特殊矩阵

主 讲 人: 马 丽 艳
办 公 室: 计 1013
Email: liyanma@t.shu.edu.cn
计算机工程与科学学院

主要内容

- 埃尔米特矩阵
- 置换矩阵、互换矩阵、选择矩阵
- 正交矩阵与酉矩阵
- 三角阵
- 范德蒙矩阵、傅里叶矩阵
- 哈达玛矩阵(Hadamard matrix)
- 拓普利兹矩阵(Toeplitz matrix)
- 汉克矩阵(Hankel matrix)

埃尔米特矩阵

- **埃尔米特矩阵(Hermitian matrix)**
- 定义: 矩阵$A = [a_{ij}] \in M_n$ 是Hermitian矩阵，若 $A = A^H$. A是斜（反）Hermitian，若 $A = -A^H$.

$$A = \begin{bmatrix} 1 & 2 & -i \\ 2 & -1 & 1+i \\ i & 1-i & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 1+i & 2-i \\ 1+i & 2i & 4+2i \\ 2-i & 4+2i & 5 \end{bmatrix}$$

$$B^H = \begin{bmatrix} 3 & 1-\bar{i} & 2+i \\ 1-i & -2i & 4-2i \\ 2+i & 4-2i & 5 \end{bmatrix} \neq B$$

埃尔米特矩阵

- **性质:**

1. 若 A 是Hermitian，则A^k 和 A^{-1} 是 Hermitian.
2. $A + A^H$ and AA^H 是Hermitian， $A - A^H$ 是反Hermitian.
3. 任意$A \in M_n$ 可以唯一分解为$A = B + iC = B + D$，其中B, C 是Hermitian，D 是skew-Hermitian. 实际上 $B = (A + A^H)/2$, $D = iC = (A - A^H)/2$.
4. 如果A 是 Hermitian 矩阵，有

- x^HAx 是实数，for all $x \in C_n$
- A 的特征值为实数
- S^HAS 是Hermitian，for all $S \in M_n$

埃尔米特矩阵

对于二次型 $x^H Ax$，其中**A**为**Hermitian**矩阵 $A = A^H$

◆定义

一个复共轭对称矩阵A称为

正定矩阵（A>0）：若二次型 $x^H Ax > 0, \forall x \neq 0;$

半正定（非负定）矩阵（A≥0）：若 $x^H Ax \geq 0, \forall x \neq 0;$

负定矩阵（A<0）：若二次型 $x^H Ax < 0, \forall x \neq 0;$

半负定（非正定）矩阵（A≤0）：若 $x^H Ax \leq 0, \forall x \neq 0;$

不定矩阵：若二次型 $x^H Ax$ 既可能取正值，也可能取负值.

小结：作为一个性能指标，二次型刻画Hermitian矩阵的正定性.

埃尔米特矩阵—实数

◊对称矩阵： $A = A^T$　　反对称矩阵：$A = -A^T$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{pmatrix} \qquad A = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ -a_{12} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ -a_{1n} & -a_{2n} & \cdots & 0 \end{pmatrix}$$

性质：

（1）若A为反对称矩阵，则 A^T 亦为反对称矩阵.

（2）若A为反对称矩阵，则 A^{-1} 亦为反对称矩阵.

埃尔米特矩阵-Matlab

```
>> A = [0-1i 2+1i;4+2i 0-2i]

A =

   0.0000 - 1.0000i   2.0000 + 1.0000i
   4.0000 + 2.0000i   0.0000 - 2.0000i

>> B = A'

B =

   0.0000 + 1.0000i   4.0000 - 2.0000i
   2.0000 - 1.0000i   0.0000 + 2.0000i
```

```
>> A = [2 1; 9 7; 2 8; 3 5]

A =

     2     1
     9     7
     2     8
     3     5

>> B = A'

B =

     2     9     2     3
     1     7     8     5
```

初等矩阵

◊**初等矩阵：**对 $n\times n$ 单位矩阵 I_n 进行初等行(或列)变换得到的矩阵.

Ⅰ型初等矩阵 $E_{(p.q)}$：互换单位矩阵 I_n 的第p行和第q行得到的矩阵，或互换单位矩阵的第p列和第q列得到的矩阵.

$r_p \leftrightarrow r_q$ 或 $c_p \leftrightarrow c_q$

Ⅱ型初等矩阵 $E_{\alpha(p)}$：用一个非零常数α乘以单位矩阵 I_n 的第p行（或列）得到的矩阵.

$r_p \leftarrow \alpha r_p$ 或 $c_p \leftarrow \alpha c_p$

Ⅲ型初等矩阵 $E_{(p)+\alpha(q)}$：用一个非零常数α乘以单位矩阵 I_n 的第q行（或列），再加上 I_n 的第p行（或列）得到的矩阵.

$r_p \leftarrow r_p + \alpha r_p$ 或 $c_p \leftarrow c_p + \alpha c_p$

初等矩阵

初等矩阵左乘：

（1）Ⅰ型初等矩阵左乘矩阵A，即 $E_{(p,q)}A$ ，表示互换矩阵A的第p行和第q行.

（2）Ⅱ型初等矩阵左乘矩阵A，即 $E_{\alpha(p)}A$ ，表示矩阵A的第p行元素乘一个非零常数α.

（3）Ⅲ型初等矩阵左乘矩阵A，即 $E_{(p)+\alpha(q)}A$ ，表示矩阵A的第q行乘以非零常数α后，再与第p行相加.

初等矩阵右乘：

（1）Ⅰ型初等矩阵右乘A，即 $AE_{(p,q)}$ ，表示互换矩阵A的第p列和第q列.

（2）Ⅱ型初等矩阵右乘A，即 $AE_{\alpha(p)}$ ，表示矩阵A的第p列元素乘一个非零常数α.

（3）Ⅲ型初等矩阵右乘A，即 $AE_{(p)+\alpha(q)}$ ，表示矩阵A的第q列乘以非零常 数α后，再与第p列相加.

初等矩阵

R = rref(A) produces the reduced row echelon form of A using Gauss Jordan elimination with partial pivoting. A default tolerance of (max(size(A))*eps *norm(A,inf)) tests for negligible column elements.

初等矩阵

```
>>  A = magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> [R,jb] = rref(A)
R =
     1     0     0     1
     0     1     0     3
     0     0     1    -3
     0     0     0     0
jb =
     1     2     3
```

```
>> A=randi(5,5,3)
A =
     4     1     5
     3     1     4
     3     3     1
     1     1     4
     4     5     3
>> [R,jb] = rref(A)
R =
     1     0     0
     0     1     0
     0     0     1
     0     0     0
     0     0     0
jb =
     1     2     3
```

初等矩阵

```
>> A=randi(5,5,8)
A =
     2     5     4     3     2     3     5     5
     2     4     1     4     4     3     5     1
     5     1     4     4     1     5     3     3
     1     2     1     5     1     4     1     1
     5     2     4     5     4     4     2     5
>> [R,jb] = rref(A)
R =
    1.0000         0         0         0         0    0.8478    1.9130   -1.2174
         0    1.0000         0         0         0    0.4807    2.0048   -0.5990
         0         0    1.0000         0         0   -0.4155   -1.3961    2.1208
         0         0         0    1.0000         0    0.6642   -0.5411    0.0918
         0         0         0         0    1.0000   -0.7150   -0.8213    0.8357
jb =     1     2     3     4     5
```

置换矩阵

- **置换矩阵(permutation matrix)**
- **定义**：一个方阵，每一行和每一列有且仅有一个非零元素1.

$$\begin{bmatrix} 0 & \cdots & 1 & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 1 & \cdots & 0 \end{bmatrix}$$

- 多个对调单位阵的两行（或列）的初等矩阵的乘积。
- 左乘行重排，右乘列重排。

置换矩阵

- **性质：**
- 置换矩阵的转置为另一个置换矩阵；
- 置换矩阵 P 为正交矩阵，满足 $P^T P = PP^T = I$ 。
- 置换矩阵的转置等于原矩阵的逆矩阵，$P^T = P^{-1}$。

置换矩阵-Matlab

- **p = randperm(n) 返回行向量，其中包含从 1 到 n（包括二者）之间的整数随机置换。**
- **p = randperm(n,k) 返回行向量，其中包含在 1 到 n（包括二者）之间随机选择的 k 个唯一整数。**

```
>> randperm(6)

ans =

     4     5     2     3     6     1

>> randperm(6)

ans =

     4     3     1     5     6     2
```

```
>> randperm(6,3)

ans =

     6     2     4
```

置换矩阵-Matlab

- **B = permute(A,dimorder) 按照向量 dimorder 指定的顺序重新排列数组的维度。**
- **P = perms(v) 返回的矩阵包含了向量 v 中元素按字典顺序反序的所有排列。**

```
A =

    0.8147    0.9134    0.2785    0.9649
    0.9058    0.6324    0.5469    0.1576
    0.1270    0.0975    0.9575    0.9706

>> B = permute(A,[2 1])

B =

    0.8147    0.9058    0.1270
    0.9134    0.6324    0.0975
    0.2785    0.5469    0.9575
    0.9649    0.1576    0.9706
```

```
>> v = [2 4 6];
P = perms(v)

P =

     6     4     2
     6     2     4
     4     6     2
     4     2     6
     2     6     4
     2     4     6
```

置换矩阵-Matlab

- B = permute(A,dimorder) 按照向量 dimorder 指定的顺序重新排列数组的维度。

```
>>  A = rand(2,3,4,5);
>> size(A')
错误使用  '
未定义 N 维数组的转置。请改用 PERMUTE。

>>  size(permute(A,[2 1 4 3]))

ans =

     3     2     5     4
```

互换矩阵

- **互换矩阵(exchange matrix)**
- **定义**：交叉对角线上具有元素1，而所有其他元素为0的置换矩阵

$$
\begin{matrix}
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0
\end{matrix}
$$

- 又称为反射矩阵，后向单位矩阵。
- 左乘行顺序反转，右乘列顺序反转。

互换矩阵

- B = flip(A) returns array B the same size as A, but with the order of the elements reversed.
- B = flipud(A) returns A with its rows flipped in the up-down direction (that is, about a horizontal axis).
- B = fliplr(A) returns A with its columns flipped in the left-right direction (that is, about a vertical axis).
- **flipud(A) is equivalent to flip(A,1).**
- **fliplr(A) is equivalent to flip(A,2).**

互换矩阵

```
>> A = rand(2,3)

A =

    0.6596    0.9730    0.8003
    0.5186    0.6490    0.4538

>> flipud(A)

ans =

    0.5186    0.6490    0.4538
    0.6596    0.9730    0.8003
```

```
>> fliplr(A)

ans =

    0.8003    0.9730    0.6596
    0.4538    0.6490    0.5186
```

互换矩阵

```
>> A= rand(2,3,4)
A(:,:,1) =
    0.7060    0.2769    0.0971
    0.0318    0.0462    0.8235

A(:,:,2) =
    0.6948    0.9502    0.4387
    0.3171    0.0344    0.3816

A(:,:,3) =
    0.7655    0.1869    0.4456
    0.7952    0.4898    0.6463

A(:,:,4) =
    0.7094    0.2760    0.6551
    0.7547    0.6797    0.1626
```

```
>> flip(A,1)
ans(:,:,1) =
    0.0318    0.0462    0.8235
    0.7060    0.2769    0.0971

ans(:,:,2) =
    0.3171    0.0344    0.3816
    0.6948    0.9502    0.4387

ans(:,:,3) =
    0.7952    0.4898    0.6463
    0.7655    0.1869    0.4456

ans(:,:,4) =
    0.7547    0.6797    0.1626
    0.7094    0.2760    0.6551
```

互换矩阵

```
>> A= rand(2,3,4)
A(:,:,1) =
    0.7060    0.2769    0.0971
    0.0318    0.0462    0.8235

A(:,:,2) =
    0.6948    0.9502    0.4387
    0.3171    0.0344    0.3816

A(:,:,3) =
    0.7655    0.1869    0.4456
    0.7952    0.4898    0.6463

A(:,:,4) =
    0.7094    0.2760    0.6551
    0.7547    0.6797    0.1626
```

```
>> flip(A,2)
ans(:,:,1) =
    0.0971    0.2769    0.7060
    0.8235    0.0462    0.0318

ans(:,:,2) =
    0.4387    0.9502    0.6948
    0.3816    0.0344    0.3171

ans(:,:,3) =
    0.4456    0.1869    0.7655
    0.6463    0.4898    0.7952

ans(:,:,4) =
    0.6551    0.2760    0.7094
    0.1626    0.6797    0.7547
```

互换矩阵

```
>> A= rand(2,3,4)
A(:,:,1) =
    0.7060    0.2769    0.0971
    0.0318    0.0462    0.8235

A(:,:,2) =
    0.6948    0.9502    0.4387
    0.3171    0.0344    0.3816

A(:,:,3) =
    0.7655    0.1869    0.4456
    0.7952    0.4898    0.6463

A(:,:,4) =
    0.7094    0.2760    0.6551
    0.7547    0.6797    0.1626
```

```
>> flip(A,3)
ans(:,:,1) =
    0.7094    0.2760    0.6551
    0.7547    0.6797    0.1626

ans(:,:,2) =
    0.7655    0.1869    0.4456
    0.7952    0.4898    0.6463

ans(:,:,3) =
    0.6948    0.9502    0.4387
    0.3171    0.0344    0.3816

ans(:,:,4) =
    0.7060    0.2769    0.0971
    0.0318    0.0462    0.8235
```

移位矩阵

- **移位矩阵(shift matrix)**

$$\begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{matrix}$$

移位矩阵

```
>> A = rand(5,5)

A =

    0.5479    0.7011    0.1781    0.5612    0.4607
    0.9427    0.6663    0.1280    0.8819    0.9816
    0.4177    0.5391    0.9991    0.6692    0.1564
    0.9831    0.6981    0.1711    0.1904    0.8555
    0.3015    0.6665    0.0326    0.3689    0.6448

>> Y*A

ans =

    0.9427    0.6663    0.1280    0.8819    0.9816
    0.4177    0.5391    0.9991    0.6692    0.1564
    0.9831    0.6981    0.1711    0.1904    0.8555
    0.3015    0.6665    0.0326    0.3689    0.6448
    0.5479    0.7011    0.1781    0.5612    0.4607
```

```
>> A*Y

ans =

    0.4607    0.5479    0.7011    0.1781    0.5612
    0.9816    0.9427    0.6663    0.1280    0.8819
    0.1564    0.4177    0.5391    0.9991    0.6692
    0.8555    0.9831    0.6981    0.1711    0.1904
    0.6448    0.3015    0.6665    0.0326    0.3689
```

移位矩阵

- **Y = circshift(A,K,dim) circularly shifts the values in array A by K positions along dimension dim.**

```
A =

     1     1     0     0
     1     1     0     0
     0     0     0     0
     0     0     0     0

>> Y = circshift(A,1,2)

Y =

     0     1     1     0
     0     1     1     0
     0     0     0     0
     0     0     0     0
```

```
>> Y = circshift(A,[1,2])

Y =

     0     0     0     0
     0     0     1     1
     0     0     1     1
     0     0     0     0
```

移位矩阵

- **Y = circshift(A,K,dim) circularly shifts the values in array A by K positions along dimension dim.**

```
>> a= randi(6,6)

a =

     5     5     5     1     5     4
     2     5     5     3     2     1
     6     2     2     6     4     1
     1     3     5     3     5     2
     3     3     4     4     6     6
     3     4     1     2     6     2
```

```
>> circshift(a,[1,2])

ans =

     6     2     3     4     1     2
     5     4     5     5     5     1
     2     1     2     5     5     3
     4     1     6     2     2     6
     5     2     1     3     5     3
     6     6     3     3     4     4
```

广义置换矩阵

- 广义置换矩阵(generalized permutation matrix)
- 定义：一个置换矩阵和一个非奇异对角矩阵的乘积。

```
P =

     0     0     0     0     1
     0     0     1     0     0
     0     1     0     0     0
     0     0     0     1     0
     1     0     0     0     0

>> A=diag([2,3,4,5,6]);
```

```
>> P*A

ans =

     0     0     0     0     6
     0     0     4     0     0
     0     3     0     0     0
     0     0     0     5     0
     2     0     0     0     0
```

选择矩阵

- **选择矩阵(selective matrix)**
- **定义：可以选择给定矩阵的某些行或者列。**

$$J_1 = [I \quad 0], J_2 = [0 \quad I] \qquad J_1 = \begin{bmatrix} I \\ 0 \end{bmatrix}, J_2 = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

```
>> A = rand(3,6)
A =
    0.4868    0.3063    0.8176    0.3786    0.3507    0.5502
    0.4359    0.5085    0.7948    0.8116    0.9390    0.6225
    0.4468    0.5108    0.6443    0.5328    0.8759    0.5870
>> A(1:2,:)
ans =
    0.4868    0.3063    0.8176    0.3786    0.3507    0.5502
    0.4359    0.5085    0.7948    0.8116    0.9390    0.6225
```

```
>> A(:,[1,3:4])
ans =
    0.4868    0.8176    0.3786
    0.4359    0.7948    0.8116
    0.4468    0.6443    0.5328

>> A(:,[1,3,5])
ans =
    0.4868    0.8176    0.3507
    0.4359    0.7948    0.9390
    0.4468    0.6443    0.8759
```

正交矩阵与酉矩阵

正交矩阵与酉矩阵(orthogonal/unitary matrix)

◊基本矩阵：

$$E_{ij}^{(m\times n)} = e_i^{(m)}(e_j^{(n)})^T$$

性质：

$$(1)\ E_{ij}^{(m\times n)} E_{kl}^{(n\times r)} = \delta_{jk} E_{il}^{(m\times r)}$$

$$(2)\ \left(E_{ij}^{(m\times n)}\right)^T = E_{ji}^{(n\times m)}$$

$$(3)\ A = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} E_{ij}^{(m\times n)}$$

$$(4)\ E_{ij}^{(s\times m)} A E_{kl}^{(n\times r)} = a_{jk} E_{il}^{(s\times r)}$$

$$(5)\ \det(E_{ij}^{(m\times n)}) = 0, (m = n > 1)$$

```
>> a=[0 1 0 0 0]
a =
     0     1     0     0     0

>> b=[0 0 0 0 1 0 0]'
b =
     0
     0
     0
     0
     1
     0
     0
```

```
>> b*a
ans =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     1     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

正交矩阵与酉矩阵

◊**正交矩阵(实数域, orthogonal matrix)** $QQ^T = Q^T Q = I$

半正交(semi-orthogonal)矩阵: $QQ^T = I_m$ 或 $Q^T Q = I_n$

◊**酉矩阵(复数域, unitary matrix)** $UU^H = U^H U = I$

仿酉(para-unitary)矩阵: $UU^H = I_m$ 或 $U^H U = I_n$

性质:

（1） U 为酉矩阵 $\Leftrightarrow U^{-1} = U^H$

（2） $U \in R^{m\times m}$ 为酉矩阵 $\Leftrightarrow$ U为正交矩阵

（3） U为酉矩阵 $\Leftrightarrow$ U的列(行)是标准正交的向量.

正交矩阵与酉矩阵

◊正交矩阵(实数域, orthogonal matrix)

判断Q是否为正交矩阵？

$$Q = \begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3} \\ 1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3} \\ 0 & 2/\sqrt{6} & 1/\sqrt{3} \end{bmatrix}$$

```
>> H=hadamard(4)*1i/2
H =
   0.0000 + 0.5000i   0.0000 + 0.5000i   0.0000 + 0.5000i   0.0000 + 0.5000i
   0.0000 + 0.5000i   0.0000 - 0.5000i   0.0000 + 0.5000i   0.0000 - 0.5000i
   0.0000 + 0.5000i   0.0000 + 0.5000i   0.0000 - 0.5000i   0.0000 - 0.5000i
   0.0000 + 0.5000i   0.0000 - 0.5000i   0.0000 - 0.5000i   0.0000 + 0.5000i
>> H*H'
ans =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1

>> H'*H
ans =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
>> H=hadamard(8)
H =
     1     1     1     1     1     1     1     1
     1    -1     1    -1     1    -1     1    -1
     1     1    -1    -1     1     1    -1    -1
     1    -1    -1     1     1    -1    -1     1
     1     1     1     1    -1    -1    -1    -1
     1    -1     1    -1    -1     1    -1     1
     1     1    -1    -1    -1    -1     1     1
     1    -1    -1     1    -1     1     1    -1
```

正交矩阵与酉矩阵

（4） $U_{m\times m}$ 为酉矩阵，则 $U^T, U^H, U^*, U^{-1}, U^i$ 均为酉矩阵.

（5）U和V均为酉矩阵，$\Rightarrow UV$ 为酉矩阵

（6）若 $U_{m\times m}$ 为酉矩阵，则

① $|\det(U)| = 1$

② $rank(U) = m$

③ $UU^H = U^H U$

④ λ 为U的特征值 $\Rightarrow |\lambda| = 1$

⑤ $x_{m\times 1} \Rightarrow \|Ux\|_2 = \|x\|_2$

⑥ $A_{m\times n} \Rightarrow \|UA\|_F = \|A\|_F$ （等距变换）

⑦ $A_{n\times m} = \|AU\|_F = \|A\|_F$

正交矩阵与酉矩阵

- 若A是酉矩阵，则 $|\det A| = 1$，或 $\det A\,\overline{\det A} = \det A \det \overline{A} = 1$

证明：

$$1=\det I=\det(A^H A)=\det A^H \det A=\det(\overline{A})^T \det A$$

$$=\det\overline{A}\det A=\overline{\det A}\det A=|\det A|^2 \Longrightarrow |\det A|=1$$

$$\det(\overline{A})=\overline{\det(A)}$$

归纳法

$$\det\overline{A_k}=\sum_{j=1}^{k} \bar{a}_{1j}(-1)^{1+j}\left|\overline{A_{k-1}}\right|_{1j}=\sum_{j=1}^{k}\overline{a_{1j}}\,\overline{(-1)^{1+j}}\,\overline{|A_{k-1}|_{1j}}$$

$$=\sum_{j=1}^{k}\overline{a_{1j}(-1)^{1+j}|A_{k-1}|_{1j}}=\overline{\sum_{j=1}^{k} a_{1j}(-1)^{1+j}|A_{k-1}|_{1j}}$$

$$=\overline{\det A_k}$$

正交矩阵与酉矩阵

表1. 实向量、实矩阵与复向量、复矩阵的性质比较

实向量、实矩阵	复向量、复矩阵
范数：$\|x\| = \sqrt{x_1^2 + \cdots + x_n^2}$	范数：$\|x\| = \sqrt{\lvert x_1\rvert^2 + \cdots + \lvert x_n\rvert^2}$
转置：$A^T = [a_{ij}], (AB)^T = B^T A^T$	转置：$A^T = [a_{ji}^*], (AB)^H = B^H A^H$
内积：$\langle x, y\rangle = x^T y$	内积：$\langle x, y\rangle = x^H y$
正交性：$x^T y = 0$	正交性：$x^H y = 0$
对称矩阵：$A^T = A$	对称矩阵：$A^H = A$
正交矩阵：$Q^T = Q^{-1}$	正交矩阵：$U^H = U^{-1}$
特征值分解：$A = Q\Sigma Q^T = Q\Sigma Q^{-1}$	特征值分解：$A = U\Sigma U^H = U\Sigma U^{-1}$
范数的正交不变性：$\|Qx\| = \|x\|$	范数的正交不变性：$\|Ux\| = \|x\|$
内积的正交不变性：$\langle Qx, Qy\rangle = \langle x, y\rangle$	内积的正交不变性：$\langle Ux, Uy\rangle = \langle x, y\rangle$

正交矩阵与酉矩阵

酉变换： 若U为酉矩阵，则称线性变换 Ux 称为x的酉变换.

酉等价： 若U为酉矩阵，则称矩阵 $B=U^{H}AU$ 与A酉等价.

特别地，如果U取实数（因而是实正交的），则称B与A正交等价.

◊**正规矩阵（normal matrix）：** 满足 $A^{H}A=AA^{H}$ 的矩阵 $A\in C^{n\times n}$.

正交矩阵与酉矩阵

```
A =

     1     0     1

    -1    -2     0

     0     1    -1

r = rank(A)  =   3
```

```
>> Q = orth(A)

Q =

   -0.1200   -0.8097    0.5744

    0.9018    0.1531    0.4042

   -0.4153    0.5665    0.7118
```

```
>> A = A + i*1
1.0000 + 1.0000i   0.0000 + 1.0000i   1.0000 + 1.0000i
  -1.0000 + 1.0000i  -2.0000 + 1.0000i   0.0000 + 1.0000i
   0.0000 + 1.0000i   1.0000 + 1.0000i  -1.0000 + 1.0000i
>> Q = orth(A)
-0.3087 - 0.4630i  -0.4044 + 0.0215i  -0.7249 + 0.0270i
   0.4801 - 0.5402i  -0.3588 + 0.2344i   0.3628 + 0.4030i
  -0.0942 - 0.3989i   0.5503 - 0.5911i  -0.0138 + 0.4237i
```

三角阵

三角阵(upper/lower triangular matrix)

◊对角矩阵与反(斜)对角矩阵

$$D=\begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix}=diag(d_1,d_2,\ldots,d_n) \qquad \overline{D}=\begin{bmatrix} & & & d_1 \\ & & d_2 & \\ & \iddots & & \\ d_n & & & \end{bmatrix}$$

◊上三角矩阵与下三角矩阵

$$R=\begin{bmatrix} * & * & \cdots & * \\ & * & \cdots & * \\ & & \ddots & \vdots \\ & & & * \end{bmatrix} \qquad L=\begin{bmatrix} * & & & \\ * & * & \ddots & \\ \vdots & \vdots & \ddots & \\ * & * & \cdots & * \end{bmatrix}$$

*当对角元素为1时，称为**单位上(下)三角矩阵**.

*当对角元素为0时，称为严格**上(下)三角矩阵**.

三角阵

上(下)三角矩阵的性质

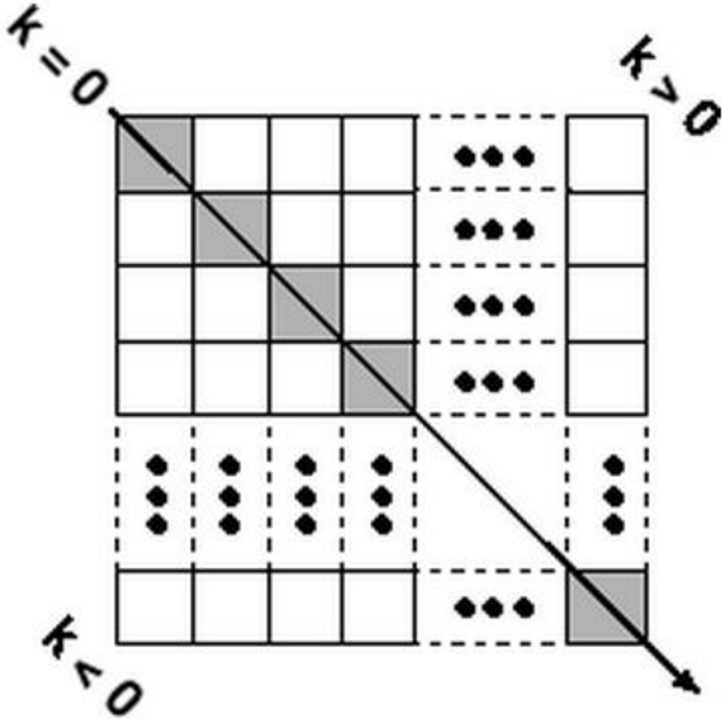
- 上（下）三角矩阵的和、差、乘积仍为上（下）三角矩阵的.
- 上（下）三角矩阵的k次幂仍为上（下）三角矩阵的，且其第i个对角线元素为 $r_{ii}^k(l_{ii}^k)$
- 上（下）三角矩阵的转置为下（上）三角矩阵.
- 上（下）三角矩阵的的逆仍为上（下）三角矩阵的.
- 上（下）三角矩阵的行列式等于其对角线元素之积，即

$$\det(R) = r_{11}r_{22}\ldots r_{nn} = \prod_{i=1}^{n} r_{ii}$$

- 上（下）三角矩阵的特征值等于其各对角线元素.
- 若矩阵 $A_{n\times n} > 0$ (正定)，则A可以分解为一个下三角矩阵与其复共轭转置之积 $A = LL^H$ （Cholesky分解)

三角阵

- **U = triu(X) returns the upper triangular part of X.**
- **U = triu(X,k) returns the element on and above the kth diagonal of X. k = 0 is the main diagonal, k > 0 is above the main diagonal, and k < 0 is below the main diagonal.**
- **L = tril(X) returns the lower triangular part of X.**
- **L = tril(X,k) returns the elements on and below the kth diagonal of X. k = 0 is the main diagonal, k > 0 is above the main diagonal, and k < 0 is below the main diagonal.**



三角阵

```
>> A = randi(10,5,5)
A =
     4     3     1     2     2
     9     8     1     6     8
     6     8     6     5     4
     6     4     8     1     6
    10     6    10     4     2
>> triu(A)
ans =
     4     3     1     2     2
     0     8     1     6     8
     0     0     6     5     4
     0     0     0     1     6
     0     0     0     0     2
```

```
>> triu(A,2)
ans =
     0     0     1     2     2
     0     0     0     6     8
     0     0     0     0     4
     0     0     0     0     0
     0     0     0     0     0
>> triu(A,-2)
ans =
     4     3     1     2     2
     9     8     1     6     8
     6     8     6     5     4
     0     4     8     1     6
     0     0    10     4     2
```

三角阵

```
>> diag(A)
ans =
     4
     8
     6
     1
     2
>> tril(A)
ans =
     4     0     0     0     0
     9     8     0     0     0
     6     8     6     0     0
     6     4     8     1     0
    10     6    10     4     2
```

```
>> tril(A,2)
ans =
     4     3     1     0     0
     9     8     1     6     0
     6     8     6     5     4
     6     4     8     1     6
    10     6    10     4     2
>> tril(A,-2)
ans =
     0     0     0     0     0
     0     0     0     0     0
     6     0     0     0     0
     6     4     0     0     0
    10     6    10     0     0
```

三角阵

```
>> inv(triu(A))
ans =
    0.2500   -0.0938   -0.0260    0.1927   -0.4010
         0    0.1250   -0.0208   -0.6458    1.4792
         0         0    0.1667   -0.8333    2.1667
         0         0         0    1.0000   -3.0000
         0         0         0         0    0.5000

>> inv(tril(A))
ans =
    0.2500   -0.0000    0.0000   -0.0000    0.0000
   -0.2813    0.1250   -0.0000    0.0000   -0.0000
    0.1250   -0.1667    0.1667   -0.0000    0.0000
   -1.3750    0.8333   -1.3333    1.0000   -0.0000
    1.7188   -1.2083    1.8333   -2.0000    0.5000
```

三角阵

- chol Cholesky factorization. 用于正定矩阵分解。
- R = chol(A) produces an upper triangular matrix R from the diagonal and upper triangle of matrix A, satisfying the equation R'*R=A.
- L = chol(A,'lower') uses only the diagonal and the lower triangle of A to produce a lower triangular L so that L*L' = A.

```
>> A=gallery('moler',5)
A =
     1    -1    -1    -1    -1
    -1     2     0     0     0
    -1     0     3     1     1
    -1     0     1     4     2
    -1     0     1     2     5
```

```
>> C=chol(A)
C =
     1    -1    -1    -1    -1
     0     1    -1    -1    -1
     0     0     1    -1    -1
     0     0     0     1    -1
     0     0     0     0     1
```

三角阵

```
>> L = chol(A,'lower')
L =
     1     0     0     0     0
    -1     1     0     0     0
    -1    -1     1     0     0
    -1    -1    -1     1     0
    -1    -1    -1    -1     1
>> L*L'
ans =
     1    -1    -1    -1    -1
    -1     2     0     0     0
    -1     0     3     1     1
    -1     0     1     4     2
    -1     0     1     2     5
```

```
>> A = diag([2 3 4 5 6])
A =
     2     0     0     0     0
     0     3     0     0     0
     0     0     4     0     0
     0     0     0     5     0
     0     0     0     0     6
>> C=chol(A)
C =
    1.4142         0         0         0         0
         0    1.7321         0         0         0
         0         0    2.0000         0         0
         0         0         0    2.2361         0
         0         0         0         0    2.4495
```

范德蒙矩阵

- 范德蒙矩阵（Vandermonde matrix)
- 每行或每列是一个等比序列。
- 性质：当 $x_1,\cdots,x_n$ 各异时，矩阵A非奇异

$$A=\begin{bmatrix}1 & x_1 & x_1^2 & \cdots & x_1^{n-1}\\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1}\\ \vdots & \vdots & \vdots & & \vdots\\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1}\end{bmatrix}\quad A=\begin{bmatrix}1 & 1 & \cdots & 1\\ x_1 & x_2 & \cdots & x_n\\ x_1^2 & x_2^{n-1} & \cdots & x_n^{n-1}\\ \vdots & \vdots & & \vdots\\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1}\end{bmatrix}$$

范德蒙矩阵

- 证明范德蒙德(Vandermonde)行列式

$$D_n = \begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{vmatrix} = \prod_{n \ge i > j \ge 1} (x_i - x_j). \quad (1)$$

证明　用数学归纳法

$$D_2 = \begin{vmatrix} 1 & 1 \\ x_1 & x_2 \end{vmatrix} = x_2 - x_1 = \prod_{2 \ge i > j \ge 1} (x_i - x_j)$$

所以n=2时(1)式成立.

范德蒙矩阵

假设(1)对于$n-1$阶范德蒙行列式成立，从第n行开始，后行减去前行的x_1倍：

$$D_n = \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & x_2 - x_1 & x_3 - x_1 & \cdots & x_n - x_1 \\ 0 & x_2(x_2 - x_1) & x_3(x_3 - x_1) & \cdots & x_n(x_n - x_1) \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & x_2^{n-2}(x_2 - x_1) & x_3^{n-2}(x_3 - x_1) & \cdots & x_n^{n-2}(x_n - x_1) \end{vmatrix}$$

按照第1列展开，并提出每列的公因子$(x_i - x_1)$，就有

范德蒙矩阵

$$= (x_2 - x_1)(x_3 - x_1)\cdots(x_n - x_1)\begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_2 & x_3 & \cdots & x_n \\ \vdots & \vdots & & \vdots \\ x_2^{n-2} & x_3^{n-2} & \cdots & x_n^{n-2} \end{vmatrix}$$

$n-1$阶范德蒙德行列式

$$\therefore \quad D_n = (x_2 - x_1)(x_3 - x_1)\cdots(x_n - x_1)\prod_{n \ge i > j \ge 2}(x_i - x_j)$$

$$= \prod_{n \ge i > j \ge 1}(x_i - x_j).$$

范德蒙矩阵

复Vandermonde矩阵的逆矩阵

$$A=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_n \\ \vdots & \vdots & & \vdots \\ a_1^{n-1} & a_2^{n-1} & \cdots & a_n^{n-1} \end{bmatrix}, a_k \in C$$

$$\sigma_1(a_1,a_2,\cdots,a_k)=a_1+a_2+\cdots+a_k$$

$$\sigma_2(a_1,a_2,\cdots,a_k)=a_1a_2+\cdots+a_1a_k+a_2a_3+\cdots+a_2a_k+\cdots+a_{k-1}a_k$$

$$\vdots$$

$$\sigma_k(a_1,a_2,\cdots,a_k)=a_1a_2\cdots a_k$$

$$A^{-1}=$$

$$\begin{bmatrix} \dfrac{\sigma_{n-1}(a_2,a_3,\cdots,a_{n)}}{\prod\limits_{k=2}^{n}(a_k-a_1)} & -\dfrac{\sigma_{n-2}(a_2,a_3,\cdots,a_{n)}}{\prod\limits_{k=2}^{n}(a_k-a_1)} & \cdots & \dfrac{(-1)^{n+1}}{\prod\limits_{k=2}^{n}(a_k-a_1)} \\ -\dfrac{\sigma_{n-1}(a_1,a_3,\cdots,a_{n)}}{(a_2-a_1)\prod\limits_{k=3}^{n}(a_k-a2)} & \dfrac{\sigma_{n-2}(a_1,a_3,\cdots,a_{n)}}{(a_2-a_1)\prod\limits_{k=3}^{n}(a_k-a2)} & \cdots & \dfrac{(-1)^{n+2}}{(a_2-a_1)\prod\limits_{k=3}^{n}(a_k-a2)} \\ \vdots & \vdots & \vdots & \vdots \\ \dfrac{\sigma_{n-1}(a_1,a_2,\cdots,a_{n-1)}}{(-1)^{n+1}\prod\limits_{k=1}^{n-1}(a_n-a_k)} & \dfrac{\sigma_{n-2}(a_2,a_3,\cdots,a_{n)}}{(-1)^{n+2}\prod\limits_{k=1}^{n-1}(a_n-a_k)} & \cdots & \dfrac{1}{\prod\limits_{k=1}^{n-1}(a_n-a_k)} \end{bmatrix}$$

范德蒙矩阵

```
>> v = 1:.5:3
A = vander(v)
v =     1.0000    1.5000    2.0000    2.5000    3.0000
A =
    1.0000    1.0000    1.0000    1.0000    1.0000
    5.0625    3.3750    2.2500    1.5000    1.0000
   16.0000    8.0000    4.0000    2.0000    1.0000
   39.0625   15.6250    6.2500    2.5000    1.0000
   81.0000   27.0000    9.0000    3.0000    1.0000
>> A = fliplr(vander(v))
A =
    1.0000    1.0000    1.0000    1.0000    1.0000
    1.0000    1.5000    2.2500    3.3750    5.0625
    1.0000    2.0000    4.0000    8.0000   16.0000
    1.0000    2.5000    6.2500   15.6250   39.0625
    1.0000    3.0000    9.0000   27.0000   81.0000
```

- A = vander(v) returns the Vandermonde Matrix such that its columns are powers of the vector v.

范德蒙矩阵

- **Prony方法：用指数模型拟合信号**

$$B_q(z) = b_0 + b_1 z^{-1} + \ldots + b_q z^{-q}$$

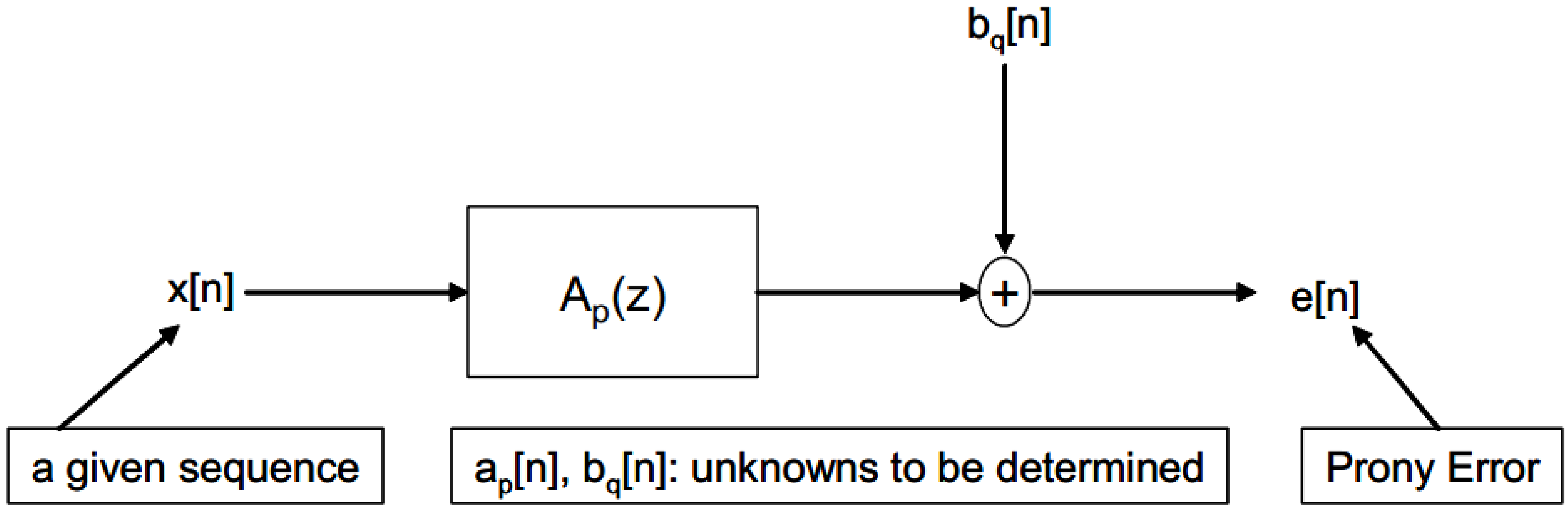
$$A_p(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_p z^{-p}$$

Let $e'[n] = x[n] - h[n]$ then $E'(z) = X(z) - H(z) = X(z) - \frac{Bq(z)}{A_p(z)}$.

We call $E(z) = E'(z)A_p(z)$ as the Prony error:

范德蒙矩阵

- **Prony方法：用指数模型拟合信号**



$$e[n] = a_p[n] * x[n] - b_q[n] = \begin{cases} x[n] + \sum_{l=1}^{P} a_p[l]x[n-l] - b_q[n] & 0 \le n \le q \\ x[n] + \sum_{l=1}^{P} a_p[l]x[n-l] & n \ge q+1 \end{cases}$$

范德蒙矩阵

- **Prony方法：用指数模型拟合信号**

Hence Prony cost function optimizes over $a_p[n]$ first and it is defined as

$$J(a_p) = \sum_{n=q+1}^{\infty} |e[n]|^2 \text{ where } q \text{ is the number of poles}$$

Then equating $\frac{\partial}{\partial a_p^*[k]} J(a_p) = 0$ for $1 \leq k \leq p$, we get the following system of equations:

$$\begin{bmatrix} r_x(1,1) & r_x(1,2) & \ldots & r_x(1,P) \\ r_x(2,1) & r_x(2,2) & \ldots & r_x(2,P) \\ \vdots & \vdots & & \vdots \\ r_x(P,1) & r_x(P,2) & \ldots & r_x(P,P) \end{bmatrix} \begin{bmatrix} a_p[1] \\ a_p[2] \\ \vdots \\ a_p[P] \end{bmatrix} = - \begin{bmatrix} r_x(1,0) \\ r_x(2,0) \\ \vdots \\ r_x(P,0) \end{bmatrix}$$

From the equation system, we can solve for the unknown $a_p[k]$'s.

傅里叶矩阵

Fourier矩阵： Fourier变换（DFT）

$$\hat{x}(k)=\sum_{n=0}^{N-1}x(n)e^{-j2\pi nk/N}=\sum_{n=0}^{N-1}x(n)w^{nk},k=0,1,\ldots,N-1$$

$$\begin{bmatrix}\hat{x}(0)\\ \hat{x}(1)\\ \vdots\\ \hat{x}(N-1)\end{bmatrix}=\begin{bmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & w & w^2 & \cdots & w^{N-1}\\ \vdots & \vdots & \vdots & & \vdots\\ 1 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)(N-1)}\end{bmatrix}\begin{bmatrix}x(0)\\ x(1)\\ \vdots\\ x(N-1)\end{bmatrix}$$

简记作 $\hat{x}=Fx$ ，其中F 称为**Fourier矩阵**. 由 $F^HF=NI$，可得

$$x=\frac{1}{N}F^H\hat{x}$$

--**非对称Fourier变换对**.

傅里叶矩阵

若

$$\overline{F}=\frac{1}{\sqrt{N}}\begin{bmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & w & w^2 & \cdots & w^{N-1}\\ \vdots & \vdots & \vdots & & \vdots\\ 1 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)(N-1)}\end{bmatrix}, w=e^{-j\frac{2\pi}{N}}$$

则

$$\hat{x}(k)=\frac{1}{\sqrt{N}}\sum_{n=0}^{N-1}x(n)e^{-j2\pi nk/N}, k=0,1,\ldots,N-1$$

$$x(n)=\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}\hat{x}(k)e^{j2\pi nk/N}, n=0,1,\ldots,N-1$$

归一化Fourier是正交矩阵.

即有 $\hat{x}=\overline{F}x$ 和 $x=\overline{F}^H\hat{x}$ **--对称Fourier变换对**

傅里叶矩阵

Fourier矩阵的性质

（1）　$F^T = F,\qquad F^{-1} = \frac{1}{N}F^*$

（2）　$$\boxed{\frac{1}{N}\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}nm} = \begin{cases} 1, & m=0,\pm N,\pm 2N,\ldots \\ 0, & \text{otherwise} \end{cases} = \sum_{k=-\infty}^{\infty}\delta[m-kN]\,.}$$

$$e^{j2\pi n(kN)/N} = e^{j2\pi nk} = 1 \text{ so } \frac{1}{N}\sum_{n=0}^{N-1} 1 = 1.$$

其他 m 有 $\frac{1}{N}\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}nm} = \frac{1}{N}\sum_{n=0}^{N-1}\left(e^{j\frac{2\pi}{N}m}\right)^n = \frac{1}{N}\frac{1-\left(e^{j\frac{2\pi}{N}m}\right)^N}{1-\left(e^{j\frac{2\pi}{N}m}\right)} = \frac{1}{N}\frac{1-1}{1-\left(e^{j\frac{2\pi}{N}m}\right)} = 0.$

$$\langle \boldsymbol{w}^k, \boldsymbol{w}^l\rangle = \sum_{n=0}^{N-1} w_n^k (w_n^l)^* = \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}kn}\, e^{-j\frac{2\pi}{N}ln} = \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(k-l)n} = N\,\delta[k-l],$$

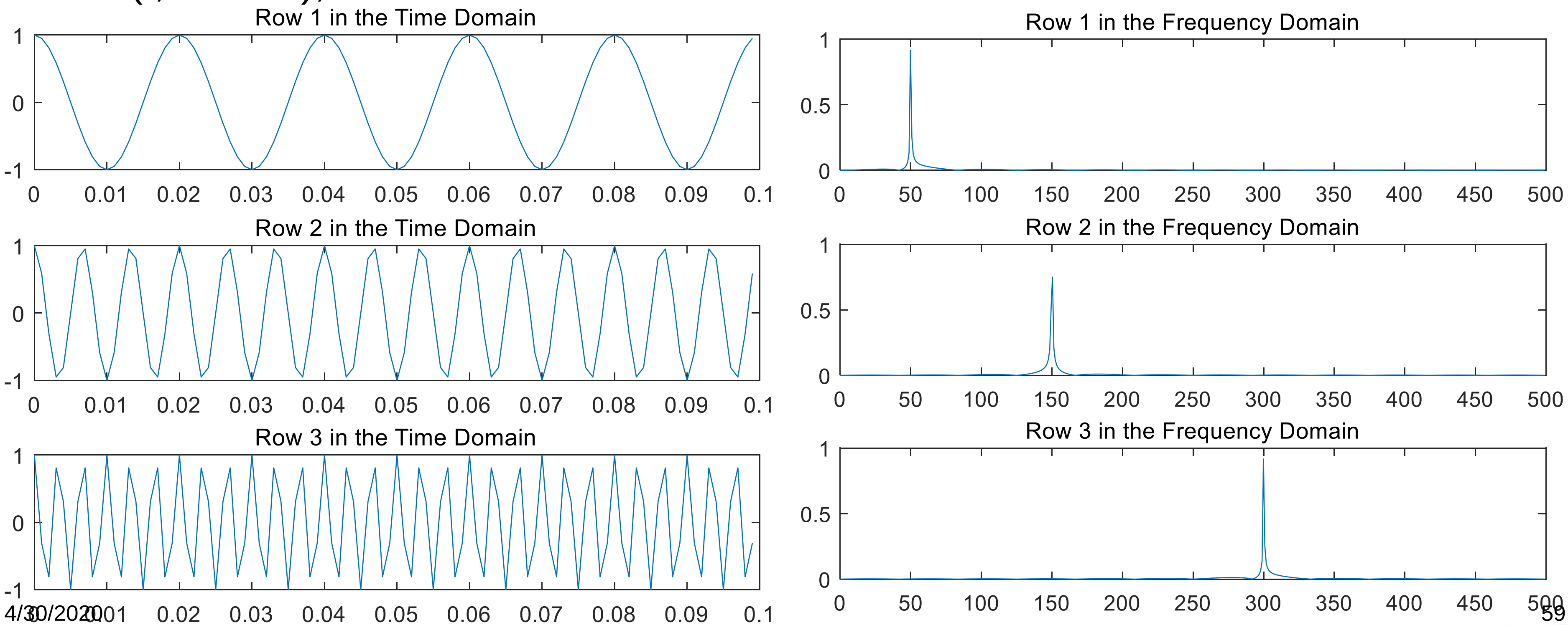
（3）　$\bar{F}^2 = [e_1, e_N, e_{N-1}, \cdots, e_2]$,置换矩阵

$\bar{F}^4 = I$

傅里叶矩阵

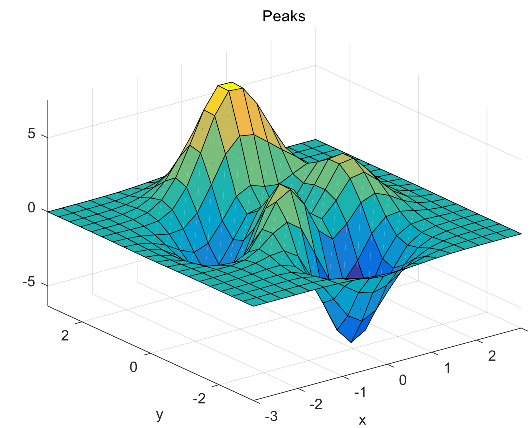
Y = fft(X) 一维傅里叶变换，反变换：ifft

Fs = 1000; T = 1/Fs; L = 1000; t = (0:L-1)*T; x1 = cos(2*pi*50*t);
x2 = cos(2*pi*150*t); x3 = cos(2*pi*300*t); X = [x1; x2; x3]; n = 2^nextpow2(L);
dim = 2;Y = fft(X,n,dim); P2 = abs(Y/n); P1 = P2(:,1:n/2+1);P1(:,2:end-1) =
2*P1(:,2:end-1);

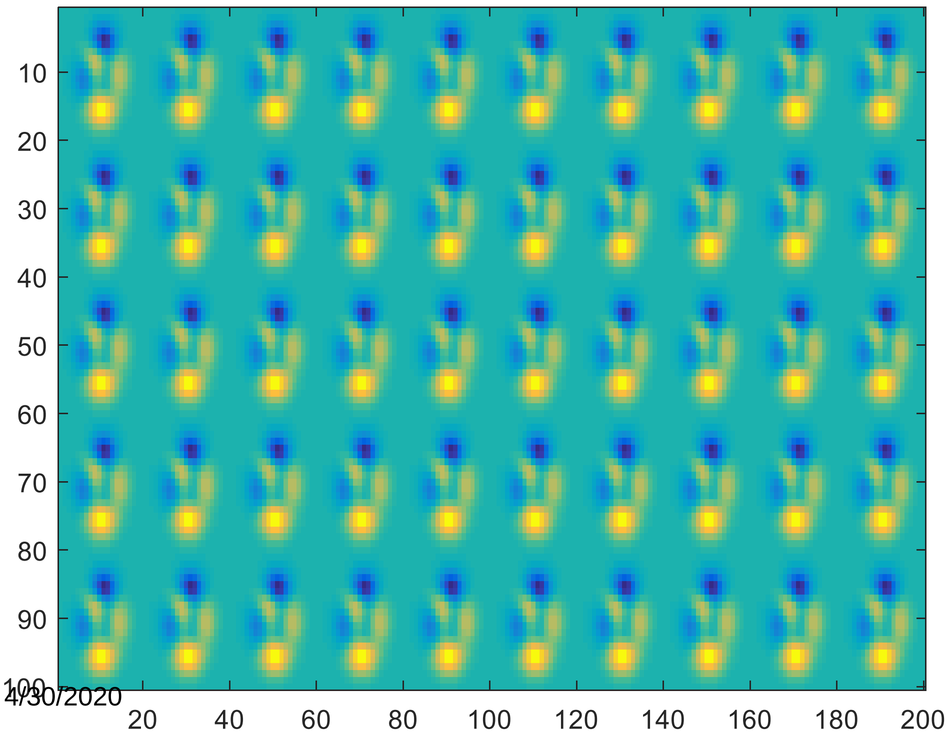


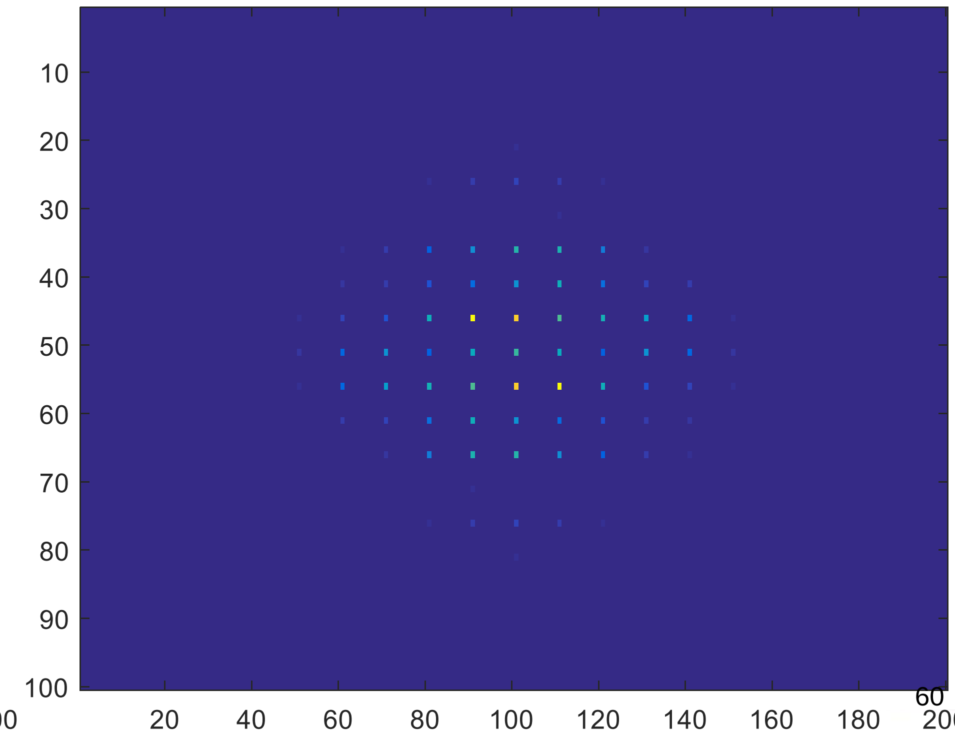
傅里叶矩阵

Y = fft2(X) 二维傅里叶变换，反变换ifft2




```
P = peaks(20); X = repmat(P,[5 10]);
imagesc(X)
Y = fft2(X);
 imagesc(abs(fftshift(Y)))
```





傅里叶矩阵

Y = fft2(X) 二维傅里叶变换，反变换ifft2

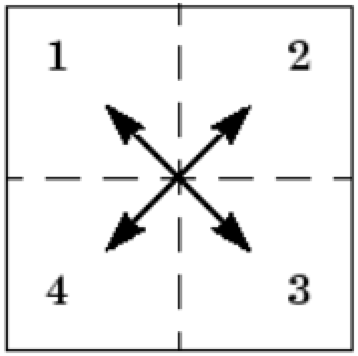
fftshift(Y)

1、在matlab中，经过fft变换后，数据的频率范围是从[0,fs]排列的。

2、而一般，我们在画图或者讨论的时候，是从[-fs/2,fs/2]的范围进行分析。

3、因此，需要将经过fft变换后的图像的[fs/2,fs]部分移动到[-fs/2,0]这个范围内。

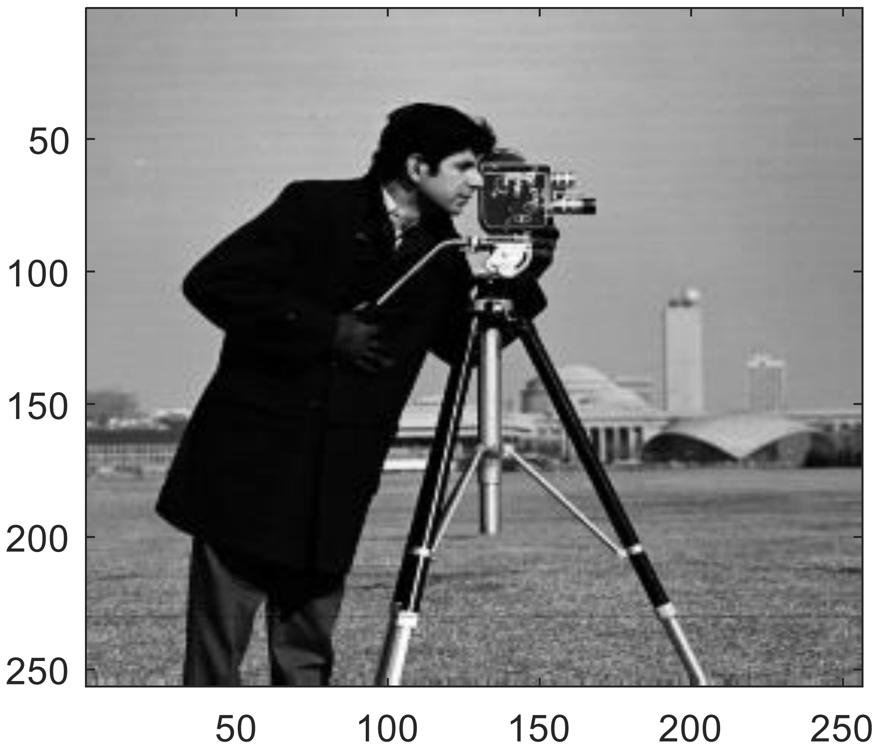
而fftshift就是度完成这个功能。通常，如果想得到所见的中间是0频的图像，经过fft变换后，都要再经过fftshift。

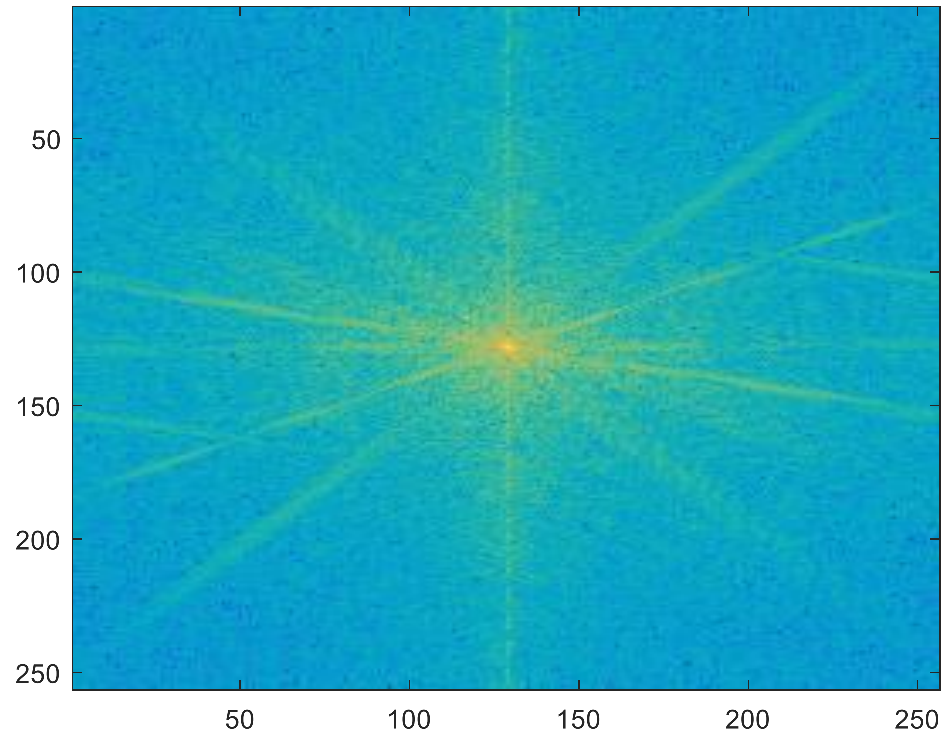


傅里叶矩阵

Y = fft2(X) 二维傅里叶变换

```
I=imread('cameraman.tif');
Y = fft2(I);
imagesc(log(abs(fftshift(Y))))
```



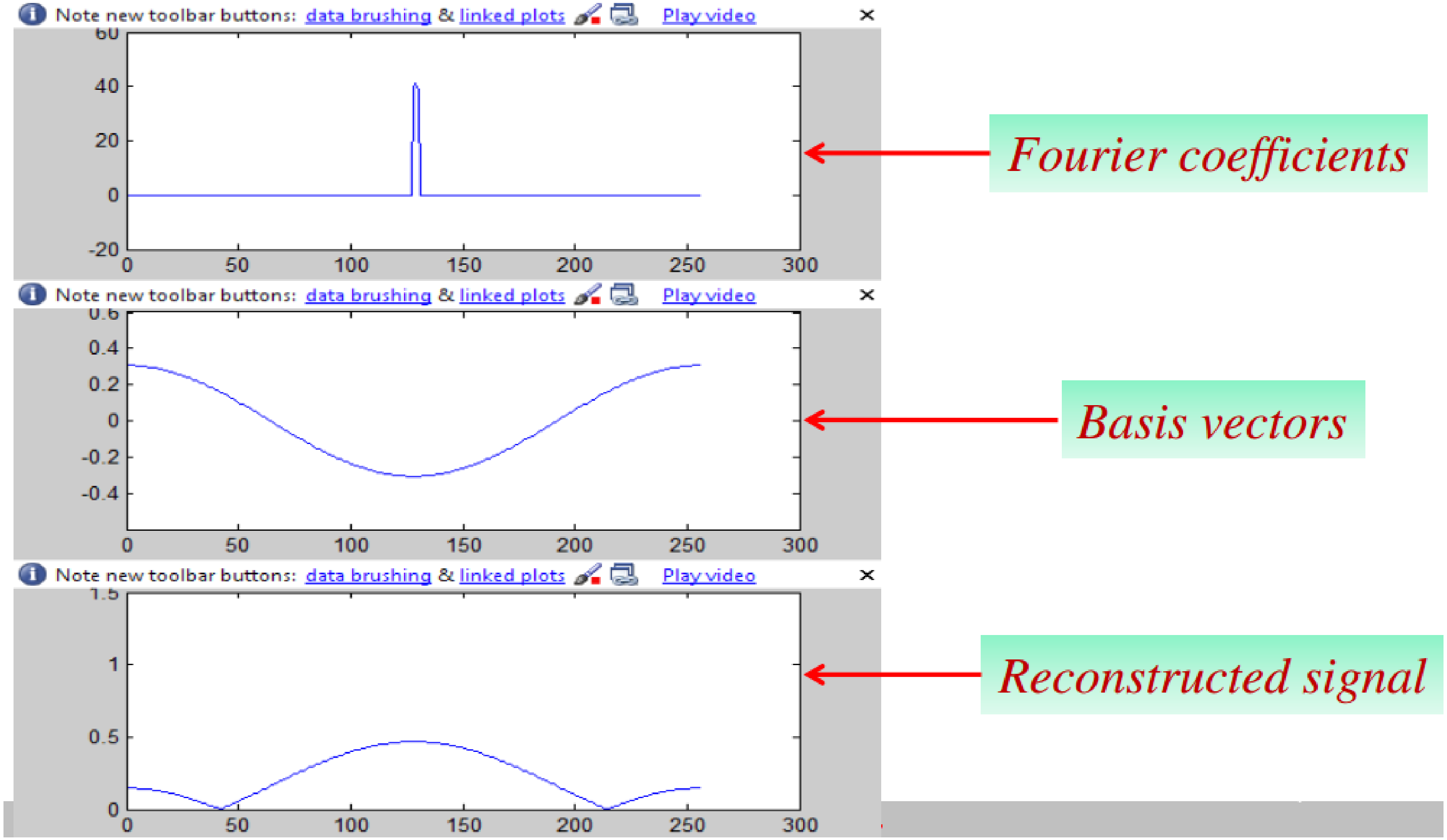


傅里叶矩阵

- Fourier basis is a collection of harmonics
  - Note that complex exponentials are simply sines and cosines
- Therefore the FT simply decomposes a signal into its harmonic components
- FT gives direct information about the sharpness and oscillations present in the data
- An “alternate view” of the data

From Ashish Raj(cornell)

傅里叶矩阵



From Ashish Raj(cornell)

傅里叶矩阵

- FTs are great, but they capture global features
  - Harmonic components of the entire signal
  - They are obtained by dot-producting the WHOLE signal
- Problem1: local features can get lost
- Problem2: if signal is not stationary (features change with time or in space) then this is not captured by FT
- Therefore need a transform that provides frequency information LOCALLY

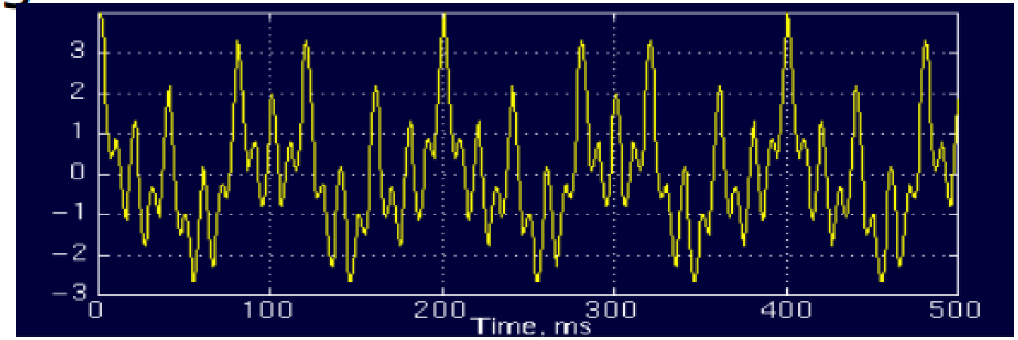
From Ashish Raj(cornell)

傅里叶矩阵

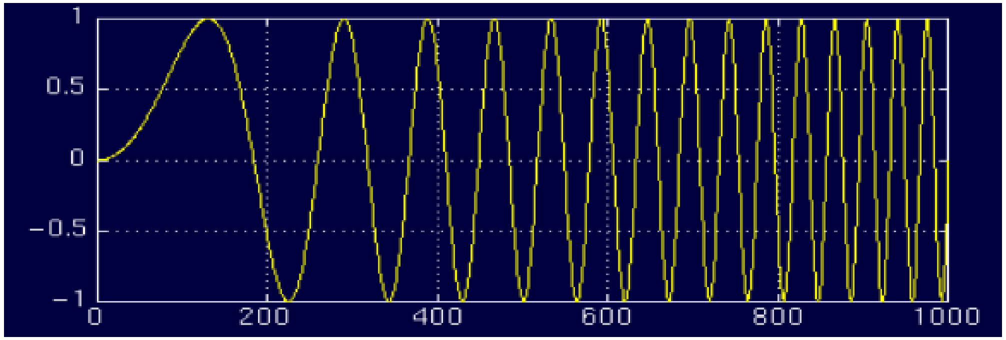
- For example consider the following signal

x(t)=cos(2*pi*10*t)+cos(2*pi*25*t)+cos(2*pi*50*t)+cos(2*pi*100*t)

- Has frequencies 10, 25, 50, and 100 Hz at any given time instant



stationary signal, FT can provide full info



Non-stationary, frequency content changes with time
FT CANNOT provide full info

From Ashish Raj(cornell)

Wavelet Transform

I'm going to ignore the complex conjugate from now on, assuming that we're using real wavelets

time-series

$$\gamma(s,\tau) = \int f(t)\psi^{*}_{s,\tau}(t)\,dt$$

coefficient of wavelet with
scale, s and time, τ

complex conjugate of wavelet with
scale, s and time, τ

小波变换

normalization

shift in time

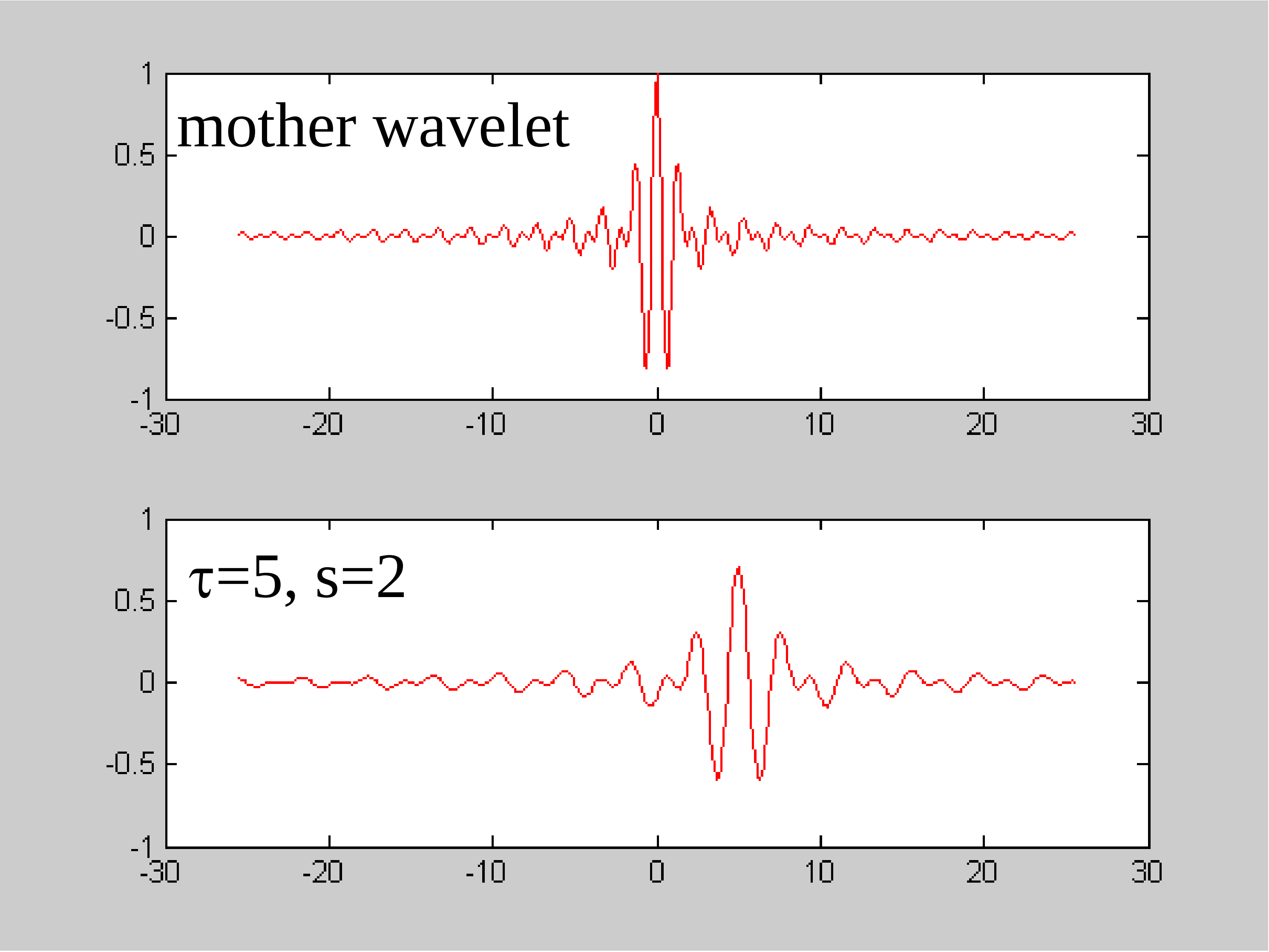
$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\,\psi\!\left(\frac{t-\tau}{s}\right)$$

change in scale:
big s means long
wavelength

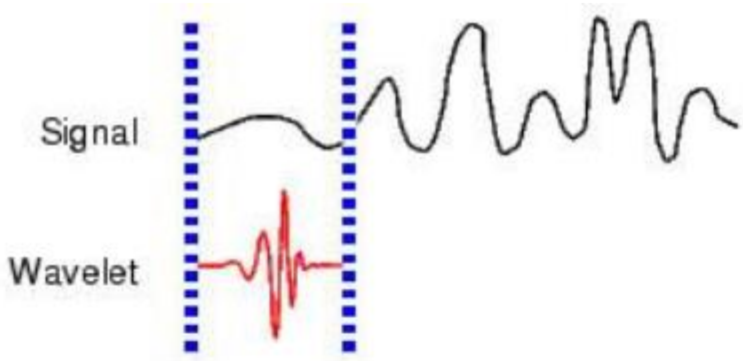
wavelet with
scale, s and time, τ

Mother wavelet

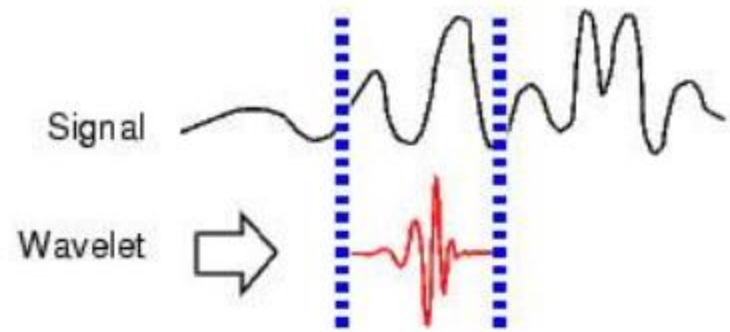
小波变换



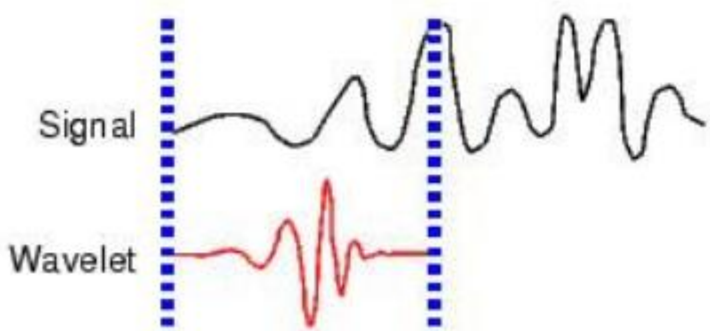
小波变换



Low value for $W_{\psi}(s,\tau)$



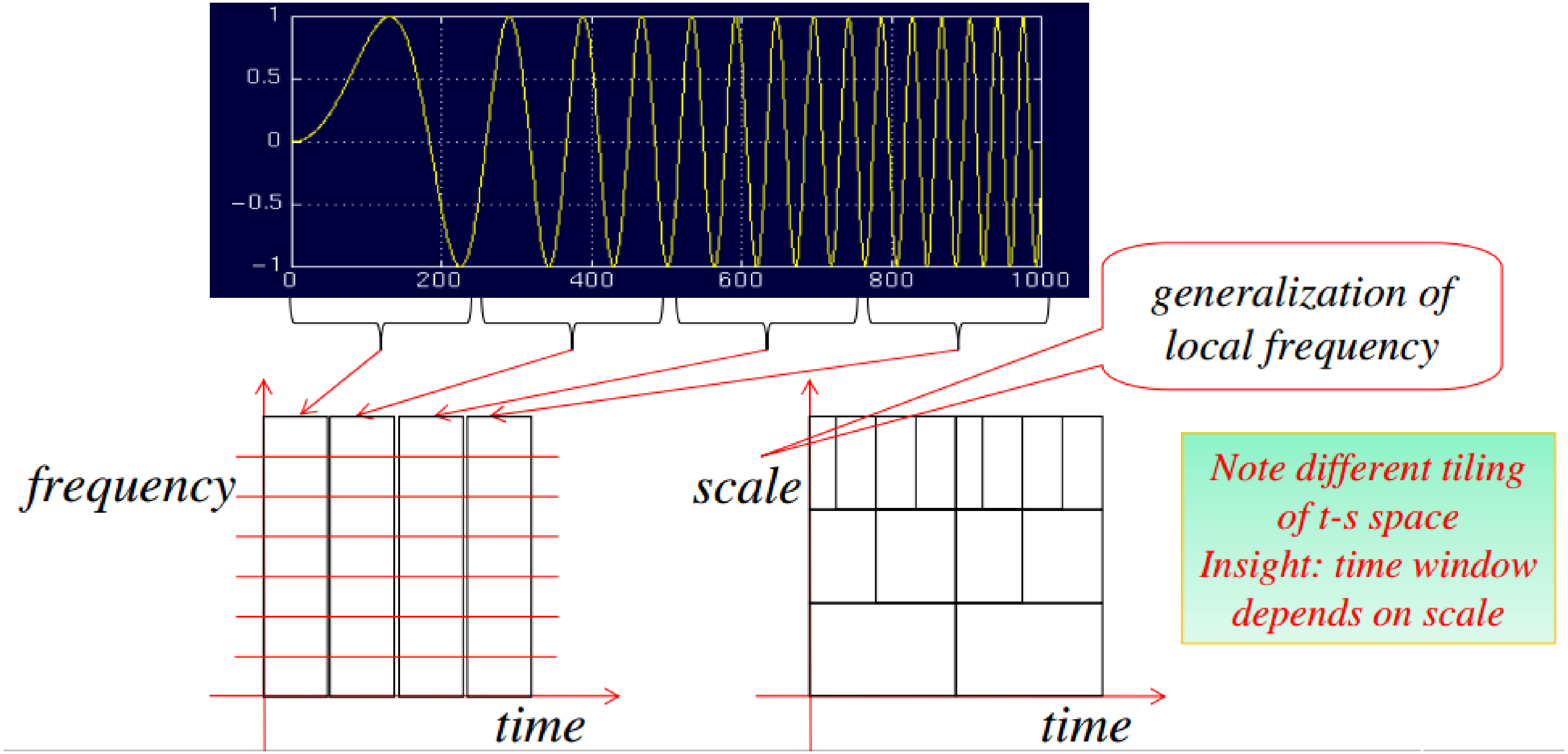
Higher value of $W_{\psi}(s,\tau_2)$



Different scale

小波变换

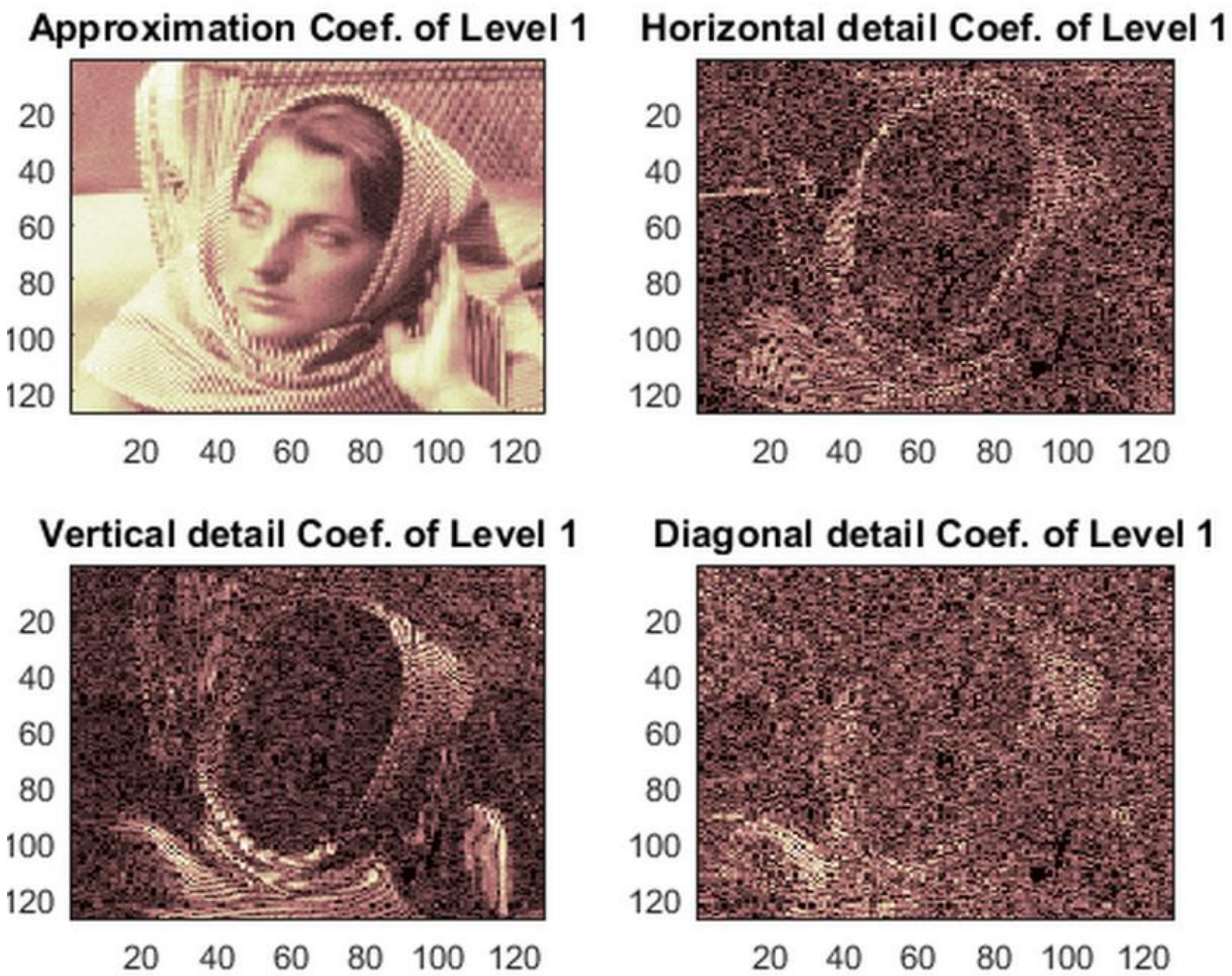
- What we need is a time-frequency analysis
- Do FT in a local time window



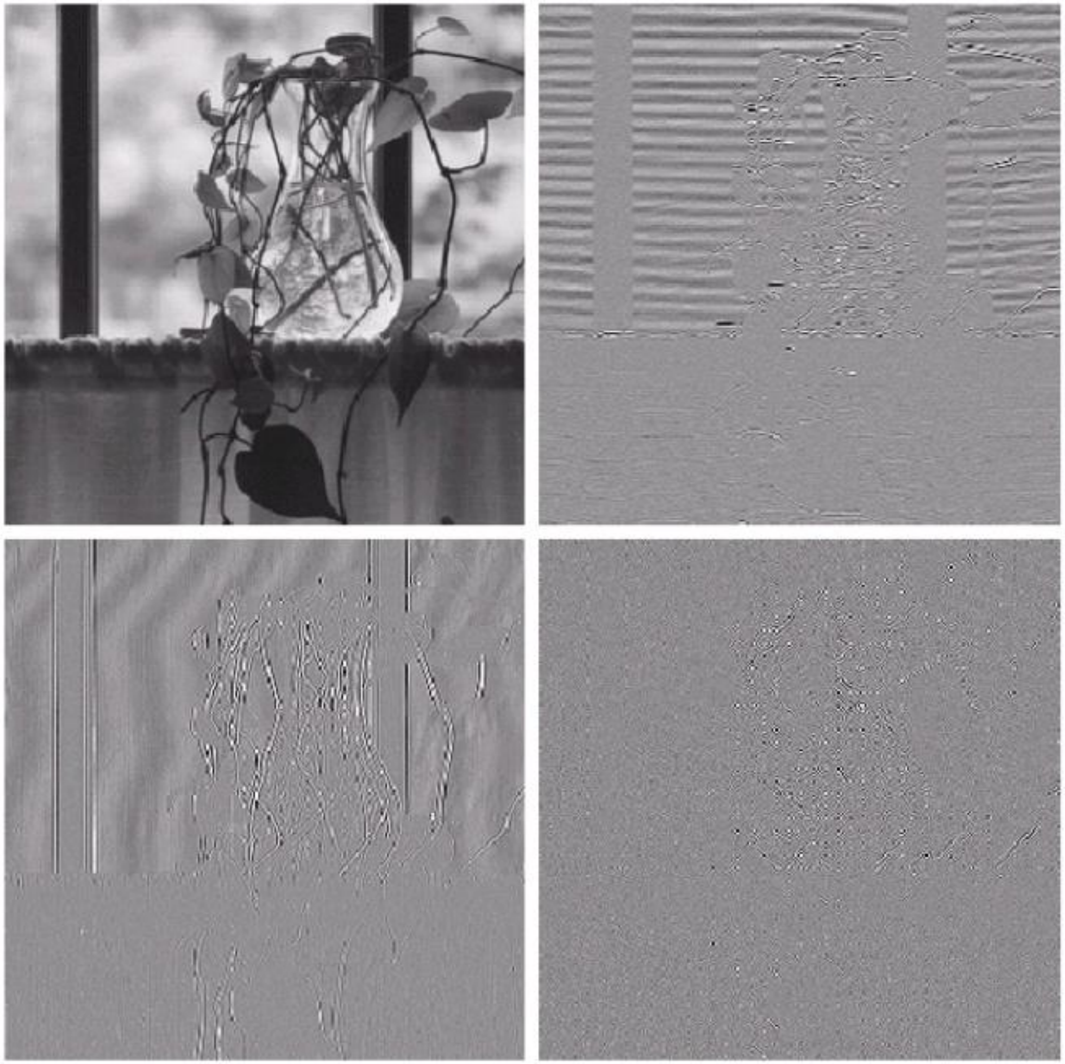
From Ashish Raj(cornell)

小波变换

wavedec2 的参考页

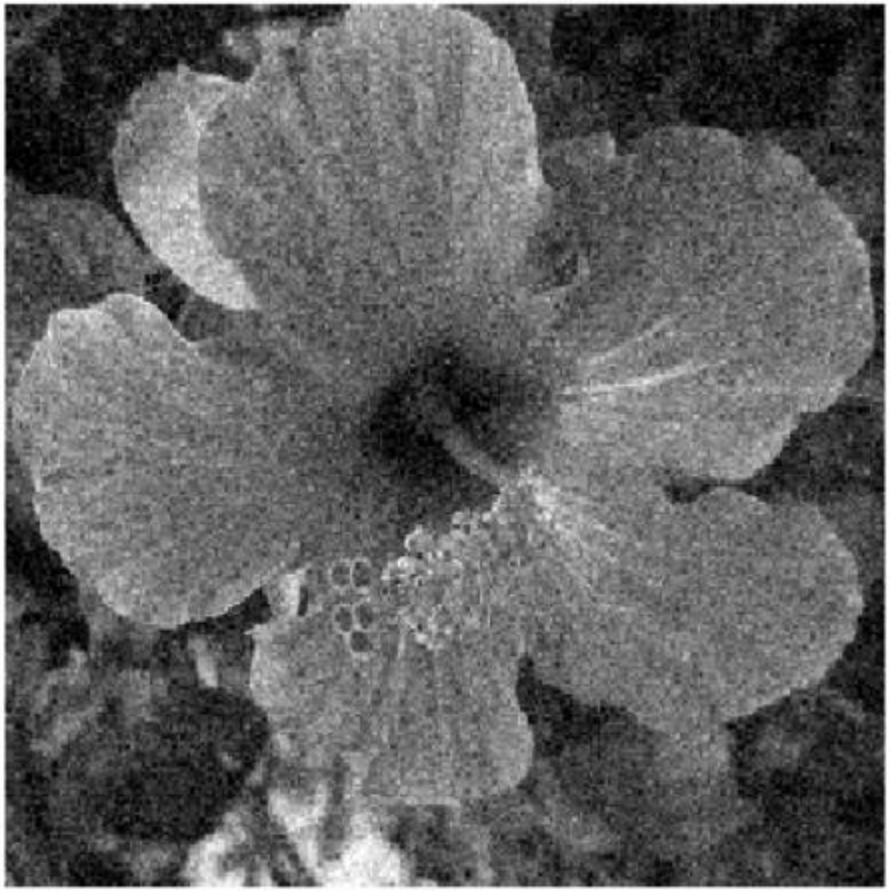


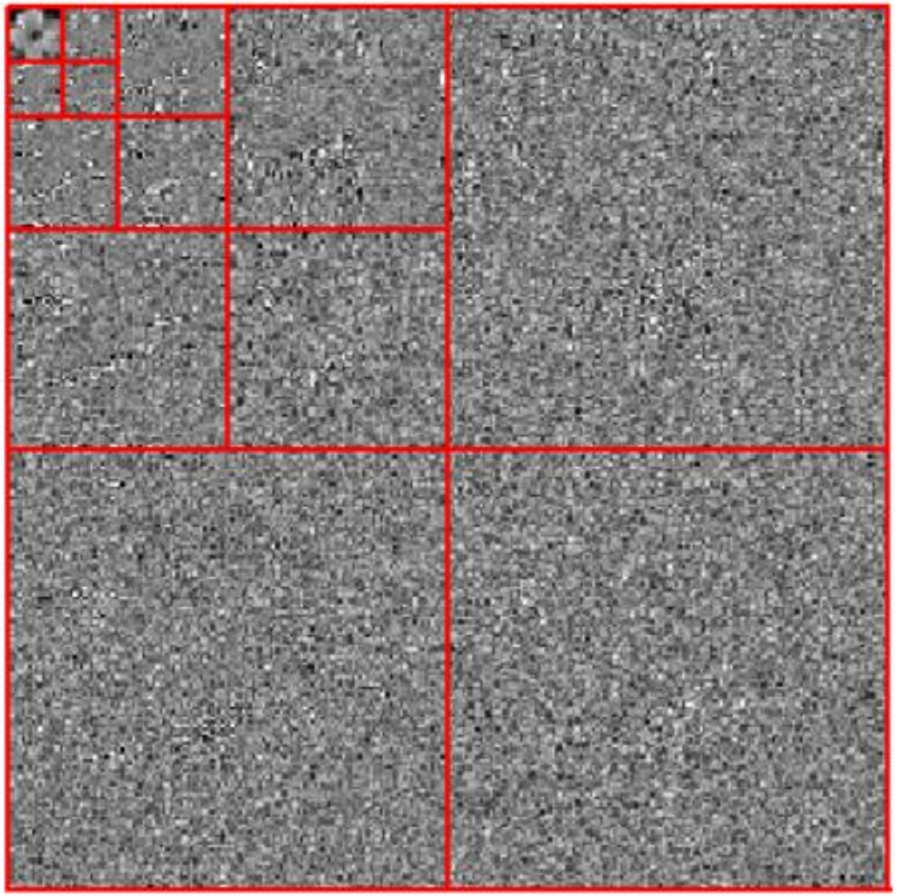
小波变换



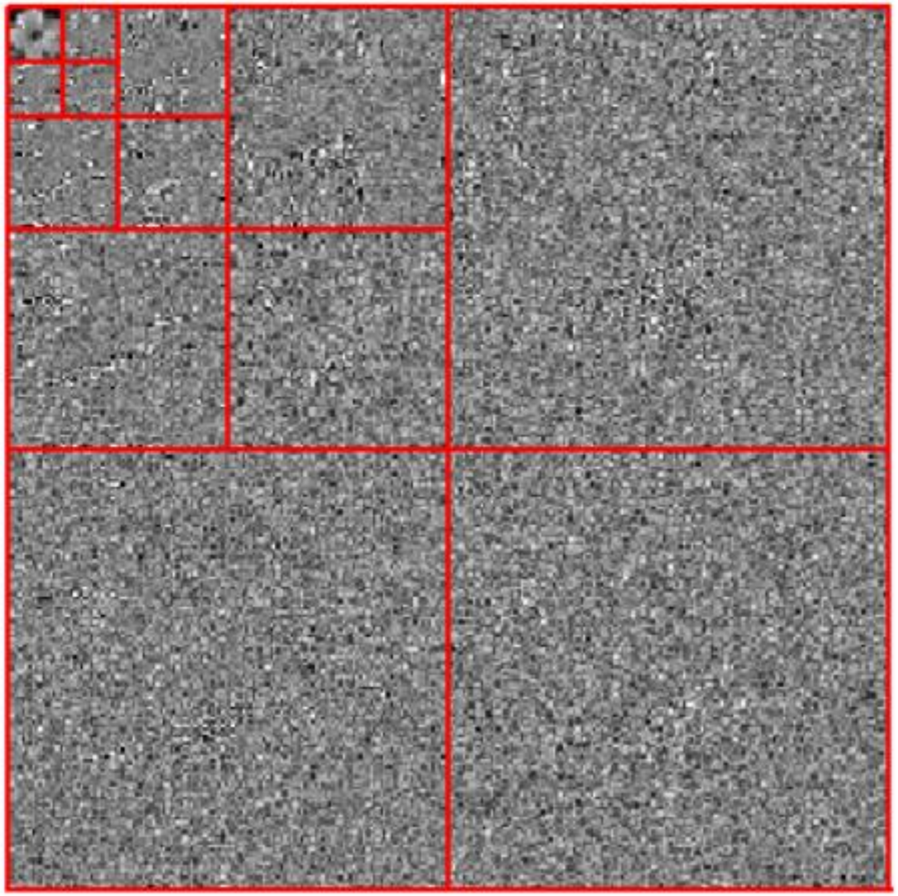
小波变换

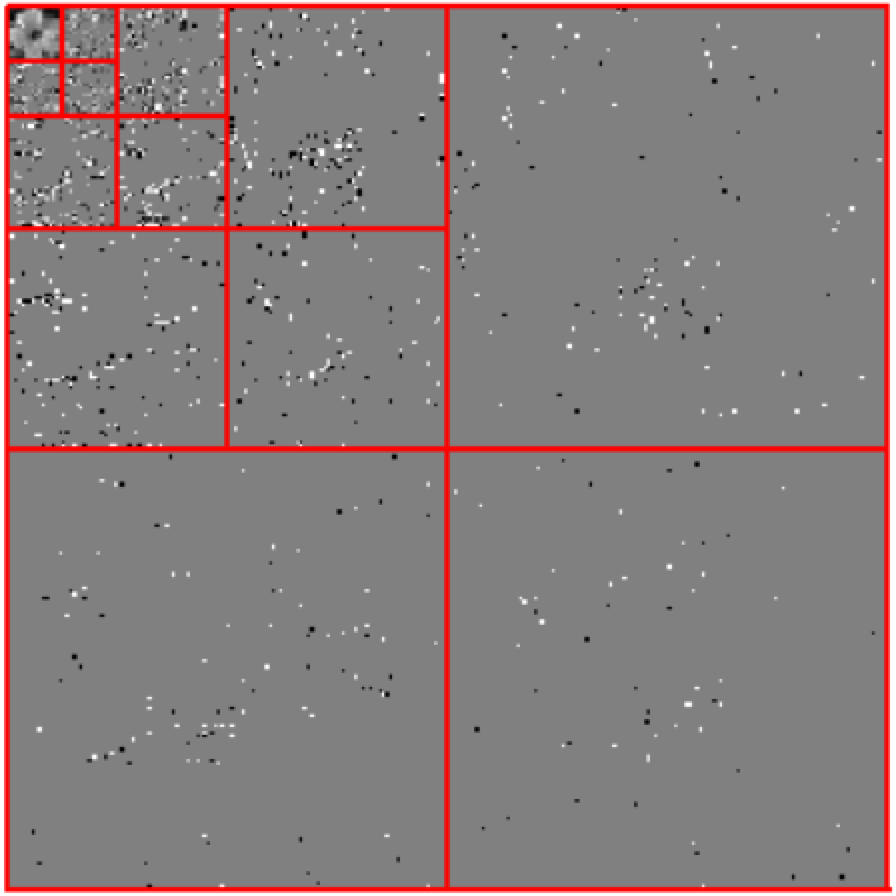
Noisy, SNR=14.1





小波变换



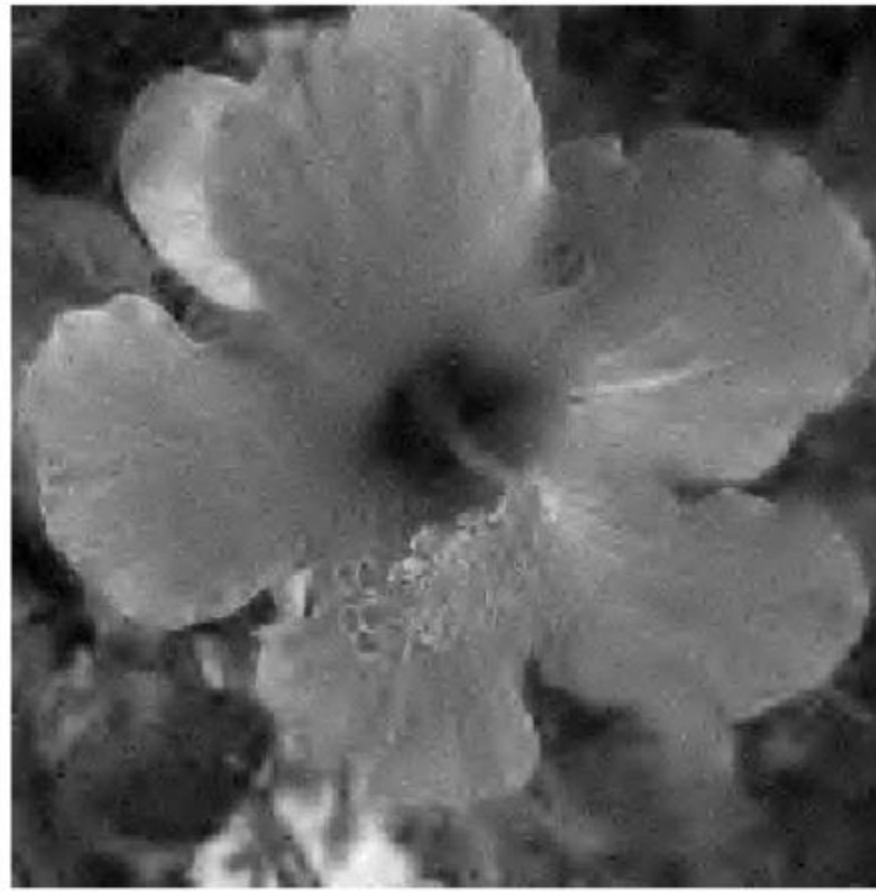


去噪

小波变换

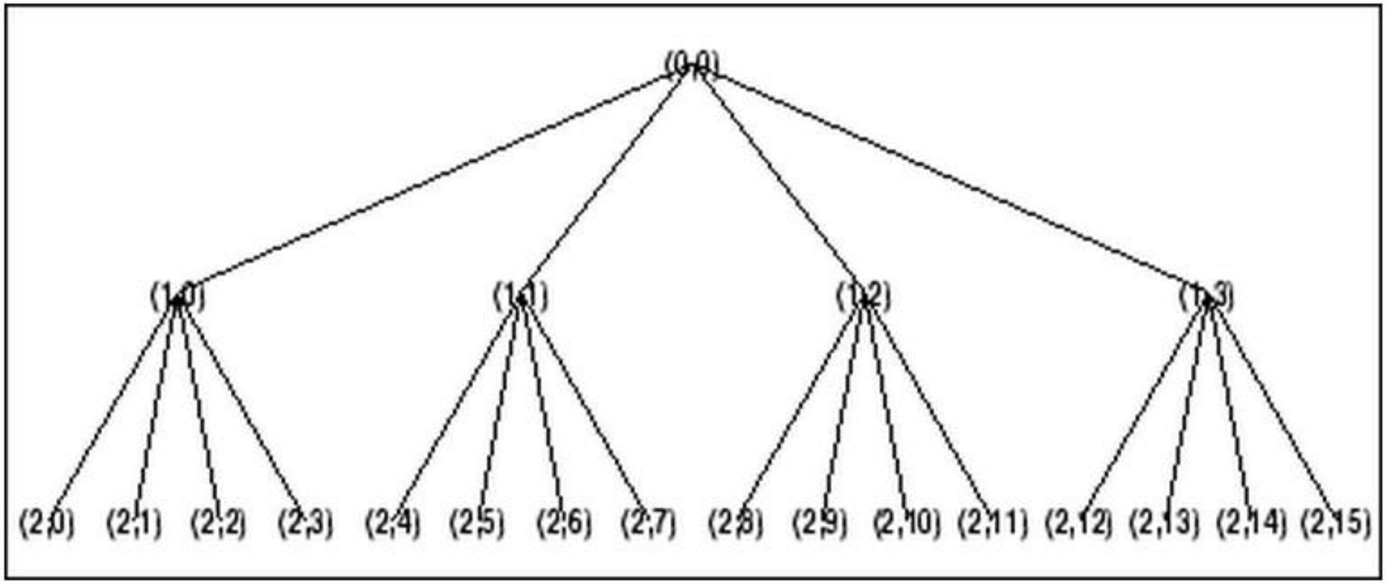
Hard denoising, SNR=19.2

Soft denoising, SNR=19.8



小波变换

wpdec2



小波变换

- Many, many applications!
- Audio, image and video compression
- New JPEG standard includes wavelet compression
- FBI's fingerprints database saved as wavelet-compressed
- Signal denoising, interpolation, image zooming, texture analysis, time-scale feature extraction
- In our context, WT will be used primarily as a feature extraction tool
- Remember, WT is just a change of basis, in order to extract useful information which might otherwise not be easily seen

From Ashish Raj(cornell)

哈达玛矩阵

◊ **Hadamard矩阵**：所有元素取+1或-1，并且满足

$$H_n H_n^T = H_n^T H_n = nI_n$$

的 $n \times n$ 正方矩阵 H_n 称为n阶**Hadamard矩阵**. n、n/12 或 n/20 必须为 2 的幂。

性质：

（1）Hadamard矩阵的每一行或列均由+1或-1构成，且两两正交.特别地，$\frac{1}{\sqrt{n}}H_n$ 为标准正交矩阵.

（2）用-1乘Hadamard矩阵的任意一行(列),所得结果仍为一Hadamard矩阵.于是，可以得到第一列和第一行的所有元素为 +1 的Hadamard矩阵，并称之为**规范化Hadamard矩阵**.

哈达玛矩阵

（3） $\det(H_n) = \pm n^{n/2}$

定理：令 $n=2^k, k=1,2,\ldots$ ，则 $\bar{H}_{2n} = \frac{1}{\sqrt{2}}\begin{bmatrix} \bar{H}_n & \bar{H}_n \\ \bar{H}_n & -\bar{H}_n \end{bmatrix}$ ，其中

$$\bar{H}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

注意：$\bar{H}_{2n}$ 为规范化的标准正交Hadamard矩阵。

可以用归纳法证明。

哈达玛矩阵

```
>> H = hadamard(4)
H =
     1     1     1     1
     1    -1     1    -1
     1     1    -1    -1
     1    -1    -1     1
det(H)
ans =    16
```

```
>> H = hadamard(8)
H =
     1     1     1     1     1     1     1     1
     1    -1     1    -1     1    -1     1    -1
     1     1    -1    -1     1     1    -1    -1
     1    -1    -1     1     1    -1    -1     1
     1     1     1     1    -1    -1    -1    -1
     1    -1     1    -1    -1     1    -1     1
     1     1    -1    -1    -1    -1     1     1
     1    -1    -1     1    -1     1     1    -1
>> det(H)
ans =        4096
```

拓普利兹矩阵

◊**Toeplitz矩阵：**

任何一条对角线的元素取相同值的特殊矩阵

$$A=\begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & a_{-n} \\ a_1 & a_0 & a_{-1} & \cdots & a_{-n+1} \\ a_2 & a_1 & a_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & a_{-1} \\ a_n & a_{n-1} & \cdots & a_1 & a_0 \end{bmatrix}=\left[a_{i-j}\right]_{i,j=0}^{n}$$

★任一Toeplitz矩阵均为斜对称矩阵；

拓普利兹矩阵

对称Toeplitz矩阵：满足对称关系 $a_{-i}=a_i, i=1,2,\ldots,n$ 的Toeplitz矩阵.

★对称Toeplitz矩阵可仅由其第一行元素完全描述.因此，常将其简记作 $A=Toep[a_0,a_1,\ldots,a_n]$

Hermitian Toeplitz 矩阵：斜Hermitian Toeplitz 矩阵：

$$A=\begin{bmatrix} a_0 & a_1^* & a_2^* & \cdots & a_n^* \\ a_1 & a_0 & a_1^* & \cdots & a_{n-1}^* \\ a_2 & a_1 & a_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & a_1^* \\ a_n & a_{n-1} & \cdots & a_1 & a_0 \end{bmatrix}$$

$$A_S=\begin{bmatrix} 0 & -a_1^* & -a_2^* & \cdots & -a_n^* \\ a_1 & 0 & -a_1^* & \cdots & -a_{n-1}^* \\ a_2 & a_1 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & -a_1^* \\ a_n & a_{n-1} & \cdots & a_1 & 0 \end{bmatrix}$$

拓普利兹矩阵

- T = toeplitz(r) returns the symmetric Toeplitz matrix
- T = toeplitz(c,r) returns a nonsymmetric Toeplitz matrix with c as its first column and r as its first row.

```
>> r = [1 2 3];
toeplitz(r)
ans =
     1     2     3
     2     1     2
     3     2     1
```

```
>> c = [1  2  3 4];r = [1 5 6];
toeplitz(c,r)
ans =
     1     5     6
     2     1     5
     3     2     1
     4     3     2
```

拓普利兹矩阵

```
>> c = [1+3i 2-5i -1+3i];
r = [1+3i 3-1i -1-2i];
T = toeplitz(c,r)

T =
   1.0000 + 3.0000i   3.0000 - 1.0000i  -1.0000 - 2.0000i
   2.0000 - 5.0000i   1.0000 + 3.0000i   3.0000 - 1.0000i
  -1.0000 + 3.0000i   2.0000 - 5.0000i   1.0000 + 3.0000i
```

汉克矩阵

◊汉克矩阵(Hankel matrix)

任何一条交叉对角线的元素取相同值的特殊矩阵

$$H=\begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_n \\ h_1 & h_2 & h_3 & \cdots & h_{n+1} \\ h_2 & h_3 & h_4 & \cdots & h_{n+2} \\ \vdots & \vdots & \vdots & & \vdots \\ h_n & h_{n+1} & h_{n+2} & \cdots & h_{2n} \end{bmatrix}$$

性质： （1）JH 和 HJ 均为Toeplitz矩阵T；

（2）$(JH)^T = HJ.$

（3）TJ = H

其中J为互换矩阵。

$$\begin{matrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{matrix}$$

汉克矩阵

- H = hankel(c) returns the square Hankel matrix whose first column is c and whose elements are zero below the first anti-diagonal.
- H = hankel(c,r) returns a Hankel matrix whose first column is c and whose last row is r. If the last element of c differs from the first element of r, the last element of c prevails.

```
>> c = 1:3;
>> h = hankel(c)
h =
     1     2     3
     2     3     0
     3     0     0
```

```
>> c = 1:3;
r = 3:6;
h = hankel(c,r)
h =
     1     2     3     4
     2     3     4     5
     3     4     5     6
```

循环矩阵

◊右循环矩阵

$$C_R=(c_{ij}),\text{其中}c_{ij}=\begin{cases}c_{j-i}, & j-i\geq 0\\ c_{n+j-i}, & j-i<0\end{cases}$$

左循环矩阵

$$C_L=(c_{ij}),\text{其中}c_{ij}=\begin{cases}c_{n+1-i-j}, & j+i\leq n+1\\ c_{2n+1-i-j}, & j+i>n+1\end{cases}$$

性质：

（1）右循环矩阵是一特殊的Toeplitz矩阵.

（2）左循环矩阵是一特殊的Hankel矩阵.

（3）循环矩阵可由其第一行（列）元素完全确定，并记为

$$C_R=C_R(c_0,c_1,\cdots,c_{n-1})$$
$$C_L=C_L(c_0,c_1,\cdots,c_{n-1})$$

循环矩阵

创建循环矩阵

```
>> v = [9 1 3 2];
toeplitz([v(1) fliplr(v(2:end))], v)

ans =

     9     1     3     2
     2     9     1     3
     3     2     9     1
     1     3     2     9
```

```
>> hankel([(v(2:end)),v(1)], v)

ans =

     1     3     2     9
     3     2     9     1
     2     9     1     3
     9     1     3     2
```

离散卷积运算

循环矩阵

```
>> x = [1 8 3 2 5];   h = [3 5 2];   r = [x zeros(1,length(h)-1)]
r =     1     8     3     2     5     0     0
>> c = [x(1) zeros(1,length(h)-1)]
c =     1     0     0
>> xConv = toeplitz(c,r)
     1     8     3     2     5     0     0
     0     1     8     3     2     5     0
     0     0     1     8     3     2     5
>>h*xConv
ans =     3    29    51    37    31    29    10
>> conv(x,h)
ans =     3    29    51    37    31    29    10
```

循环矩阵

（4）循环矩阵的线性运算和乘积仍为循环矩阵.

（5）若A,B同为（右或左）循环矩阵，则AB为右循环矩阵.

（6）若A,B为右循环矩阵，则 $AB = BA$

```
>> v = [9 1 3 2];
A=toeplitz([v(1) fliplr(v(2:end))], v)
v = [1 2 3 4];
B=toeplitz([v(1) fliplr(v(2:end))], v)
A =
     9     1     3     2
     2     9     1     3
     3     2     9     1
     1     3     2     9
B =
     1     2     3     4
     4     1     2     3
     3     4     1     2
     2     3     4     1
```

```
>> A*B
ans =
    26    37    40    47
    47    26    37    40
    40    47    26    37
    37    40    47    26
>> B*A
ans =
    26    37    40    47
    47    26    37    40
    40    47    26    37
    37    40    47    26
```

循环矩阵

```
>> v = [9 1 3 2];
>> A=hankel([(v(2:end)),v(1)], v)
A =
     1     3     2     9
     3     2     9     1
     2     9     1     3
     9     1     3     2
>> v = [3 4 5 2];
>> B=hankel([(v(2:end)),v(1)], v)
B =
     4     5     2     3
     5     2     3     4
     2     3     4     5
     3     4     5     2
>> A*B
ans =
    50    53    64    43
    43    50    53    64
    64    43    50    53
    53    64    43    50
```

```
> B*A

ans =

    50    43    64    53
    53    50    43    64
    64    53    50    43
    43    64    53    50
```

矩阵运算

■向量化函数与矩阵化函数

（列）向量化函数（按列拉直）

$$A=\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}_{m\times n} \to vec(A)=\begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \\ \vdots \\ a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix}$$

行向量化函数 $rvec(A)=[a_{11},\cdots,a_{1n},\cdots,a_{m1},\cdots,a_{mn}]$

矩阵化函数

$$unvec_{m,n}(a)=A_{m\times n}=\begin{bmatrix} a_1 & a_{m+1} & \cdots & a_{m(n-1)+1} \\ a_2 & a_{m+2} & \cdots & a_{m(n-1)+2} \\ \vdots & \vdots & & \vdots \\ a_m & a_{2m} & \cdots & a_{mn} \end{bmatrix}$$

Matlab: reshape

矩阵运算

◊矩阵化算子和向量化算子的关系

$$unvec_{m,n}(a) = A_{m\times n} \Rightarrow vec(A_{m\times n}) = a$$

◊向量化算子和行向量化算子的关系

$$rvec(A) = (vec(A^T))^T, rvec(A^T) = (rvec(A))^T$$

◊ $K_{mn} vec(A) = vec(A^T)$

其中 K_{mn} 为交换矩阵（commutation matrix）

$$K_{mn} = \sum_{j=1}^{n} (e_j^T \otimes I_m \otimes e_j)$$

矩阵运算

■Kronecker积

◇ $m\times n$ 矩阵A和 $p\times q$ 矩阵的右Kronecker积

$$A\otimes B=[a_{ij}B]=\begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}_{mp\times nq}$$

◇ $m\times n$ 矩阵A和 $p\times q$ 矩阵的左Kronecker积

$$[A\otimes B]_{left}=[Ab_{ij}]=\begin{bmatrix} Ab_{11} & Ab_{12} & \cdots & Ab_{1q} \\ Ab_{21} & Ab_{22} & \cdots & Ab_{2q} \\ \vdots & \vdots & & \vdots \\ Ab_{p1} & Ab_{p2} & \cdots & Ab_{pq} \end{bmatrix}_{mp\times nq}$$

◇ $vec(ab^T)=b\otimes a$

矩阵运算

K = kron(A,B) returns the Kronecker tensor product of matrices A and B. If A is an m-by-n matrix and B is a p-by-q matrix, then kron(A,B) is an m*p-by-n*q matrix formed by taking all possible products between the elements of A and the matrix B.

```
>> A = eye(4); B = [1 -1;-1 1];
>> K = kron(A,B)
K =
     1    -1     0     0     0     0     0     0
    -1     1     0     0     0     0     0     0
     0     0     1    -1     0     0     0     0
     0     0    -1     1     0     0     0     0
     0     0     0     0     1    -1     0     0
     0     0     0     0    -1     1     0     0
     0     0     0     0     0     0     1    -1
     0     0     0     0     0     0    -1     1
```

```
>> A = [1 2 3; 4 5 6];  B = ones(2);
>> K = kron(A,B)
K =

     1     1     2     2     3     3
     1     1     2     2     3     3
     4     4     5     5     6     6
     4     4     5     5     6     6
```

Kronecker积

Kronecker积的性质

（1） $A\otimes B\neq B\otimes A$

（2） $AB\otimes CD=(A\otimes C)(B\otimes D)$

（3） $A\otimes(B\pm C)=A\otimes B\pm A\otimes C$

$$(B\pm C)\otimes A=B\otimes A\pm C\otimes A$$

```
>> A=randi(10,3);
>> B=randi(10,3)
>> C=randi(10,3);
>> D=randi(10,3)
>> g = abs(kron(A*B,C*D)-kron(A,C)*kron(B,D));
>> sum(g(:))
ans =     0
```

Kronecker积

Kronecker积的性质

（4）特别地，若A和B是可逆的正方矩阵，则

$$(A\otimes B)^{-1}=A^{-1}\otimes B^{-1}$$

```
>> g = abs(inv(kron(A,B))-kron(inv(A),inv(B)));
>> sum(g(:))
ans =   3.4326e-16
```

（5） $(A\otimes B)^T=A^T\otimes B^T$； $(A\otimes B)^H=A^H\otimes B^H$

（6） $rank(A\otimes B)=rank(A)rank(B)$

$$A\otimes B=[a_{ij}B]=\begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}_{mp\times nq}$$

Kronecker积

（7）对于$A_{m\times m}, B_{n\times n}$，有 $\det(A\otimes B)=(\det(A))^m(\det(B))^n$

```
>> det(kron(A,B))
ans =  -6.5548e+13
>> det(A)^3*det(B)^3
ans =  -6.5548e+13
```

（8）$tr(A\otimes B)=tr(A)tr(B)$

（9）对于矩阵$A_{m\times n}, B_{m\times n}, C_{p\times q}, D_{p\times q}$，有

$$(A+B)\otimes(C+D)=A\otimes C+A\otimes D+B\otimes C+B\otimes D$$

$$A\otimes B=[a_{ij}B]=\begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}_{mp\times nq}$$

Kronecker积

定理 令 $A_{m\times p}, B_{p\times q}, C_{q\times n}$，则

$$vec(ABC) = (C^T \otimes A)vec(B)$$

特别地，当其中一个矩阵为单位矩阵时，有

$$\begin{aligned} vec(BC) &= (C^T \otimes I_m)vec(B) \\ &= (C^T \otimes B)vec(I_q) \\ &= (I_n \otimes B)vec(C) \end{aligned}$$

应用1： 矩阵方程 $AXB = D$ 的求解.

应用2： 矩阵方程 $AX + XB = C$ 的求解.

Kronecker积

广义Kronecker积

给定N个 $m \times r$ 矩阵 $A_i, i=1,2,\ldots,N$,它们组成矩阵组 $\{A\}_N$.该矩阵组与 $N \times l$ 矩阵B的Kronecker积称为广义Kronecker积，定义为

$$\{A\}_N \otimes B = \begin{bmatrix} A_1 \otimes b_1 \\ A_2 \otimes b_2 \\ \vdots \\ A_N \otimes b_N \end{bmatrix}$$

式中, b_i 是矩阵的第i个行向量.

广义Kronecker积在滤波器组的分析、Haar变换和Hadamard变换的快速算法的推导中有着重要的作用[386].

Hadamard积

■Hadamard积

定义： $m\times n$ 矩阵 $A=[a_{ij}]$ 与 $m\times n$ 矩阵 $B=[b_{ij}]$ 的**Hadamard**积记作 A ⊙ B，并定义为

$$A \odot B = [a_{ij}b_{ij}]_{m\times n}$$

◊Hadamard积也称为**Schur**积或对应元素乘积（**elementwise product**）.

性质：

（1）若A，B均为 $m\times n$ 矩阵，则

$$A \odot B = B \odot A$$

$$(A\odot B)^T = A^T \odot B^T$$

$$(A\odot B)^H = A^H \odot B^H$$

$$(A\odot B)^* = A^* \odot B^*$$

（2）若 $A\in C^{m\times n}$ ，则 $A\odot O_{m\times n} = O_{m\times n}\odot A = O_{m\times n}$

（3）若c为常数，则 $c(A\odot B) = (cA)\odot B = A\odot(cB)$

矩阵直和

■矩阵直和

定义：$m\times m$ 矩阵与 $n\times n$ 矩阵的直和定义为

$$A\oplus B=\begin{bmatrix} A & O_{m\times n} \\ O_{n\times m} & B \end{bmatrix}$$

性质：

（1）若c为常数，则 $c(A\oplus B)=cA\oplus cB$.

（2）一般情况下，$A\oplus B\neq B\oplus A$

（3）

$$(A\oplus B)^{*}=A^{*}\oplus B^{*}$$

$$(A\oplus B)^{T}=A^{T}\oplus B^{T}$$

$$(A\oplus B)^{H}=A^{H}\oplus B^{H}$$

$$(A\oplus B)^{-1}=A^{-1}\oplus B^{-1}$$

矩阵直和

$$(4) A \oplus (B \oplus C) = (A \oplus B) \oplus C = A \oplus B \oplus C$$

$$(5) (A \pm B) \oplus (C \pm D) = (A \oplus C) \pm (B \oplus D)$$

$$(A \oplus C)(B \oplus D) = AB \oplus CD$$

(6)

$$\det(\oplus_{i=1}^{N} A_i) = \prod_{i=1}^{N} \det(A_i)$$

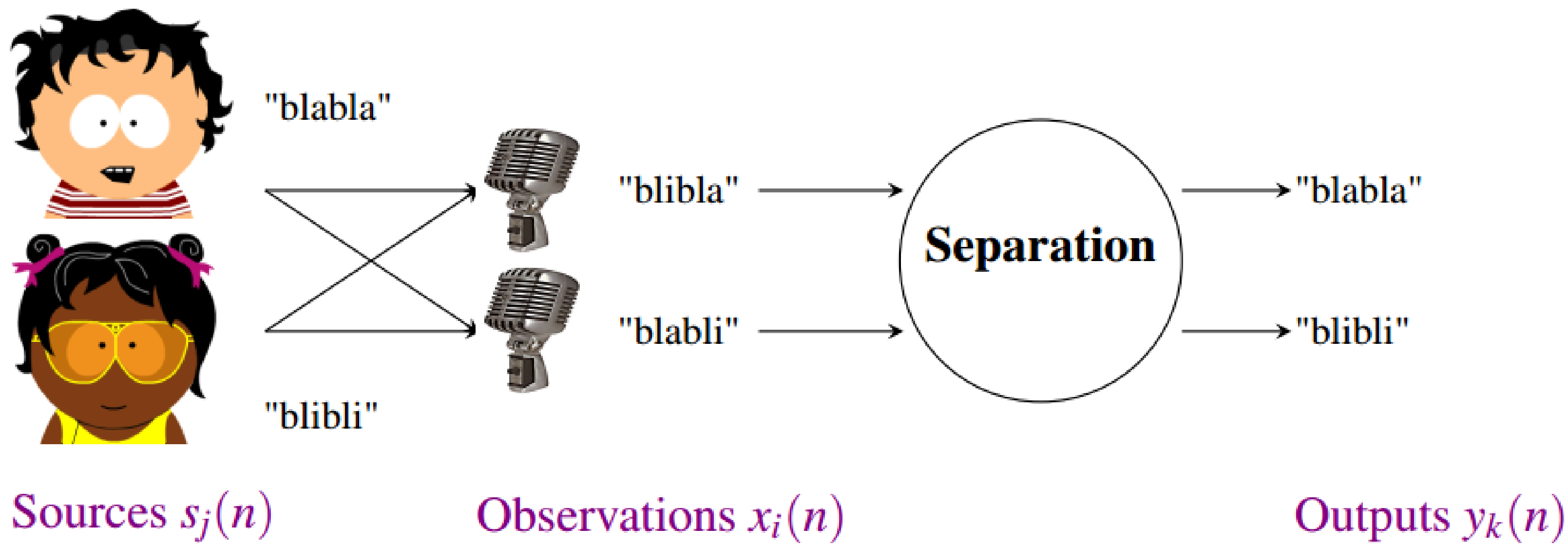
$$tr(\oplus_{i=1}^{N} A_i) = \sum_{i=1}^{N} tr(A_i)$$

$$rank(\oplus_{i=1}^{N} A_i) = \sum_{i=1}^{N} rank(A_i)$$

（7）若A,B分别为$m \times m$ 和 $n \times n$ 正交矩阵，则 $A \oplus B$ 为 $(m+n) \times (m+n)$ 正交矩阵.

应用-盲源分离

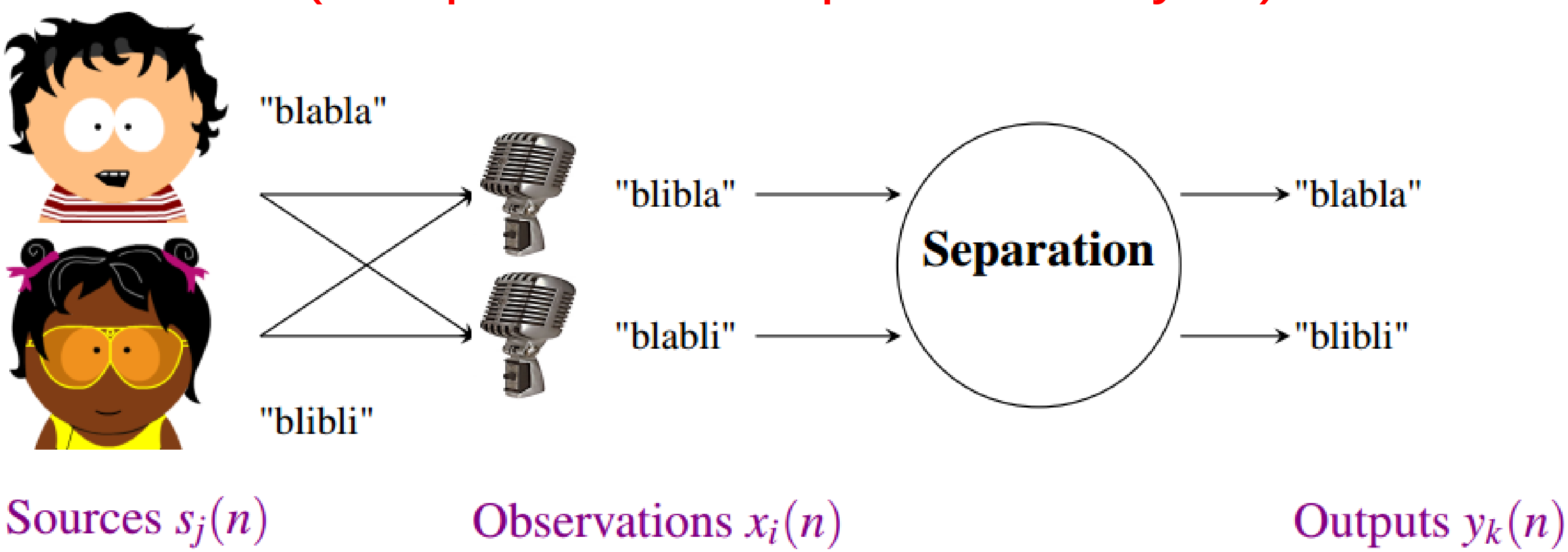
- 当传输信道不可知或难以预测时，如何从接收到的信号中筛选出自己感兴趣的信号及需要的真实信息。
- 它起源于“鸡尾酒会”问题在嘈杂的鸡尾酒会场，有多个人的说话声，还混杂着背景音乐和其他噪声。人们仍然可以根据自己的兴趣听到某个人说话的声音。



From Matthieu Puigt

应用-盲源分离

- 从观测数据中对源信号进行估计，寻找一种合适的滤波器或逆系统，使得输出的信号尽可能地接近源信号。
- 术语“盲的”有两重含义：(1)源信号不能被观测；(2)源信号如何混合是未知的。
- 独立分量分析(Independent Component Analysis)理论



From Matthieu Puigt

应用-盲源分离

- **线性瞬时混合模型:**
- 假设N个源信号经过线性瞬时混合被M个传感器接收，则每个观测信号是这N个信号的一个线性组合。

$$x_j(t) = \sum_{i=1}^{N} a_{ji} s_i(t) \qquad \mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)$$

- 瞬时混合盲源分离的目标是找到一个$N \times M$ 解混矩阵W使得 $\mathbf{y}(t) = \mathbf{W}\mathbf{x}(t)$ 为源信号的估计。
- 考虑加性噪声：

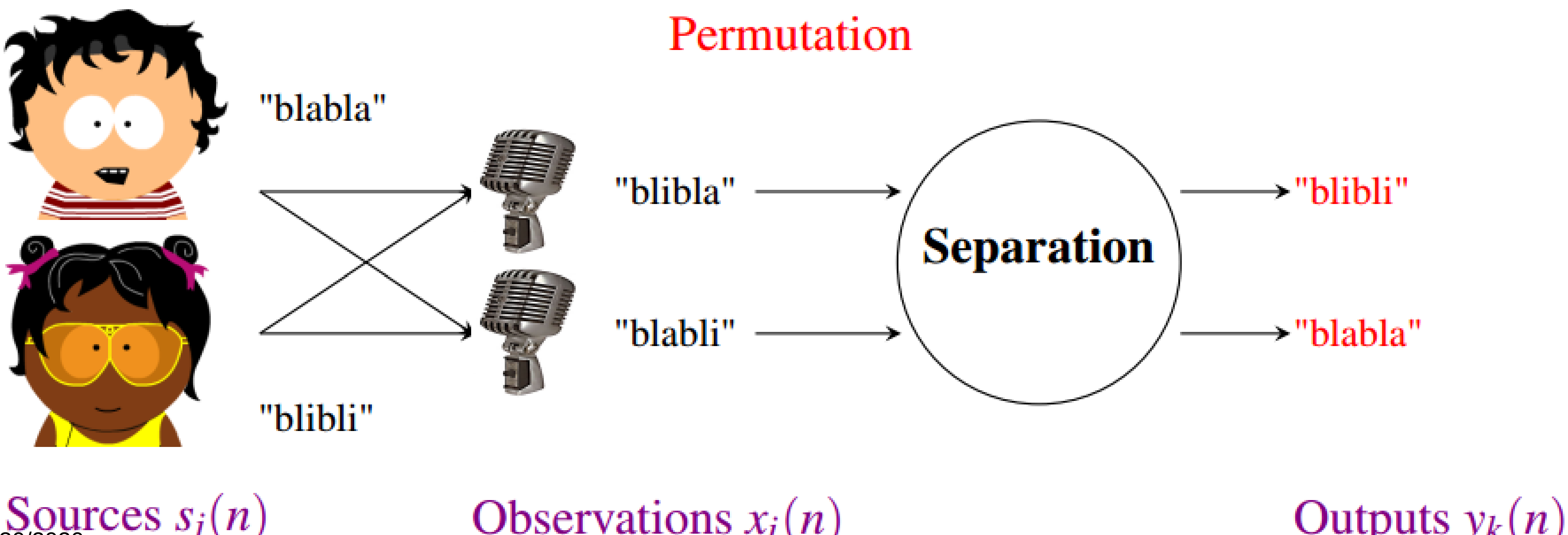
$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{n}(t)$$

应用-盲源分离

- **线性瞬时混合模型:**

$$\mathbf{x}(t) = GA\hat{\mathbf{s}}(t)$$

- G = PD 为广义置换矩阵。P为顺序的不确定性，D为尺度的不确定性 。



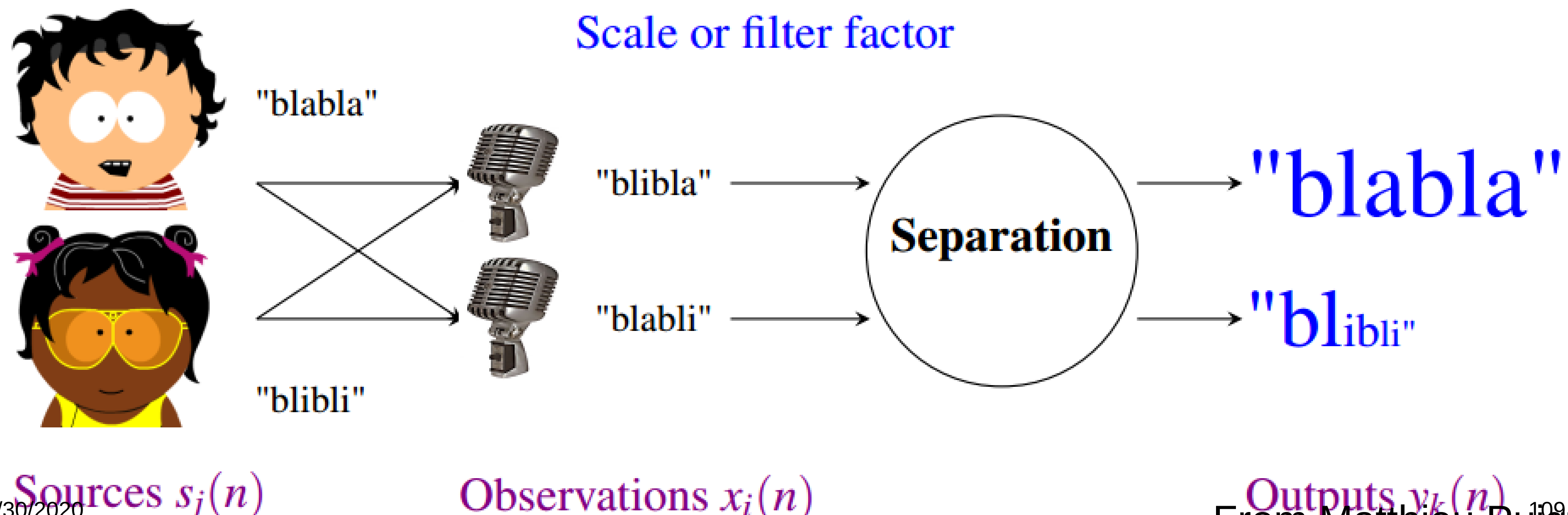
From Matthieu Puigt

应用-盲源分离

- **线性瞬时混合模型:**

$$x(t) = GA\hat{s}(t)$$

- G = PD 为广义置换矩阵。P为顺序的不确定性，D为尺度的不确定性 。

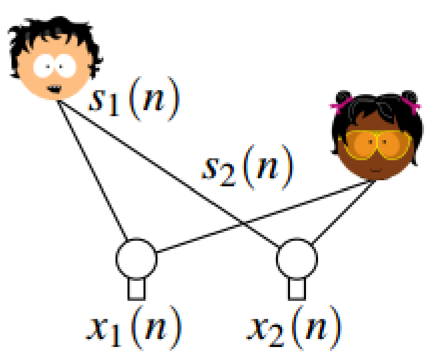


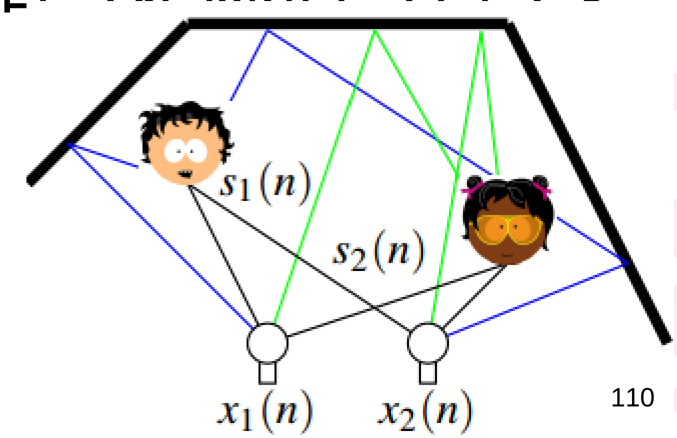
应用-盲源分离

- **线性卷积混合模型：**

① 实际中每一个源信号**不会同时到达**所有的传感器，每一个传感器对不同的源延时不同，延时值的大小取决于传感器与源信号间的相对位置以及信号的传播速度；

② 源信号到达传感器是经过**多径传播**的，假设信号是线性组合的，则从传感器观测到的信号是源信号各个延时值的线性组合。





From Matthieu Puigt

应用-盲源分离

- **线性卷积混合模型：**实际系统中，传感器接收到的信号往往是源信号经不同时延的线性组合。

$$x_j(t)=\sum_{i=1}^{N}a_{ji}*s_i(t)=\sum_{i=1}^{N}\sum_{\tau=-\infty}^{\infty}a_{ji,\tau}s_i(t-\tau),\quad j=1,2,\cdots,M$$

$$\boldsymbol{x}(t)=\sum_{\tau=-\infty}^{\infty}\boldsymbol{A}_{\tau}\boldsymbol{s}(t-\tau)$$

$$\boldsymbol{y}(t)=\sum_{p=-\infty}^{+\infty}\boldsymbol{W}_{p}\boldsymbol{x}(t-p)$$

应用-盲源分离

- **线性卷积混合模型：**

$$x(t)=\sum_{\tau=-\infty}^{\infty} A_\tau s(t-\tau) \qquad y(t)=\sum_{p=-\infty}^{+\infty} W_p x(t-p)$$

- 在 z 变换域中输入、输出系统可表示为

$$X(z)=A(z)S(z)$$

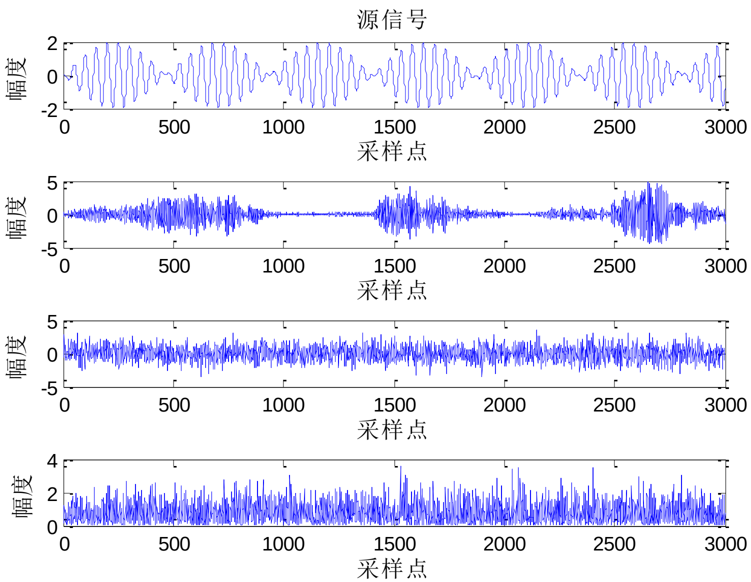
$$Y(z)=W(z)X(z)=W(z)A(z)S(z)=C(z)S(z)$$

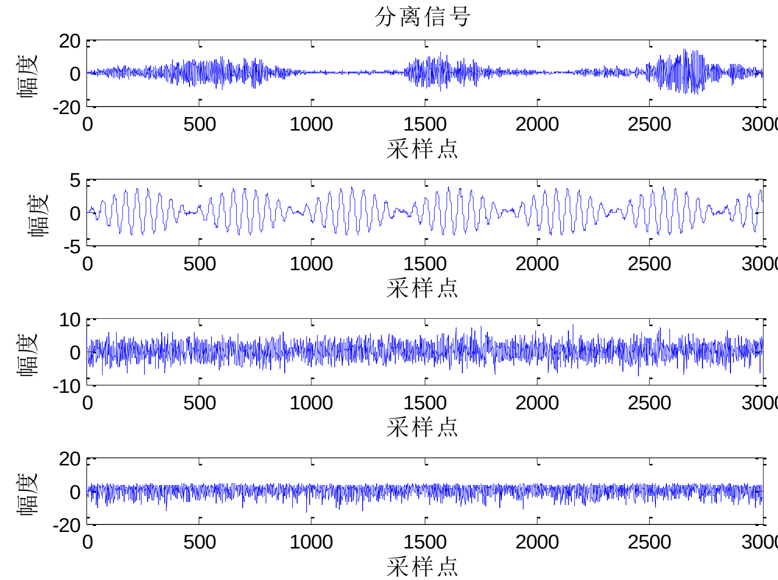
$$W(z)=\sum_{p=-\infty}^{\infty} W_p z^{-p}, \quad A(z)=\sum_{p=-\infty}^{\infty} A_p z^{-p}, \quad C(z)=W(z)A(z)$$

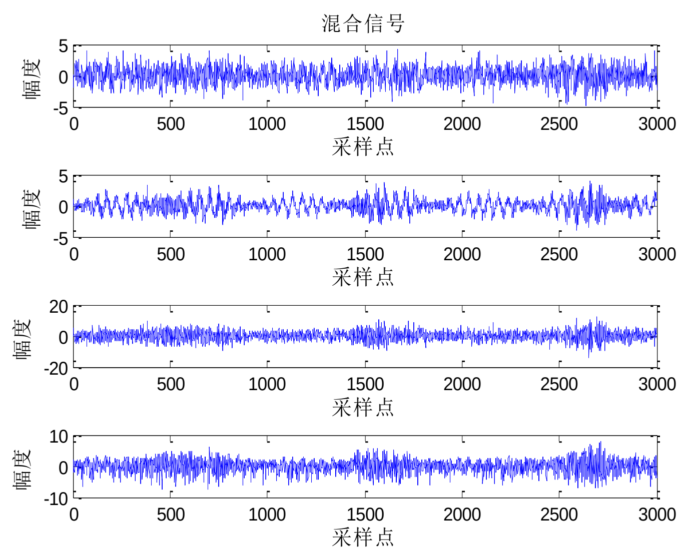
- 当 y 为 s 的估计时有 $C(z)=PD(z)$

其中 P 为任意置换矩阵，D为非奇异对角矩阵

应用-盲源分离


源信号
幅度
采样点
0
500
1000
1500
2000
2500
3000
(a)


分离信号
幅度
采样点
0
500
1000
1500
2000
2500
3000
(c)


混合信号
幅度
采样点
0
500
1000
1500
2000
2500
3000
(b)

应用-盲源分离

- https://www.cs.helsinki.fi/u/ahyvarin/papers/fastica.shtml

The FastICA algorithm

[This is probably the most widely used algorithm for performing independent component analysis, a variant of factor analysis that is completely identifiable unlike classical methods, and able to perform blind source separation.]

FastICA package for Matlab and other systems

A. Hyvärinen. **Fast and Robust Fixed-Point Algorithms for Independent Component Analysis**. *IEEE Transactions on Neural Networks* 10(3):626-634, 1999.
pdf
[The fundamental paper on the FastICA algorithm, which is computationally very efficient yet statistically robust.]

- **https://github.com/vsubhashini/ica**

应用-盲源分离

A "generic" problem

Many applications: biomedical, audio processing and audio coding, telecommunications, astrophysics, image classification, underwater acoustics, finance, etc.

作 业

矩阵论与工程应用，p.46

① 2.11

② 2.12

③ 2.14

作 业

2.11 [6,p.68] 矩阵 $\boldsymbol{A}=[a_{ij}](i,j=1,2,3,4)$ 称为 Lorentz 矩阵，若变换 $\boldsymbol{x}=\boldsymbol{A}\boldsymbol{y}$ 使得二次型 $Q(\boldsymbol{x})=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}=x_1^2-x_2^2-x_3^2-x_4^2$ 不变，即 $Q(\boldsymbol{x})=Q(\boldsymbol{y})$。证明：两个 Lorentz 矩阵的乘积仍然为 Lorentz 矩阵。

2.12 [6,p.265] $n\times n$ 矩阵 $\boldsymbol{M}$ 称为 Markov 矩阵，若其元素满足条件 $m_{ij}\geqslant 0$, $\sum_{i=1}^{n}m_{ij}=1, j=1,2,\cdots,n$。假定 $\boldsymbol{P}$ 和 $\boldsymbol{Q}$ 均为 Markov 矩阵，证明：

(1) 对于常数 $0\leqslant\lambda\leqslant 1$，矩阵 $\lambda\boldsymbol{P}+(1-\lambda)\boldsymbol{Q}$ 是 Markov 矩阵。

(2) 矩阵乘积 $\boldsymbol{PQ}$ 也为 Markov 矩阵。

2.14 满足 $\boldsymbol{A}\boldsymbol{A}^{\mathrm{H}}=\boldsymbol{A}^{\mathrm{H}}\boldsymbol{A}$ 的正方矩阵 $\boldsymbol{A}$ 称为正规矩阵。证明：若 $\boldsymbol{A}$ 为正规矩阵，则 $\boldsymbol{A}-\lambda\boldsymbol{I}$ 也为正规矩阵。

谢　　谢！