\boldsymbol{\mathcal{H}}\mathbf{f}](\mathbf{r}_d) = g(\mathbf{r}_d) = \int_{\infty} d^q r \; h(\mathbf{r}_d, \mathbf{r})\, f(\mathbf{r})$$

$$\left[\boldsymbol{\mathcal{H}}^{\dagger}\mathbf{g}\right](\mathbf{r}) = \int_{\infty} d^s r_d \; h^{(\dagger)}(\mathbf{r}, \mathbf{r}_d)\, g(\mathbf{r}_d) = \int_{\infty} d^s r_d \, h^*(\mathbf{r}_d, \mathbf{r})\, g(\mathbf{r}_d)$$

For a CD operator, the forward and adjoint operators are:

$$[\boldsymbol{\mathcal{H}}\mathbf{f}]_m = \int_{\infty} d^q r \; h_m(\mathbf{r})\, f(\mathbf{r})$$

$$\left[\boldsymbol{\mathcal{H}}^{\dagger}\mathbf{g}\right](\mathbf{r}) = \sum_{m=1}^{M} h_m^*(\mathbf{r})\, g_m$$

Notation: Inner and outer products

An ND vector $\mathbf{a}$ can be written as an $N \times 1$ matrix, so its adjoint is $1 \times N$:

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \qquad \mathbf{a}^t = (a_1, a_2, a_3)\ , \qquad \mathbf{a}^\dagger = (a_1^*, a_2^*, a_3^*)$$

The scalar or inner product of two ND vectors can be written as

$$(\mathbf{a}, \mathbf{b}) = \mathbf{a}^\dagger \mathbf{b} = (a_1^*, a_2^*, a_3^*) \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \sum_{n=1}^{3} a_n^* b_n$$

The outer or tensor product of an ND vector and an MD vector is an $N \times M$ matrix denoted $\mathbf{b}\mathbf{a}^\dagger$:

$$\mathbf{b}\mathbf{a}^\dagger = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \begin{bmatrix} a_1^* & a_2^* & a_3^* \end{bmatrix} = \begin{bmatrix} b_1 a_1^* & b_1 a_2^* & b_1 a_3^* \\ b_2 a_1^* & b_2 a_2^* & b_2 a_3^* \\ b_3 a_1^* & b_3 a_2^* & b_3 a_3^* \\ b_4 a_1^* & b_4 a_2^* & b_4 a_3^* \end{bmatrix}$$

$$[\mathbf{b}\mathbf{a}^\dagger]_{mn} = b_m\, a_n^* \qquad \text{Rank} = 1$$

Outer products as operators

Any matrix can be regarded as an operator, and $\mathbf{ba}^\dagger$ is no exception. If $\mathbf{ba}^\dagger$ is $M \times N$, its operand $\mathbf{c}$ must be $N \times 1$ (i.e., the same size as $\mathbf{a}$), and the result of the operation is

$$\mathbf{ba}^\dagger\mathbf{c} = \mathbf{b}\left(\mathbf{a}^\dagger\mathbf{c}\right) = \left(\mathbf{a}^\dagger\mathbf{c}\right)\mathbf{b} = \text{scalar} \times \mathbf{b}$$

since $\mathbf{a}^\dagger\mathbf{c}$ is the scalar product of $\mathbf{a}$ and $\mathbf{c}$, which can be placed on either side of the vector $\mathbf{b}$. Thus $\mathbf{ba}^\dagger\mathbf{c}$ is parallel to $\mathbf{b}$ for all $\mathbf{c}$.

With this interpretation, the concept of outer product can be extended to vectors in *any* two Hilbert spaces. Let $\mathbf{u}$ and $\mathbf{f}$ be arbitrary vectors in $\mathbb{U}$ and $\mathbf{v}$ be an arbitrary vector in $\mathbb{V}$. Then $\mathbf{vu}^\dagger$ maps $\mathbb{U}$ to $\mathbb{V}$ as follows:

$$\mathbf{vu}^\dagger\mathbf{f} = \mathbf{v}\left(\mathbf{u}^\dagger\mathbf{f}\right) = \left(\mathbf{u}^\dagger\mathbf{f}\right)\mathbf{v} = \text{scalar} \times \mathbf{v}$$

Since we know how to compute scalar products of functions, this definition works for infinite-dimensional Hilbert spaces (function spaces) as well as finite-dimensional spaces.

Construction of Hermitian operators

From any linear operator $\boldsymbol{\mathcal{H}}$, we can construct two Hermitian operators, $\boldsymbol{\mathcal{H}}^\dagger\boldsymbol{\mathcal{H}}$ and $\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger$.

Example – $\boldsymbol{\mathcal{H}}$ a CD operator, $\mathbb{U} \to \mathbb{V}$

$\boldsymbol{\mathcal{H}}^\dagger\boldsymbol{\mathcal{H}}$ is a CC operator mapping $\mathbb{U} \to \mathbb{U}$ with kernel given by

$$[\boldsymbol{\mathcal{H}}^\dagger\boldsymbol{\mathcal{H}}](\mathbf{r}, \mathbf{r}') = \sum_{m=1}^{M} h_m^*(\mathbf{r})\, h_m(\mathbf{r}')$$

$\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger$ is a DD operator ($M \times M$ matrix), mapping $\mathbb{V} \to \mathbb{V}$, with matrix elements given by

$$[\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger]_{mm'} = \int d^2r\, h_m(\mathbf{r})\, h_{m'}^*(\mathbf{r})$$

Eigenanalysis of $\boldsymbol{\mathcal{H}^\dagger\mathcal{H}}$

SVD begins with eigenalysis of the Hermitian operators $\boldsymbol{\mathcal{H}^\dagger\mathcal{H}}$ and $\boldsymbol{\mathcal{H}\mathcal{H}^\dagger}$.

The eigenvalue problem for $\boldsymbol{\mathcal{H}^\dagger\mathcal{H}}$ is:

$$\boldsymbol{\mathcal{H}^\dagger\mathcal{H}}\,\mathbf{u}_n = \mu_n\,\mathbf{u}_n \qquad (1.111)\,.$$

Set $\{\mathbf{u}_n\}$ is orthonormal and complete in ND space $\mathbb{U}$ (N may be ∞):

$$\mathbf{u}_n^\dagger\mathbf{u}_m = \delta_{nm}\,, \qquad \sum_{n=1}^{N}\mathbf{u}_n\,\mathbf{u}_n^\dagger = \mathbf{I}_{\mathbb{U}}$$

Eigenvalues are real and non-negative ($\mu_n \geq 0$) and can be ordered as:

$$\mu_1 \geq \mu_2 \geq .... \geq \mu_R > 0\,, \qquad (1.114)$$

where R = rank = number of nonzero eigenvalues ($R \leq N$)

If $\mu_n = 0$, then $\mathbf{u}_n$ is a null vector of $\boldsymbol{\mathcal{H}^\dagger\mathcal{H}}$ and also of $\boldsymbol{\mathcal{H}}$:

$$\boldsymbol{\mathcal{H}^\dagger\mathcal{H}}\mathbf{u}_n = 0\,, \qquad \boldsymbol{\mathcal{H}}\mathbf{u}_n = 0$$

Eigenanalysis of $\boldsymbol{\mathcal{H}\mathcal{H}}^\dagger$

$$\boldsymbol{\mathcal{H}\mathcal{H}}^\dagger \mathbf{v}_n = \mu_n \mathbf{v}_n \,,$$

The eigenvalues are the same as for $\boldsymbol{\mathcal{H}}^\dagger \boldsymbol{\mathcal{H}}$ and hence the rank is the same. Note that $R \leq min(N, M)$.

Set $\{\mathbf{u}_n\}$ is orthonormal and complete in MD space $\mathbb{V}$ (M may be ∞):

$$\mathbf{v}_n^\dagger \mathbf{v}_m = \delta_{nm} \,, \qquad \sum_{n=1}^{M} \mathbf{v}_n \, \mathbf{v}_n^\dagger = \mathbf{I}_{\mathbb{V}}$$

If $\mu_n \neq 0$, the solutions are related by

$$\mathbf{v}_n = \frac{1}{\sqrt{\mu_n}} \boldsymbol{\mathcal{H}} \mathbf{u}_n \,, \qquad \mathbf{u}_n = \frac{1}{\sqrt{\mu_n}} \boldsymbol{\mathcal{H}}^\dagger \mathbf{v}_n$$

Representations of vectors and operators

An arbitrary object can be written as

$$\mathbf{f} = \sum_{n=1}^{N} \alpha_n \mathbf{u}_n \,, \qquad \alpha_n = \mathbf{u}_n^\dagger \mathbf{f} \,.$$

An arbitrary image (even a noisy one) can be written as

$$\mathbf{g} = \sum_{m=1}^{M} \beta_m \mathbf{v}_m \,, \qquad \beta_m = \mathbf{v}_m^\dagger \mathbf{g}$$

The imaging operator and its adjoint are:

$$\boldsymbol{\mathcal{H}} = \sum_{n=1}^{R} \sqrt{\mu_n}\, \mathbf{v}_n \mathbf{u}_n^\dagger \,, \qquad \boldsymbol{\mathcal{H}}^\dagger = \sum_{n=1}^{R} \sqrt{\mu_n}\, \mathbf{u}_n \mathbf{v}_n^\dagger$$

and the original Hermitian operators are:

$$\boldsymbol{\mathcal{H}}^\dagger \boldsymbol{\mathcal{H}} = \sum_{n=1}^{R} \mu_n\, \mathbf{u}_n \mathbf{u}_n^\dagger \qquad \boldsymbol{\mathcal{H}} \boldsymbol{\mathcal{H}}^\dagger = \sum_{n=1}^{R} \mu_n\, \mathbf{v}_n \mathbf{v}_n^\dagger$$

The SVD imaging equation

The (noise-free) imaging equation for an arbitrary linear operator is

$$\mathbf{g} = \boldsymbol{\mathcal{H}}\mathbf{f} \, .$$

The object and image can be expanded in eigenfunctions of $\boldsymbol{\mathcal{H}}^\dagger\boldsymbol{\mathcal{H}}$ and $\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger$, respectively:

$$\mathbf{f} = \sum_{n=1}^{N} \alpha_n \mathbf{u}_n \, , \qquad \mathbf{g} = \sum_{m=1}^{M} \beta_m \mathbf{v}_m \, .$$

and the coefficients are related by

$$\beta_n = \sqrt{\mu_n} \, \alpha_n$$

Matrix methods

There are two situations in which finite matrices play a key role in SVD:

- DD problems, where $\boldsymbol{\mathcal{H}}$ is the $M \times N$ matrix $\mathbf{H}$
- CD problems, where $\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger$ is an $M \times M$ matrix

In CC problems, infinite matrices arise for compact operators, but for non-compact operators such as convolutions, the discrete matrix indices have to be replaced by continuous variables (example forthcoming).

SVD for a finite matrix

$$\boldsymbol{\mathcal{H}} \rightarrow \mathbf{H}\,, \qquad (M \times N)$$

Eigenvalue problems

$$\mathbf{H}^{\dagger}\mathbf{H}\mathbf{u}_n = \mu_n\,\mathbf{u}_n\,, \qquad \mathbf{u}_n : N \times 1\,, \qquad n = 1, ..., N$$

$$\mathbf{H}\mathbf{H}^{\dagger}\mathbf{v}_m = \mu_m\,\mathbf{v}_m\,, \qquad \mathbf{v}_m : M \times 1\,, \qquad m = 1, ..., M$$

SVD representation of $\mathbf{H}$:

$$\mathbf{H} = \sum_{n=1}^{R} \sqrt{\mu_n}\,\mathbf{v}_n\mathbf{u}_n^{\dagger} = \mathbf{V}\mathbf{S}\mathbf{U}^{\dagger}\,,$$

$\mathbf{V}$ is $M \times M$ matrix with $\mathbf{v}_m$ as m^{th} column;

$\mathbf{U}$ is $N \times N$ matrix with $\mathbf{u}_n$ as n^{th} column;

$\mathbf{S}$ is $M \times N$ matrix with elements $S_{mn} = \sqrt{\mu_n}\,\delta_{mn}$

Claim:

$$\mathbf{H} = \sum_{n=1}^{R} \sqrt{\mu_n}\, \mathbf{v}_n \mathbf{u}_n^\dagger = \mathbf{V}\mathbf{S}\mathbf{U}^\dagger\,,$$

Check:

$$\left[\mathbf{V}\mathbf{S}\mathbf{U}^\dagger\right]_{mn} = \sum_j \sum_k V_{mj} S_{jk} U_{nk}^*$$

But V_{mj} is row m of column j in $\mathbf{V}$, where j^{th} column is $\mathbf{v}_j$. Therefore

$$V_{mj} = [\mathbf{v}_j]_m \equiv v_{jm} \qquad \text{(tricky)}$$

and similarly for $\mathbf{U}$. Thus

$$\left[\mathbf{V}\mathbf{S}\mathbf{U}^\dagger\right]_{mn} = \sum_j \sum_k v_{jm} \sqrt{\mu_j}\, \delta_{jk} u_{kn}^* = \sum_j v_{jm} \sqrt{\mu_j}\, u_{jn}^*$$

$$= \sum_j \sqrt{\mu_j} \left[\mathbf{v}_j \mathbf{u}_j^\dagger\right]_{mn} = H_{mn} \qquad \text{Q.E.D.}$$

What do we know about $\mathbf{U}$ and $\mathbf{V}$?

Know that set $\{\mathbf{u}_n\}$ is orthonormal and complete:

$$\mathbf{u}_n^\dagger \mathbf{u}_m = \delta_{nm}\,, \qquad \sum_{n=1}^{N} \mathbf{u}_n\, \mathbf{u}_n^\dagger = \mathbf{I}_N$$

These conditions can also be expressed as

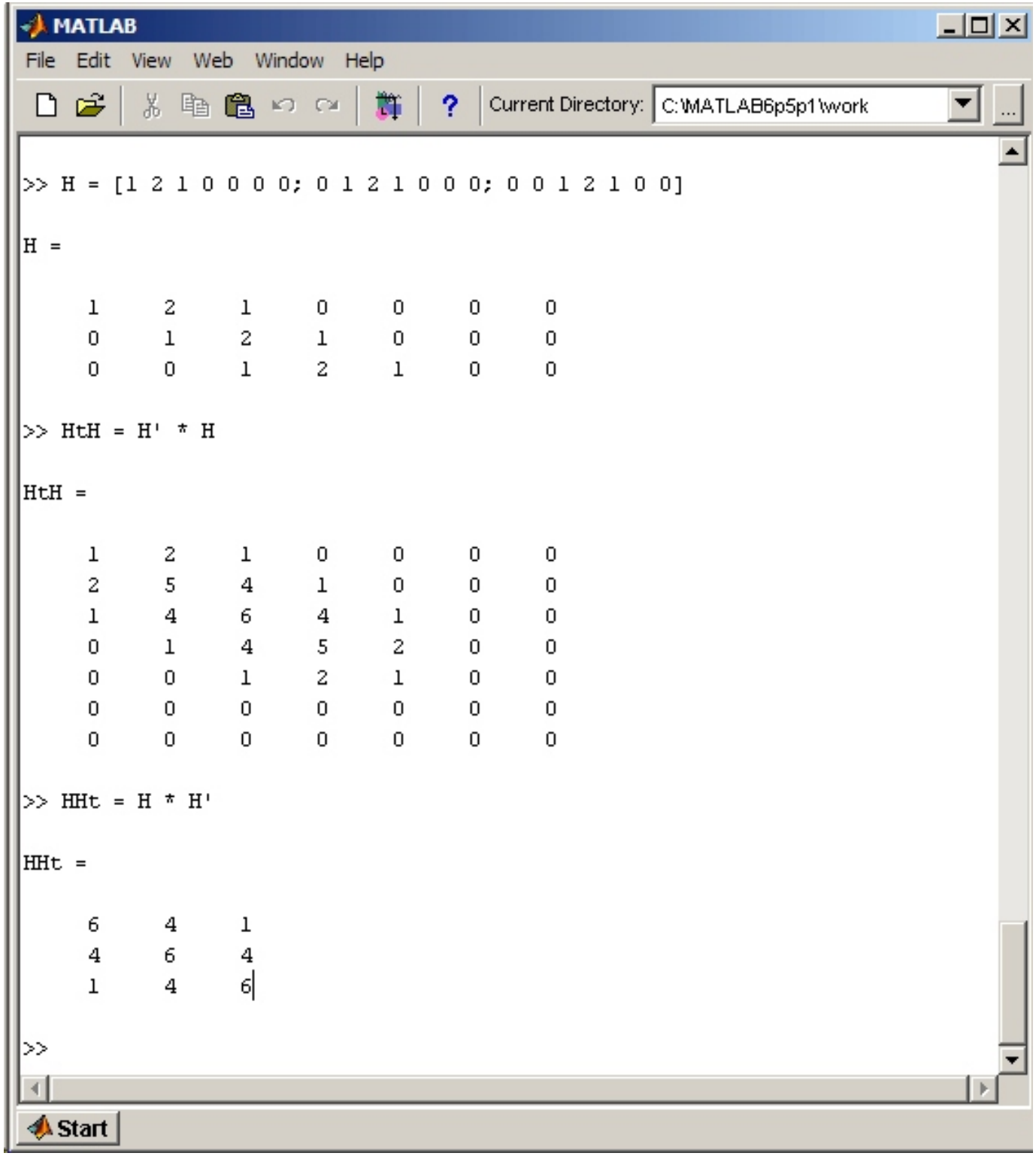
$$\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}_N\,, \qquad \mathbf{U}^\dagger\mathbf{U} = \mathbf{I}_N\,,$$

respectively (sic). Proof left as exercise for the student. Taken together, these two equations imply that $\mathbf{U}$ is an orthogonal matrix:

$$\mathbf{U}^\dagger = \mathbf{U}^{-1}$$

Similarly for $\mathbf{V}$:

$$\mathbf{V}\mathbf{V}^\dagger = \mathbf{I}_M\,, \qquad \mathbf{V}^\dagger\mathbf{V} = \mathbf{I}_M\,, \qquad \mathbf{V}^\dagger = \mathbf{V}^{-1}\,,$$


MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB6p5p1\work
>> H = [1 2 1 0 0 0 0; 0 1 2 1 0 0 0; 0 0 1 2 1 0 0]
H =
1 2 1 0 0 0 0
0 1 2 1 0 0 0
0 0 1 2 1 0 0
>> HtH = H' * H
HtH =
1 2 1 0 0 0 0
2 5 4 1 0 0 0
1 4 6 4 1 0 0
0 1 4 5 2 0 0
0 0 1 2 1 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
>> HHt = H * H'
HHt =
6 4 1
4 6 4
1 4 6
>>
Start

Current Directory: C:\MATLAB6p5p1\work

```
>> % The Matlab SVD command is usually [U, S, V] = svd(H).  We will use a notation that
>> % reflects what we have used in the lecture and also distinguishes H, HtH and HHt :
>> [VH, SH, UH] = svd(H)

VH =

   -0.5215    0.7071    0.4775
   -0.6753   -0.0000   -0.7376
   -0.5215   -0.7071    0.4775


SH =

    3.4898         0         0         0         0         0         0
         0    2.2361         0         0         0         0         0
         0         0    0.9061         0         0         0         0


UH =

   -0.1494    0.3162    0.5269    0.7655    0.1184         0         0
   -0.4924    0.6325    0.2399   -0.4824   -0.2593         0         0
   -0.6859    0.0000   -0.5741    0.1994    0.4003         0         0
   -0.4924   -0.6325    0.2399    0.0837   -0.5413         0         0
   -0.1494   -0.3162    0.5269   -0.3668    0.6823         0         0
         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0    1.0000

>>
```

Start

Current Directory: C:\MATLAB6p5p1\work

```
>> % Does the decomposition work?
>> VH * SH * UH'

ans =

    1.0000    2.0000    1.0000    0.0000   -0.0000         0         0
    0.0000    1.0000    2.0000    1.0000    0.0000         0         0
   -0.0000    0.0000    1.0000    2.0000    1.0000         0         0

>> %Are the columns of UH orthogonal
>> uH1 = UH(:,1)

uH1 =

   -0.1494
   -0.4924
   -0.6859
   -0.4924
   -0.1494
         0
         0

>> uH3 = UH(:,3);
>> uH1' * uH3

ans =

 -9.7145e-017
```

Start

Current Directory: C:\MATLAB6p5p1\work

```
>> % Now look at the SVD of HtH
>> [VHtH, SHtH, UHtH] = svd(HtH)

VHtH =

   -0.1494   -0.3162    0.5269   -0.0368   -0.7737         0         0
   -0.4924   -0.6325    0.2399   -0.1578    0.5245         0         0
   -0.6859    0.0000   -0.5741    0.3525   -0.2753         0         0
   -0.4924    0.6325    0.2399   -0.5471    0.0260         0         0
   -0.1494    0.3162    0.5269    0.7417    0.2232         0         0
         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0    1.0000


SHtH =

   12.1789         0         0         0         0         0         0
         0    5.0000         0         0         0         0         0
         0         0    0.8211         0         0         0         0
         0         0         0    0.0000         0         0         0
         0         0         0         0    0.0000         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0


UHtH =

   -0.1494   -0.3162    0.5269    0.7667    0.1102         0         0
   -0.4924   -0.6325    0.2399   -0.4852   -0.2542         0         0
   -0.6859    0.0000   -0.5741    0.2036    0.3982         0         0
   -0.4924    0.6325    0.2399    0.0780   -0.5421         0         0
   -0.1494    0.3162    0.5269   -0.3595    0.6861         0         0
         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0    1.0000
```

Start

Current Directory: C:\MATLAB6p5p1\work

```
>> % Note that VHtH = UHtH; columns are eigenvectors (not just singular vectors) of HtH:
>> HtH * UHtH(:,1)

ans =

   -1.8201
   -5.9968
   -8.3533
   -5.9968
   -1.8201
         0
         0

>> HtH * UHtH(:,1)/SHtH(1,1)

ans =

   -0.1494
   -0.4924
   -0.6859
   -0.4924
   -0.1494
         0
         0

>>
```

Start

Current Directory: C:\MATLAB6p5p1\work

```
>> % Now do the SVD of HHt.  (Note the switch in notation between U and V)
>> [UHHt, SHHt, VHHt] = svd(HHt)

UHHt =

   -0.5215    0.7071    0.4775
   -0.6753   -0.0000   -0.7376
   -0.5215   -0.7071    0.4775


SHHt =

   12.1789         0         0
         0    5.0000         0
         0         0    0.8211


VHHt =

   -0.5215    0.7071    0.4775
   -0.6753    0.0000   -0.7376
   -0.5215   -0.7071    0.4775

>>
>> % Same eigenvalues as for HtH, and again UHHt = VHHt
```

Start

Current Directory: C:\MATLAB6p5p1\work

```
>> % Suppose we have solved the eigenvalue problem for the smaller matrix HHt;
>> % can we get the solutions for HtH, which is 7X7 in this example?
>> H' * VHHt(:,1)

ans =

   -0.5215
   -1.7184
   -2.3936
   -1.7184
   -0.5215
         0
         0

>> H' * VHHt(:,1)/sqrt(SHHt(1,1))

ans =

   -0.1494
   -0.4924
   -0.6859
   -0.4924
   -0.1494
         0
         0

>>
```

Start

SVD of a CD operator via matrices

Recall:

$$[\boldsymbol{\mathcal{H}}\mathbf{f}]_m = \int_\infty d^q r \; h_m(\mathbf{r})\, f(\mathbf{r})\,,$$

$$\left[\boldsymbol{\mathcal{H}}^\dagger \mathbf{g}\right](\mathbf{r}) = \sum_{m=1}^{M} h_m^*(\mathbf{r})\, g_m\,,$$

from which it follows that

$$[\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger]_{mm'} = \int d^2 r\, h_m(\mathbf{r})\, h_{m'}^*(\mathbf{r})$$

These matrix elements can be evaluated by analytical or numerical integration. Often symmetries are helpful. Suppose, for example, that all of the sensitivity functions are identical except for a shift:

$$h_m(\mathbf{r}) = h(\mathbf{r} - \mathbf{r}_m)\,,$$

where $\mathbf{r}_m$ is the center of the m^{th} sensitivity function, and that there is only nearest-neighbor overlap. Then only a small number of integrals must be computed to completely characterize $\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger$.

SVD of a CD operator via matrices – cont.

The eigenvalue problem for the $M \times M$ matrix $\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger$ can be solved via standard software packages (Matlab, NAG, ...) for M up to about 10^4. On a Beowulf cluster 10^5 is probably feasible.

Once that problem is solved, the eigenvectors for $\boldsymbol{\mathcal{H}}^\dagger\boldsymbol{\mathcal{H}}$ are obtained from

$$\mathbf{u}_n = \frac{1}{\sqrt{\mu_n}} \boldsymbol{\mathcal{H}}^\dagger \mathbf{v}_n \,,$$

or

$$u_n(\mathbf{r}) = \frac{1}{\sqrt{\mu_n}} \sum_{m=1}^{M} v_{nm}\, h_m^*(\mathbf{r}) \,, \qquad (n \leq R) \,,$$

and the SVD is complete.

Example: Mirror modes

Let $h_m(\mathbf{r})$ be the response of a deformable mirror to the m^{th} actuator. The phase pattern produced when activator m is driven by signal g_m is

$$\phi(\mathbf{r}) = \sum_{m=1}^{M} g_m \, h_m(\mathbf{r}) = \left[\boldsymbol{\mathcal{H}}^{\dagger}\mathbf{g}\right](\mathbf{r})$$

As above, solve the eigenvalue problem for $\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^{\dagger}$, obtaining a set of $M \times 1$ eigenvectors $\{\mathbf{v}_n\}$, and then form

$$u_n(\mathbf{r}) = \frac{1}{\sqrt{\mu_n}} \sum_{m=1}^{M} v_{nm} \, h_m^*(\mathbf{r}) \, .$$

These functions are the orthonormal mirror modes. Any phase pattern that can be realized with any combination of actuator signals can be written as

$$\phi(\mathbf{r}) = \sum_{m=1}^{M} \alpha_m \, u_m(\mathbf{r}) \, .$$

Mirror modes: comments

No artificial discretization of the mirror response is needed if the CD viewpoint is adopted; the mirror responses $h_m(\mathbf{r})$ can be sampled arbitrarily finely or even given analytically, say as solutions to the differential equation for deformations of a thin plate.

We have implicitly assumed that the rank R is equal to the number of actuators M; this assumption is valid if none of the M eigenvalues of $\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger$ are zero.

If we want to include mirror deformations that cannot be produced by signals on actuators (e.g., vibrations), then we must write

$$\phi(\mathbf{r}) = \sum_{m=1}^{M} \alpha_m \, u_m(\mathbf{r}) + \sum_{m=M+1}^{\infty} \alpha_m \, u_m(\mathbf{r}) \,,$$

where $u_m(\mathbf{r})$ for $m > R$ is a null function of $\boldsymbol{\mathcal{H}}^\dagger\boldsymbol{\mathcal{H}}$.

Linear shift-invariant systems

Ref.: Barrett and Myers, Chap. 7

Consider a CC imaging system mapping a 2D function $f(\mathbf{r})$ to another 2D function $g(\mathbf{r}_d)$. The system is linear and shift invariant (LSIV) if

$$g(\mathbf{r}_d) = \int_\infty d^q r \; h(\mathbf{r}_d - \mathbf{r})\, f(\mathbf{r})\,. \tag{7.133}$$

By a change of variables, this convolution integral can also be written as

$$. \tag{7.134}$$

In a shorthand form,

$$g(\mathbf{r}_d) = [h * f]\,(\mathbf{r}_d) = h(\mathbf{r}_d) * f(\mathbf{r}_d)\,. \tag{7.135}$$

For 2D convolution operators, range and domain are the same:

$$\mathbb{U} = \mathbb{V} = \mathbb{L}_2(\mathbb{R}^2)$$

LSIV systems – cont.

Since range and domain are the same, convolution operators can have eigenfunctions, not just singular functions. SVD is unnecessary.

In fact the eigenfunctions of an LSIV system are just the complex exponentials (plane waves, Fourier kernels). Let

$$\psi(\mathbf{r}) = \exp(2\pi i \boldsymbol{\rho} \cdot \mathbf{r}) = \exp[2\pi i(x\xi + y\eta)]\,.$$

Then

$$[\boldsymbol{\mathcal{H}}\psi]\,(\mathbf{r}_d) = \int_\infty d^2r\ h(\mathbf{r})\, u(\mathbf{r}_d - \mathbf{r}) = \int_\infty d^2r\ h(\mathbf{r})\ \exp[2\pi i \boldsymbol{\rho} \cdot (\mathbf{r}_d - \mathbf{r})]$$

$$= \exp(2\pi i \boldsymbol{\rho} \cdot \mathbf{r}_d) \int_\infty d^q r\ h(\mathbf{r})\ \exp(-2\pi i \boldsymbol{\rho} \cdot \mathbf{r}) = \lambda \cdot \exp(2\pi i \boldsymbol{\rho} \cdot \mathbf{r}_d)\,,$$

where

$$\lambda = \int_\infty d^2r\ h(\mathbf{r})\ \exp(-2\pi i \boldsymbol{\rho} \cdot \mathbf{r}) = H(\boldsymbol{\rho})$$

LSIV systems – cont.

The complex exponentials $\exp(2\pi i \boldsymbol{\rho} \cdot \mathbf{r})$ are eigenfunctions of *any* LSIV system for *any* real 2D vector $\boldsymbol{\rho}$. [Exercise: Why aren't complex values allowed?]

Let's add a subscript and write a particular eigenfunction as

$$\psi_{\boldsymbol{\rho}_0}(\mathbf{r}) = \exp(2\pi i \boldsymbol{\rho}_0 \cdot \mathbf{r})\,. \tag{7.143}$$

The corresponding eigenvalue is

$$\lambda_{\boldsymbol{\rho}_0} = H(\boldsymbol{\rho}_0)\,. \tag{7.144}$$

SVD of LSIV systems

Even though SVD is not necessary for LSIV systems, it is of course possible. Adjoint of convolution is correlation, and from this observation it can be shown that

$$\left[\boldsymbol{\mathcal{H}}^{\dagger}\boldsymbol{\mathcal{H}}\boldsymbol{\psi}_{\boldsymbol{\rho}}\right](\mathbf{r}) = |\lambda_{\boldsymbol{\rho}}|^2\psi_{\boldsymbol{\rho}}(\mathbf{r}) = |H(\boldsymbol{\rho})|^2\psi_{\boldsymbol{\rho}}(\mathbf{r})\,, \tag{7.147}$$

As required, eigenvalue is now real and non-negative.

Comparison of usual SVD equations and familiar Fourier results:

$$f(\mathbf{r}) = \int_{\infty} d^q\rho\; F(\boldsymbol{\rho})\,\mathrm{exp}(2\pi i\boldsymbol{\rho}\cdot\mathbf{r}) \qquad \mathbf{f} = \sum_{n=0}^{\infty}\alpha_n\mathbf{u}_n$$

$$g(\mathbf{r}_d) = \int_{\infty} d^q\rho\; G(\boldsymbol{\rho})\,\mathrm{exp}(2\pi i\boldsymbol{\rho}\cdot\mathbf{r}_d) \qquad \mathbf{g} = \sum_{n=0}^{\infty}\beta_n\mathbf{v}_n$$

$$g(\mathbf{r}_d) = [h * f](\mathbf{r}_d) \qquad \mathbf{g} = \boldsymbol{\mathcal{H}}\mathbf{f}$$

$$G(\boldsymbol{\rho}) = H(\boldsymbol{\rho})\,F(\boldsymbol{\rho}) \qquad \beta_n = \sqrt{\mu_n}\,\alpha_n\,.$$

Main difference: need continuous qD index $\boldsymbol{\rho}$ instead of integer n.

Discrete shift-invariance

Often we try to represent an LSIV system with a matrix, which in 1D might look like either

$$\begin{bmatrix} c & b & a & 0 & 0 & 0 & 0 & 0 & 0 \\ d & c & b & a & 0 & 0 & 0 & 0 & 0 \\ e & d & c & b & a & 0 & 0 & 0 & 0 \\ 0 & e & d & c & b & a & 0 & 0 & 0 \\ 0 & 0 & e & d & c & b & a & 0 & 0 \\ 0 & 0 & 0 & e & d & c & b & a & 0 \\ 0 & 0 & 0 & 0 & e & d & c & b & a \\ 0 & 0 & 0 & 0 & 0 & e & d & c & b \\ 0 & 0 & 0 & 0 & 0 & 0 & e & d & c \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} c & b & a & 0 & 0 & 0 & 0 & e & d \\ d & c & b & a & 0 & 0 & 0 & 0 & e \\ e & d & c & b & a & 0 & 0 & 0 & 0 \\ 0 & e & d & c & b & a & 0 & 0 & 0 \\ 0 & 0 & e & d & c & b & a & 0 & 0 \\ 0 & 0 & 0 & e & d & c & b & a & 0 \\ 0 & 0 & 0 & 0 & e & d & c & b & a \\ a & 0 & 0 & 0 & 0 & e & d & c & b \\ b & a & 0 & 0 & 0 & 0 & e & d & c \end{bmatrix}$$

Left is a *Toeplitz* matrix, right is a *circulant* matrix. For Toeplitz, $H_{mn} = h_{m-n}$ and for circulant, $H_{mn} = h_{[m-n]}$ where the square brackets denote modulo-N arithmetic.

In either case we can proceed with numerical SVD just as with any other matrix, but for the circulant approximation there is a neat trick.

Diagonalization of a circulant matrix

Suppose we want to compute $\mathbf{g} = \mathbf{H}\mathbf{f}$, where $\mathbf{H}$ is $N \times N$. We can insert the unit matrix judiciously and write

$$\mathbf{g} = \mathbf{H}\mathbf{f} = (\mathbf{A}^{-1}\mathbf{A})\,\mathbf{H}\,(\mathbf{A}^{-1}\mathbf{A})\,\mathbf{f}$$

or

$$\mathbf{A}\mathbf{g} = \mathbf{H}\mathbf{f} = [\mathbf{A}\mathbf{H}\mathbf{A}^{-1}]\,(\mathbf{A}\mathbf{f})\ ,$$

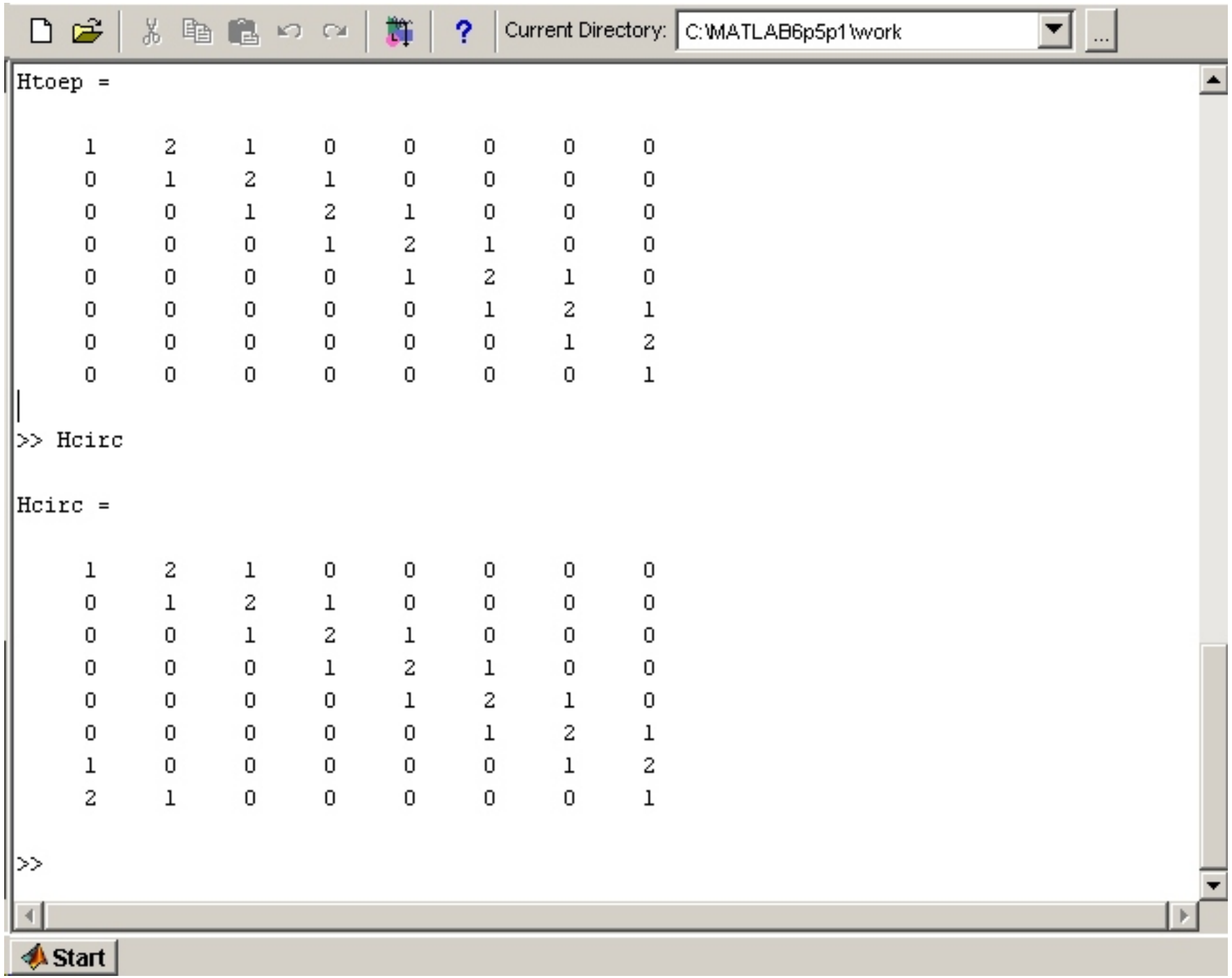
where $\mathbf{A}$ is any invertible $N \times N$ matrix.

If $\mathbf{H}$ is circulant, then $[\mathbf{A}\mathbf{H}\mathbf{A}^{-1}]$ is diagonal when $\mathbf{A}$ is the $N \times N$ matrix operator $\mathbf{W}_N$ that performs the discrete Fourier transform (DFT); the elements of $\mathbf{W}_N$ are $[W_N]_{kn} = \exp(-2\pi i k n/N)$.

Then it follows that

$$[\mathbf{W}_N\mathbf{H}\mathbf{W}_n^{-1}]_{nn'} = H_n\,\delta_{nn'}\ ,$$

where H_n is the ND DFT of one row (or column) of the circulant matrix.


Current Directory: C:\MATLAB6p5p1\work
Htoep =
1 2 1 0 0 0 0 0
0 1 2 1 0 0 0 0
0 0 1 2 1 0 0 0
0 0 0 1 2 1 0 0
0 0 0 0 1 2 1 0
0 0 0 0 0 1 2 1
0 0 0 0 0 0 1 2
0 0 0 0 0 0 0 1
>> Hcirc
Hcirc =
1 2 1 0 0 0 0 0
0 1 2 1 0 0 0 0
0 0 1 2 1 0 0 0
0 0 0 1 2 1 0 0
0 0 0 0 1 2 1 0
0 0 0 0 0 1 2 1
1 0 0 0 0 0 1 2
2 1 0 0 0 0 0 1
>>
Start

Current Directory: C:\MATLAB6p5p1\work

```
>> % Create a unitary DFT matrix with the elements F(n,m) = (1/ sqrt(8))* exp(2 * pi * i * (n-1) * (m-1)/8)
>> real(F * Hcirc * F')

ans =

    4.0000   -0.0000   -0.0000   -0.0000         0   -0.0000   -0.0000    0.0000
   -0.0000    2.4142   -0.0000    0.0000   -0.0000   -0.0000    0.0000    0.0000
   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000
   -0.0000    0.0000    0.0000   -0.4142    0.0000    0.0000    0.0000   -0.0000
         0    0.0000    0.0000    0.0000    0.0000   -0.0000   -0.0000   -0.0000
   -0.0000   -0.0000    0.0000    0.0000   -0.0000   -0.4142    0.0000    0.0000
   -0.0000   -0.0000    0.0000   -0.0000    0.0000   -0.0000   -0.0000   -0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    2.4142

>> imag(F * Hcirc * F')

ans =

         0    0.0000   -0.0000    0.0000   -0.0000   -0.0000   -0.0000    0.0000
    0.0000   -2.4142   -0.0000   -0.0000   -0.0000    0.0000    0.0000   -0.0000
    0.0000    0.0000   -2.0000    0.0000    0.0000   -0.0000   -0.0000   -0.0000
   -0.0000   -0.0000    0.0000   -0.4142    0.0000    0.0000   -0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000   -0.0000    0.0000    0.0000    0.0000
    0.0000   -0.0000   -0.0000         0    0.0000    0.4142    0.0000    0.0000
    0.0000   -0.0000    0.0000    0.0000    0.0000    0.0000    2.0000    0.0000
   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000    0.0000   -0.0000    2.4142

>>
```

Start

Current Directory: C:\MATLAB6p5p1\work

```
>> % What does the DFT transform do to a Toeplitz matrix?
>> real(F * Htoep * F')

ans =

    3.5000   -0.4634   -0.3750   -0.2866   -0.2500   -0.2866   -0.3750   -0.4634
   -0.2652    2.2374   -0.0884   -0.0518   -0.0884   -0.1768   -0.2652   -0.3018
    0.1250    0.2134    0.2500    0.2134    0.1250    0.0366   -0.0000    0.0366
    0.2652    0.3018    0.2652   -0.2374    0.0884    0.0518    0.0884    0.1768
    0.2500    0.2134    0.1250    0.0366    0.0000    0.0366    0.1250    0.2134
    0.2652    0.1768    0.0884    0.0518    0.0884   -0.2374    0.2652    0.3018
    0.1250    0.0366    0.0000    0.0366    0.1250    0.2134    0.2500    0.2134
   -0.2652   -0.3018   -0.2652   -0.1768   -0.0884   -0.0518   -0.0884    2.2374

>> imag(F * Htoep * F')

ans =

         0    0.0884    0.1250    0.0884   -0.0000   -0.0884   -0.1250   -0.0884
    0.3902   -1.9874    0.3902    0.3018    0.2134    0.1768    0.2134    0.3018
    0.3750    0.3384   -1.7500    0.1616    0.1250    0.1616    0.2500    0.3384
    0.1402    0.0518   -0.0366   -0.4874   -0.0366    0.0518    0.1402    0.1768
    0.0000   -0.0884   -0.1250   -0.0884   -0.0000    0.0884    0.1250    0.0884
   -0.1402   -0.1768   -0.1402   -0.0518    0.0366    0.4874    0.0366   -0.0518
   -0.3750   -0.3384   -0.2500   -0.1616   -0.1250   -0.1616    1.7500   -0.3384
   -0.3902   -0.3018   -0.2134   -0.1768   -0.2134   -0.3018   -0.3902    1.9874

>>
```

Start

Comments

It is *not* true that discrete convolutions are converted to products by taking the DFT. Only *cyclic* convolutions behave this way.

Although all circulant matrices are diagonalized by the DFT, there is no corresponding statement for Toeplitz matrices; if you wanna use Toeplitz, you gotta be numerical.

The whole structure of Toeplitz and circulant matrices, DFTs, etc., carries over trivially to higher-dimensional discrete images just by using vector indices.

If $N \to \infty$, then the difference between Toeplitz and circulant matrices disappears, and the DFT limits to the discrete-space Fourier transform (DSFT), discussed in Barrett and Myers Sec. 3.6.4.

Mixed CC/CD systems

So far we have discussed imaging systems in which the output is either a function (CC) or a discrete set of numbers (CD). In many cases, however, the output is a discrete set of functions:

Examples:

- Color imaging (R, G and B images)
- Through-focus microscopy
- Phase diversity in astronomy
- Images with multiple pupil masks
- Gamma-ray images with multiple coded apertures

Formalism for mixed CC/CD systems

Forward operator:

$$g_m(\mathbf{r}_d) = [\boldsymbol{\mathcal{H}}\mathbf{f}]_m(\mathbf{r}_d) = \int_\infty d^q\mathbf{r}\, h_m(\mathbf{r}_d, \mathbf{r})\, f(\mathbf{r})\,.$$

Adjoint operator:

$$\left[\boldsymbol{\mathcal{H}}^\dagger \mathbf{g}\right](\mathbf{r}) = \sum_{m=1}^{M} \int_\infty d^s\mathbf{r}_d\, h_m^*(\mathbf{r}_d, \mathbf{r})\, g_m(\mathbf{r}_d)\,,$$

where M is number of images, not total number of measurements. (Number of measurements is $\infty^2 \times M$.)

Notation: Here $\mathbf{r}$ denotes a general qD vector and $\mathbf{r}_d$ is sD. When we want to be specific to 2D, we shall use $\mathbf{r} = (x, y)$ or $\mathbf{r}_d = (x_d, y_d)$.

Axial systems

Many real-world imaging systems map 3D to 2D ($q = 3, s = 2$) and have an obvious optical axis, which we can call "depth" and denote z. For example, the z axis might be normal to a sample surface or normal to a pupil plane. It might be an axis of symmetry, though we don't require that.

With no loss of generality (yet) we can write the m^{th} image as

$$g_m(\mathbf{r}_d) = \int_\infty d^2r \int_0^\infty dz\, h_m(\mathbf{r}_d, \mathbf{r}, z)\, f(\mathbf{r}, z)\,.$$

If the system is laterally shift-invariant and the object is confined between planes z_1 and z_2, then

$$g_m(\mathbf{r}_d) = \int_\infty d^2r \int_{z_1}^{z_2} dz\, h_m(\mathbf{r}_d - \mathbf{r}, z)\, f(\mathbf{r}, z)\,, \qquad m = 1, ..., M\,.$$

Laterally shift-invariant axial systems

$$\left[\boldsymbol{\mathcal{H}}^{\dagger}\boldsymbol{\mathcal{H}}\mathbf{f}\right](\mathbf{r},z) = \int_{\infty} d^2r' \int_{z_1}^{z_2} dz' f(\mathbf{r}',z') p(\mathbf{r}-\mathbf{r}';z,z')\,,$$

where

$$p(\mathbf{r}-\mathbf{r}';z,z') = \sum_{m=1}^{M} \int_{\infty} d^2r_d\, h_m^*(\mathbf{r}_d-\mathbf{r};z)\, h_m(\mathbf{r}_d-\mathbf{r}';z')\,.$$

$$\left[\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^{\dagger}\mathbf{g}\right]_m(\mathbf{r}_d) = \int_{\infty} d^2r'_d \sum_{m'=1}^{M} g_{m'}(\mathbf{r}_d)\, k_{mm'}(\mathbf{r}_d-\mathbf{r}'_d)$$

$$k_{mm'}(\mathbf{r}_d-\mathbf{r}'_d) = \int_{\infty} d^2r \int_{z_1}^{z_2} dz\, h_m(\mathbf{r}_d-\mathbf{r};z)\, h_m^*(\mathbf{r}'_d-\mathbf{r};z)\,,$$

SVD of laterally shift-invariant axial systems

Ref: Barrett and Myers, Sec. 7.2.10 (covers the case $M = 1$)

Usual procedure: Solve eigenvalue problem for $\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger$, backproject to get eigenvectors of $\boldsymbol{\mathcal{H}}^\dagger\boldsymbol{\mathcal{H}}$, taking advantage of shift-invariance of matrix elements of $\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger$.

After some algebra, it is found that

$$u_{\boldsymbol{\rho},j}(z) = \exp(2\pi i \boldsymbol{\rho}\cdot\mathbf{r}) \sum_{m=1}^{M} \alpha_{jm}\, H_m(-\boldsymbol{\rho}; z)\,,$$

where

$$\sum_{m=1}^{M} K_{nm}(\boldsymbol{\rho})\, \alpha_{jm} = \mu_{\boldsymbol{\rho},j}\, \alpha_{jn}\,,$$

and

$$K_{nm}(\boldsymbol{\rho}) = \int_{z_1}^{z_2} dz'\, H_m(-\boldsymbol{\rho}; z') H_n(\boldsymbol{\rho}; z')\,.$$

Comments

Good news: We just have to solve an $M \times M$ eigenvalue problem. where M may be very small

Bad news: We have to do it for every $\boldsymbol{\rho}$

Open questions:

• Can we sample $\boldsymbol{\rho}$ on a sparse grid, compute the eigenvalues and interpolate?

• Does "prior" knowledge of the eigenvalues speed up the eigenvector calculation?

Summary of SVD methods from this lecture

- For a general $N \times M$ matrix:
  Plug into Matlab
  Solve eigenvalue problem for $\mathbf{H}\mathbf{H}^t$ if $M < N$

- For CD operator:
  Find eigenfunctions of matrix $\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^\dagger$, backproject

- For CC LSIV operator
  Use Fourier integral

- For matrix representations of LSIV systems
  Use circulant approximation, DFT
  For very large image arrays, use DSFT

- For 3D $\rightarrow$ 2D axial LSIV systems
  Look for solutions $\propto \exp(2\pi i \boldsymbol{\rho} \cdot \mathbf{r}) \sum_{m=1}^{N} \alpha_{jm} H_m(-\boldsymbol{\rho}; z)$
  Find coefficients by doing $M \times M$ e'val problem for each $\boldsymbol{\rho}$