Stochastic Signal Processing

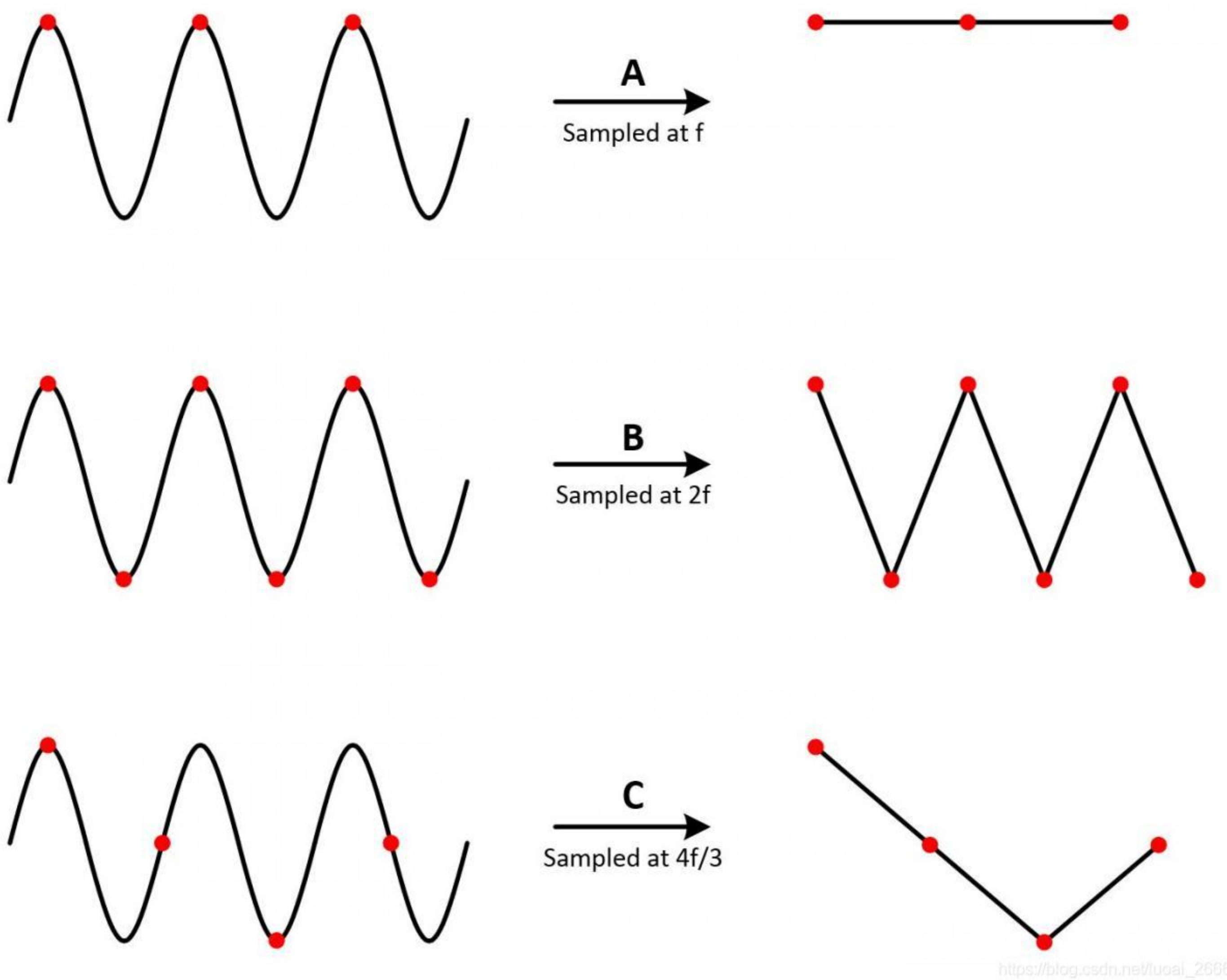
Lesson 13-2: Spectrum Estimation and Detection

Weize Sun

Frequency Estimation Problem

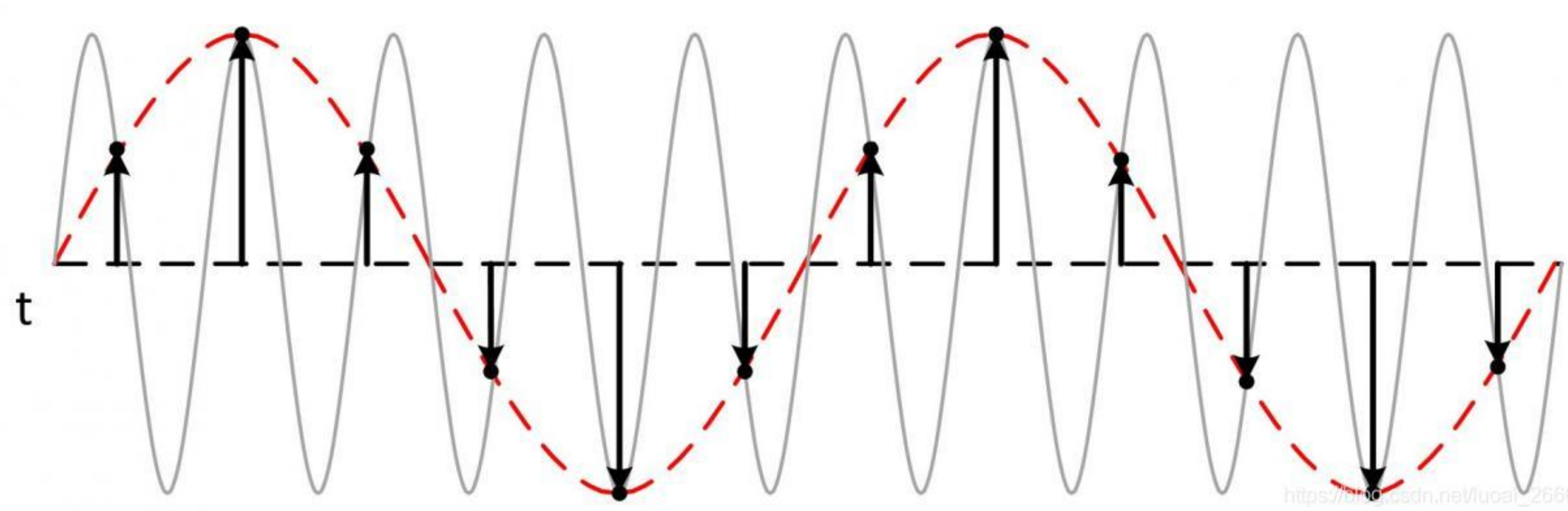
- A source send out signal.
- A sensor record the signal $x(t) = Ae^{j(\omega t+\theta)}$, where $\omega = 2\pi \frac{f}{f_s}$, where f is the frequency to be estimated. f_S is the sampling frequency.
- According to the Nyquist Sampling theory, $f_s \geq 2f$. The larger the f_s, the higher the cost of the sampling system, therefore, we usually set $f_s = 2f$, and have $\omega \in [0,\ \pi]$ (positive frequency only)
- In some applications, we can assume that there are negative frequency, then $\omega \in [-\pi,\ \pi]$, or we can also write it as $\omega \in [0,\ 2\pi]$ ($[\pi,\ 2\pi]$ refers to $[-\pi,\ 0]$)
- In the remainder, we will tackle the ω directly.

Frequency Estimation Problem



Frequency Estimation Problem

- if $f_s < 2f$



Frequency Estimation Problem

$$x(t) = Ae^{j(\omega t+\theta)}$$

- Usually, it is a discrete time system. Therefore we set $t = 0,1,2, \ldots, \infty$
- And change t to n:

$$x(n) = Ae^{j(\omega n+\theta)}$$
$$n = 0,1,2, \ldots, N-1$$

Where N is the number of samples.

- And we might have noise:

$$y(n) = Ae^{j(\omega n+\theta)} + q(n)$$

Where $q(n)$ is i.i.d Gaussian noise.

Frequency Estimation Problem

$$y(n) = Ae^{j(\omega n+\theta)} + q(n)$$

- There are three parameters unknown: A, ω and θ.
- The target is estimate A, ω and θ from the received signal $y(n)$, $n = 0,1,2,\ldots,N-1$

Frequency Estimation Problem

$$y(n) = Ae^{j(\omega n+\theta)} + q(n)\ ,\ n = 0,1,2,\dots,N-1$$

- Note that $Ae^{j(\omega n+\theta)} = Ae^{j\theta} \times e^{j\omega n} = \alpha e^{j\omega n}$
- Assume that we already get ω, then:

$$\boldsymbol{Y} = \alpha\boldsymbol{S} + \boldsymbol{Q}, \text{ or } \boldsymbol{Y} \approx \alpha\boldsymbol{S}$$

$$\boldsymbol{Y} = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix};\ \boldsymbol{S} = \begin{bmatrix} e^{j\omega 0} \\ e^{j\omega 1} \\ \vdots \\ e^{j\omega(N-1)} \end{bmatrix};\ \boldsymbol{Q} = \begin{bmatrix} q(0) \\ q(1) \\ \vdots \\ q(N-1) \end{bmatrix}$$

- Then $\alpha = (\boldsymbol{S}^H\boldsymbol{S})^{-1}\boldsymbol{S}^H\boldsymbol{Y}$, α can be calculated from a linear equation

Frequency Estimation Problem

$$y(n) = Ae^{j(\omega n+\theta)} + q(n)\ , n = 0,1,2, \ldots, N-1$$
$$Ae^{j(\omega n+\theta)} = Ae^{j\theta} \times e^{j\omega n} = \alpha e^{j\omega n}$$

$$\boldsymbol{Y} = \alpha\boldsymbol{S} + \boldsymbol{Q}, \text{ or } \boldsymbol{Y} \approx \alpha\boldsymbol{S}$$

- Then $\alpha = (\boldsymbol{S}^H\boldsymbol{S})^{-1}\boldsymbol{S}^H\boldsymbol{Y}$, α can be calculated from a linear equation; and

$$A = |\alpha|, \theta = \angle\alpha$$

- A and θ can be calculated from a linear equation, thus called linear parameter
- ω, instead, should be search from $[-\pi, \pi]$ and thus called non-linear parameter

Frequency Estimation Problem

$$y(n) = Ae^{j(\omega n+\theta)} + q(n)\ ,\ n = 0,1,2,\ldots,N-1$$
$$Ae^{j(\omega n+\theta)} = Ae^{j\theta} \times e^{j\omega n} = \alpha e^{j\omega n}$$

$$\boldsymbol{Y} = \alpha\boldsymbol{S} + \boldsymbol{Q}, \text{ or } \boldsymbol{Y} \approx \alpha\boldsymbol{S}$$
$$\alpha = (\boldsymbol{S}^H\boldsymbol{S})^{-1}\boldsymbol{S}^H\boldsymbol{Y}$$

$A = |\alpha|$, $\theta = \angle\alpha$: linear parameter

ω: non-linear parameter

- Therefore, we usually first estimate ω then $A = |\alpha|$, $\theta = \angle\alpha$
- We can use the periodogram

Frequency Estimation Problem

code1

```
clear
clc

N = 100;
F_sam_freq=1000;   %采样频率
F_freq = 200;
omega = F_freq/F_sam_freq*2*pi;
SNR = 10;
Amplitude_A = 1;
n_array=[0:N-1]';
FFT_length = 4096;
```

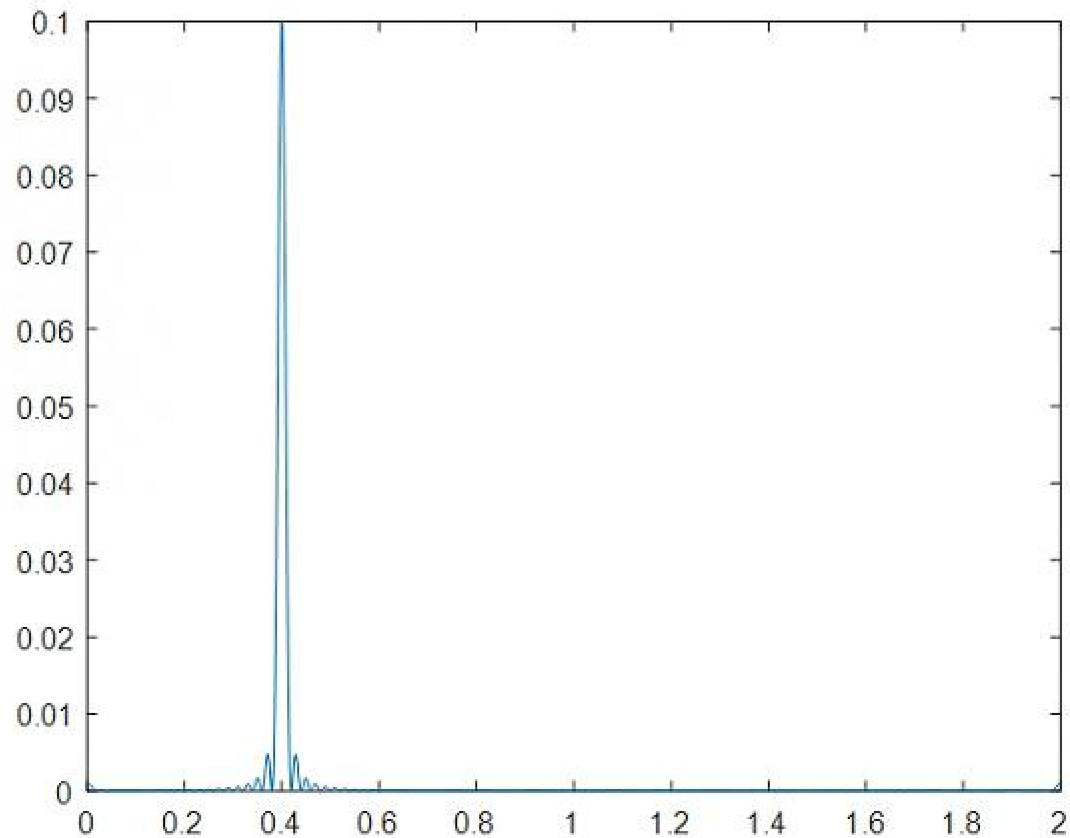
```
Theta = rand(1)*2*pi;
X_signal = Amplitude_A*exp(1j*(omega*n_array+Theta));
Noise_sigma2 = Amplitude_A^2 / [10^(SNR/10)];
Noise = sqrt(Noise_sigma2) * randn(N, 1) .* exp(1j*rand(N, 1)*2*pi);
Y_receive = X_signal + Noise_sigma2;
window = boxcar(N);      %矩形窗
[Peri_Y, f_Y] = periodogram(Y_receive,window,FFT_length,F_sam_freq);
[Max_value, Max_index] = max(Peri_Y);

F_freq_estimate = (Max_index-1)/FFT_length*F_sam_freq;
omega_estimate  = (Max_index-1)/FFT_length*2*pi;
```

```
figure(1)
X_plot = [0:FFT_length-1]/FFT_length*2;
plot(X_plot, Peri_Y)
```

Frequency Estimation Problem

- The estimation is correct, but cannot reach 200Hz even in noise free case.
- The ‘FFT_length = 4096’ is called ‘grid No.’, the larger the ‘grid No.’, the higher the accuracy, but the longer to compute the periodogram.

```
F_freq_estimate =

  199.9512

>> omega_estimate

omega_estimate =

    1.2563
```



Frequency Estimation Problem

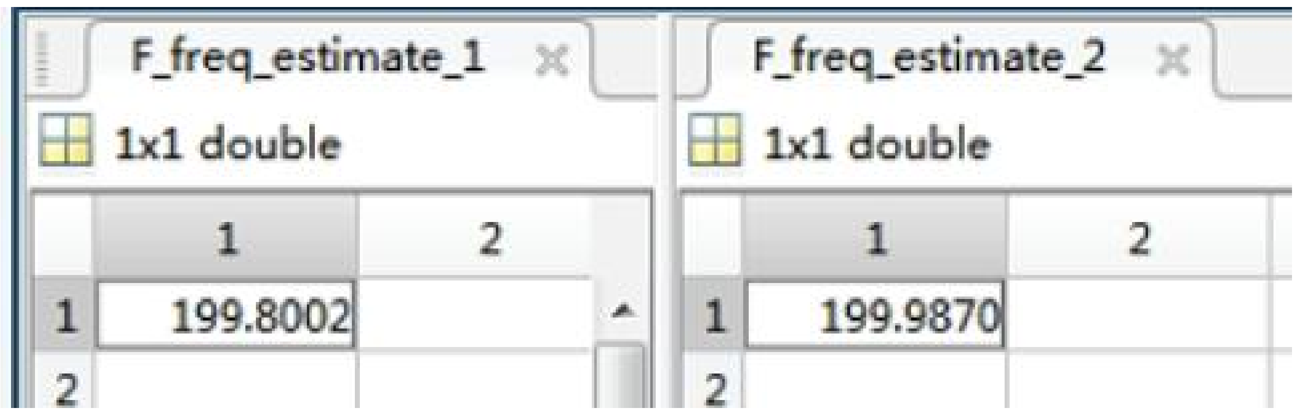
- The estimation is correct, but cannot reach 200Hz even in noise free case.
- The 'FFT_length = 4096' is called 'grid No.', the larger the 'grid No.', the higher the accuracy, but the longer to compute the periodogram.
- We can test it by

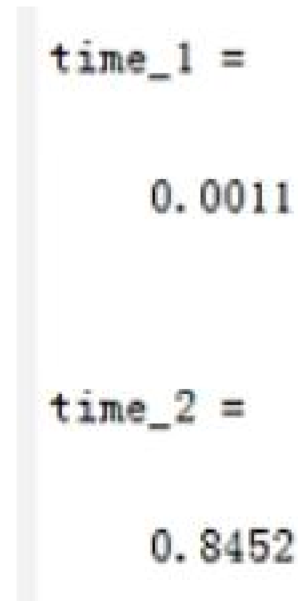
```
%% test of computational time of 'periodogram'
length_1 = 1001;
length_2 = 10000001;
tic
[Peri_Y, f_Y] = periodogram(Y_receive,window,length_1,F_sam_freq);
time_1 = toc;
[Max_value, Max_index] = max(Peri_Y);
F_freq_estimate_1 =  (Max_index-1)/length_1*F_sam_freq;

tic
[Peri_Y, f_Y] = periodogram(Y_receive,window,length_2,F_sam_freq);
time_2 = toc;
[Max_value, Max_index] = max(Peri_Y);
F_freq_estimate_2 =  (Max_index-1)/length_2*F_sam_freq;
```

code1

Frequency Estimation Problem

- The estimation is correct, but cannot reach 200Hz even in noise free case.
- The ‘FFT_length = 4096’ is called ‘grid No.’, the larger the ‘grid No.’, the higher the accuracy, but the longer to compute the periodogram.
- We can test it by





Frequency Estimation Problem

- The estimation is correct, but cannot reach 200Hz even in noise free case.
- The ‘FFT_length = 4096’ is called ‘grid No.’, the larger the ‘grid No.’, the higher the accuracy, but the longer to compute the periodogram.
- We can also set ‘FFT_length = 1024’, then use ‘bisection method'(二分法) to do the accurate estimation.
- However, such method requires a ‘search’, therefore referred to as ‘grid search’ method.
- Is there other methods?

Frequency Estimation Problem

$$y(n) = x(n) + q(n) = Ae^{j(\omega n+\theta)} + q(n)$$

- We have $x(n) = x(n-1) \times e^{j\omega}$, or says,

$$y(n) \approx y(n-1) \times e^{j\omega}$$

In matrix form: $\boldsymbol{Y}_1 \approx \beta \boldsymbol{Y}_2, \quad \beta = e^{j\omega}$

$$\boldsymbol{Y}_1 = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N-1) \end{bmatrix}; \boldsymbol{Y}_2 = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-2) \end{bmatrix}$$

$$\beta = (\boldsymbol{Y}_2^H \boldsymbol{Y}_2)^{-1} \boldsymbol{Y}_2^H \boldsymbol{Y}_1$$

$$\omega = \angle\beta$$

Let’s test it!

Frequency Estimation Problem

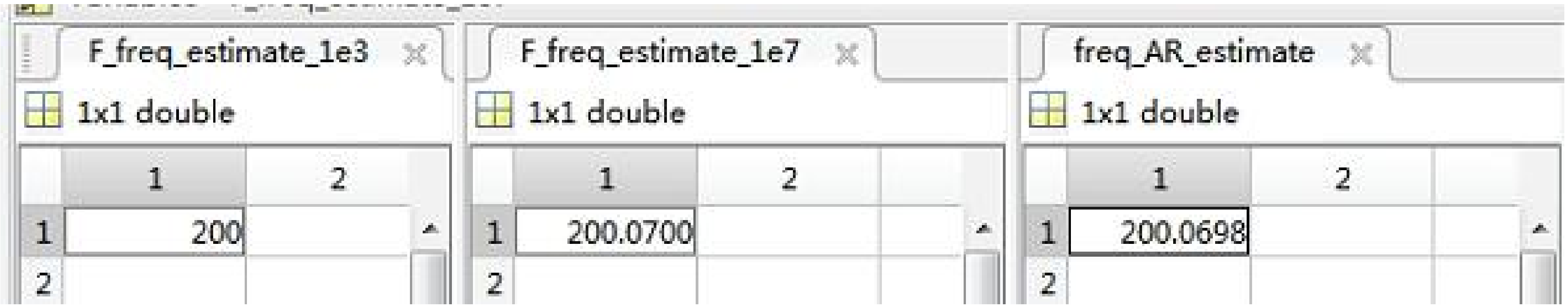
code2

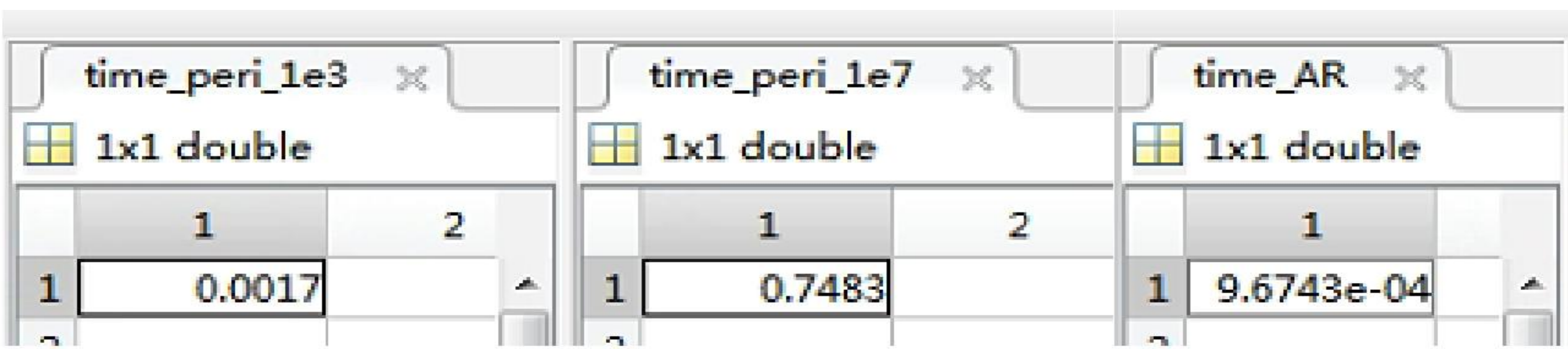
```
%% the Auto Regressive (AR) method
Y_r1 = Y_receive(2:end);
Y_r2 = Y_receive(1:end-1);
tic
beta_est = (Y_r2'*Y_r2)\Y_r2'*Y_r1;
time_AR = toc;
omega_AR_estimate = angle(beta_est);
freq_AR_estimate = omega_estimate/2/pi*F_sam_freq;
```

Frequency Estimation Problem

Frequency 200.07Hz, SNR = 40dB

When noise is small, AR method is fastest almost the same accuracy comparing to periodogram with 'grid No.' 10^7

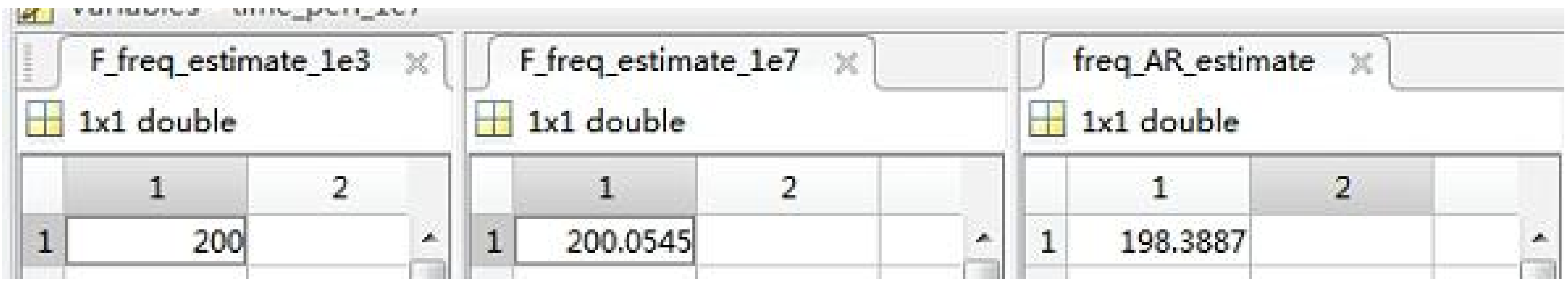




Frequency Estimation Problem

Frequency 200.07Hz, SNR = 10dB

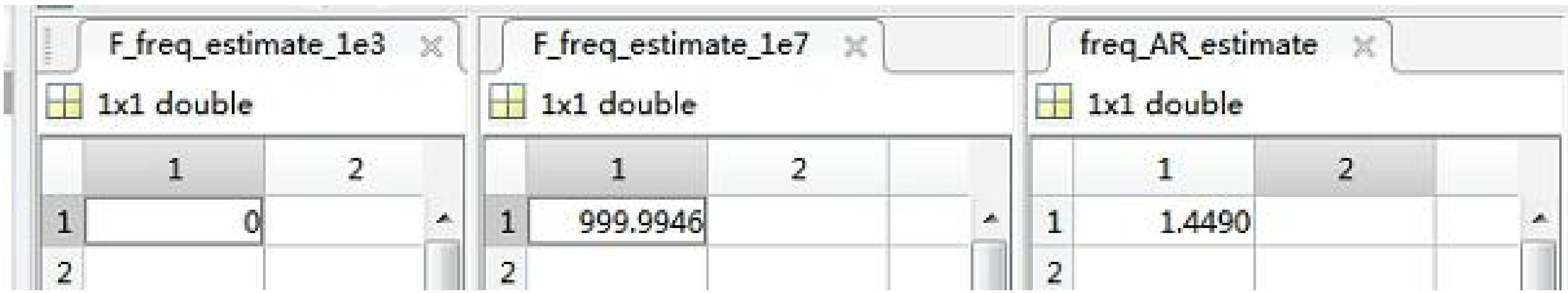
When noise is large, the accuracy of AR method is poor comparing to periodogram even with ‘grid No.’ 10^3



Frequency Estimation Problem

Frequency 200.07Hz, SNR = -10dB

When noise is very large, all methods fail



You can test the methods under different ω and/or SNR, and do the statistics of ‘MSE’ or ‘probability of success’ by yourself

Hilbert transforms（回顾）

- A system with system function

$$H(\omega) = -j\,\mathrm{sgn}\,\omega = \begin{cases} -j & \omega > 0 \\ j & \omega < 0 \end{cases}, \quad \text{where } \mathrm{sgn}\,\omega = \begin{cases} 1 & \omega > 0 \\ -1 & \omega < 0 \end{cases}$$

is called a **quadrature filter** (正交滤波器). The corresponding impulse response equals $1/\pi t$.

- $H(\omega)$ is all-pass with $-90°$ phase shift; hence its response to $\cos\omega t$ equals $\cos(\omega t - 90°) = \sin\omega t$ and its response to $\sin\omega t$ equals $\sin(\omega t - 90°) = -\cos\omega t$.

- The response of a quadrature filter to a real process $\mathbf{x}(t)$ is denoted by $\hat{\mathbf{x}}(t)$ and it is called the Hilbert transform of $\mathbf{x}(t)$. Thus

$$\hat{\mathbf{x}}(t) = \mathbf{x}(t) * \frac{1}{\pi t} = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{\mathbf{x}(\alpha)}{t-\alpha} d\alpha$$

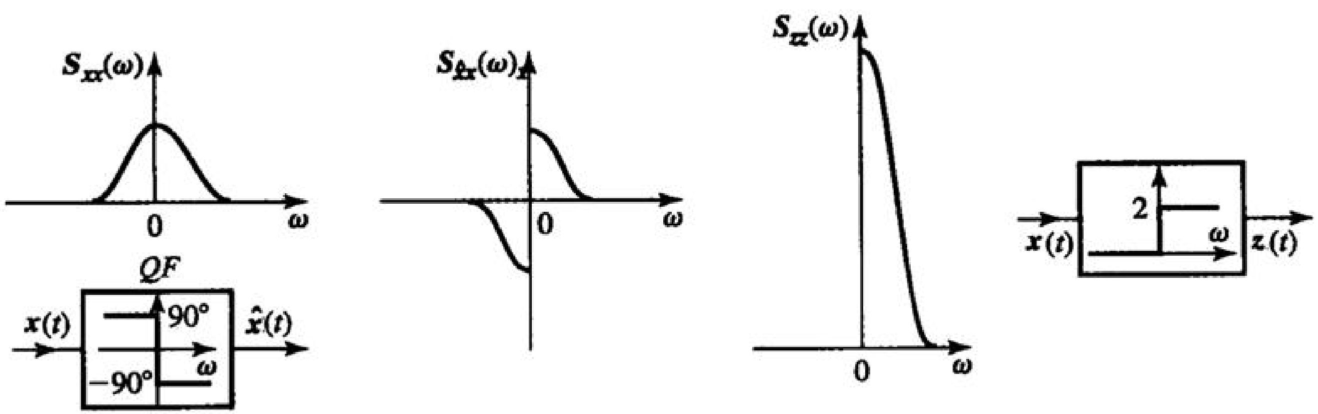
Hilbert transforms（回顾）

- The response of a quadrature filter to a real process $\mathbf{x}(t)$ is denoted by $\hat{\mathbf{x}}(t)$ and it is called the Hilbert transform of $\mathbf{x}(t)$. Thus

$$\hat{\mathbf{x}}(t) = \mathbf{x}(t) * \frac{1}{\pi t} = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{\mathbf{x}(\alpha)}{t-\alpha} d\alpha$$

$$S_{X\hat{X}}(\omega) = jS_X(\omega)\mathrm{sgn}\omega = -S_{\hat{X}X}(\omega)$$

$$S_{\hat{X}}(\omega) = S_X(\omega)$$



Hilbert transforms（回顾）

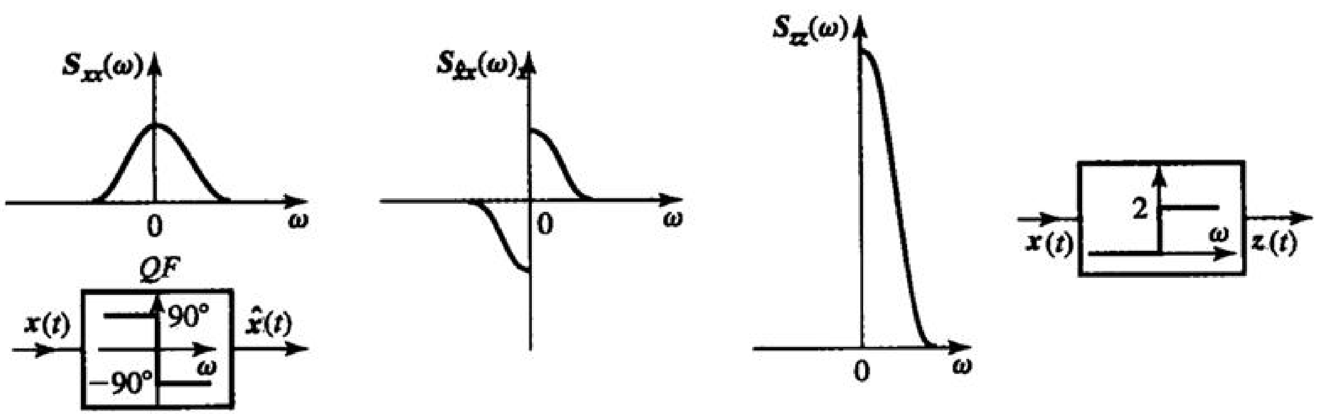
- The complex process $\mathbf{z}(t) = \mathbf{x}(t) + j\hat{\mathbf{x}}(t)$ is calked the analytic signal associated with $\mathbf{x}(t)$. Clearly, $\mathbf{z}(t)$ is the response of the system

$$1 + j(-j\text{sgn}\,\omega) = 2U(\omega)$$

- with input $\mathbf{x}(t)$. Hence:

$$S_Z(\omega) = 4S_X(\omega)U(\omega) = 2S_X(\omega) + 2jS_{\hat{x}x}(\omega)$$

$$R_Z(\tau) = 2R_X(\tau) + 2jR_{\hat{X}X}(\tau)$$



Frequency Estimation Problem

$$y(n) = x(n) + q(n)\ ,\ n = 0,1,2,\ldots,N-1$$
$$x(n) = Acos(\omega n + \theta)$$

$$x(n) \neq \alpha x(n-1) \text{ for any } \alpha$$

- Is there other methods?
- The Hilbert transform of $cos(\omega n + \theta)$ is $sin(\omega n + \theta)$, and Hilbert transform of $sin(\omega n + \theta)$ is $-cos(\omega n + \theta)$. (You can try to prove it yourself)

Frequency Estimation Problem

$$y(n) = x(n) + q(n)\ ,\ n = 0,1,2,\ldots,N-1$$
$$x(n) = A cos(\omega n + \theta)$$

$$x(n) \neq \alpha x(n-1) \text{ for any } \alpha$$
$$H\{y(n)\} = H\{x(n)\} + H\{q(n)\}$$
$$H\{x(n) = A cos(\omega n + \theta)\} = A sin(\omega n + \theta)$$

Define

$$y'(n) = y(n) + j * H\{y(n)\} = x'(n) + q'(n)$$

then

$$x'(n) = A\,\{cos(\omega n + \theta) + j sin(\omega n + \theta)\}$$
$$= Ae^{j(\omega n + \theta)}$$

Frequency Estimation Problem

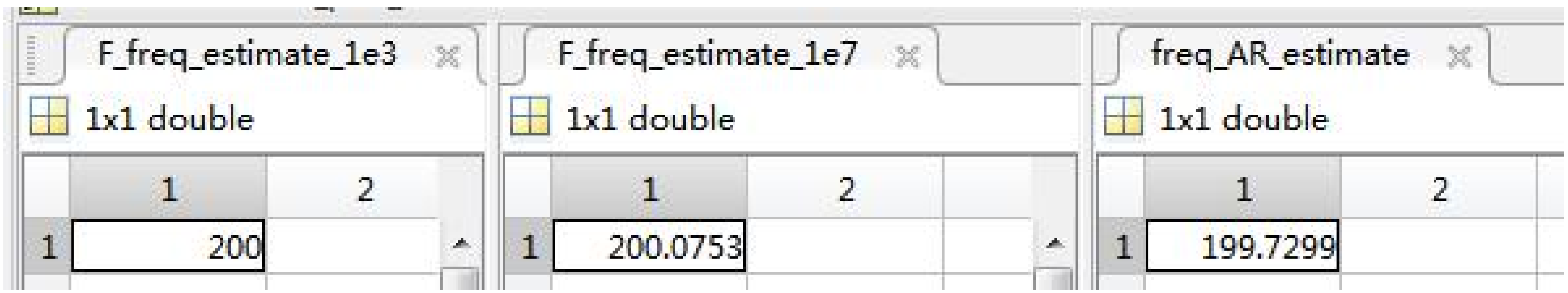
code3

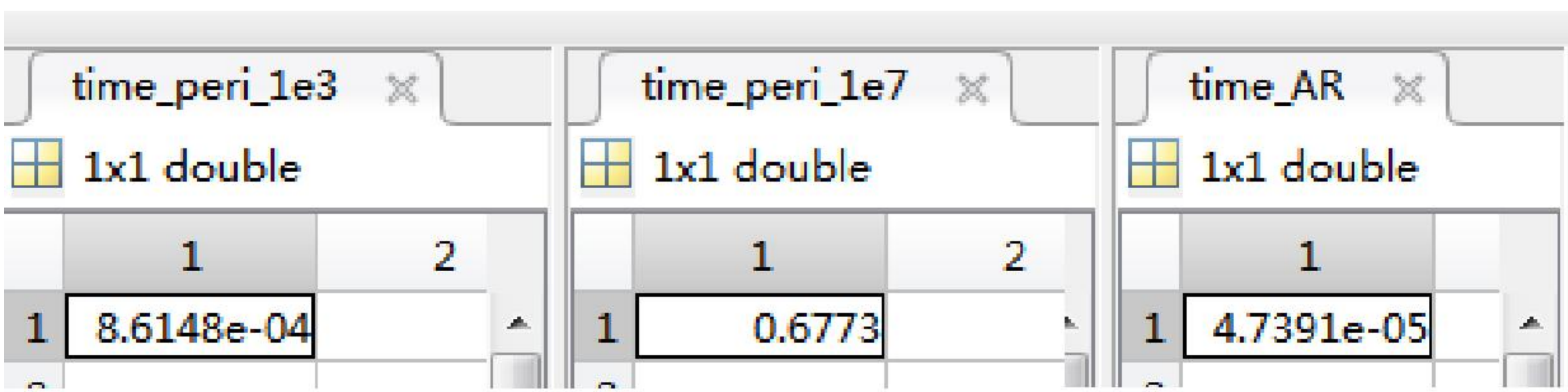
```
Theta = rand(1)*2*pi;
X_signal = Amplitude_A * cos(omega*n_array+Theta);
Noise_sigma2 = mean(X_signal.^2) / [10^(SNR/10)];
Noise = sqrt(Noise_sigma2) * randn(N, 1);
Y_receive_real = X_signal + Noise_sigma2;
%%use Hilbert transform to get the iscrete-time analytic signal
Y_receive = hilbert(Y_receive_real);
window = boxcar(N);        %矩形窗
%% the Auto Regressive (AR) method
Y_r1 = Y_receive(2:end);
Y_r2 = Y_receive(1:end-1);
```

Frequency Estimation Problem

$$y(n) = x(n) + q(n)\ ,\ n = 0,1,2, \ldots, N-1$$
$$x(n) = A\cos(\omega n + \theta)$$
$$x'(n) = A\,\{\cos(\omega n + \theta) + j\sin(\omega n + \theta)\} = Ae^{j(\omega n+\theta)}$$

Frequency 200.07Hz, SNR = 10dB





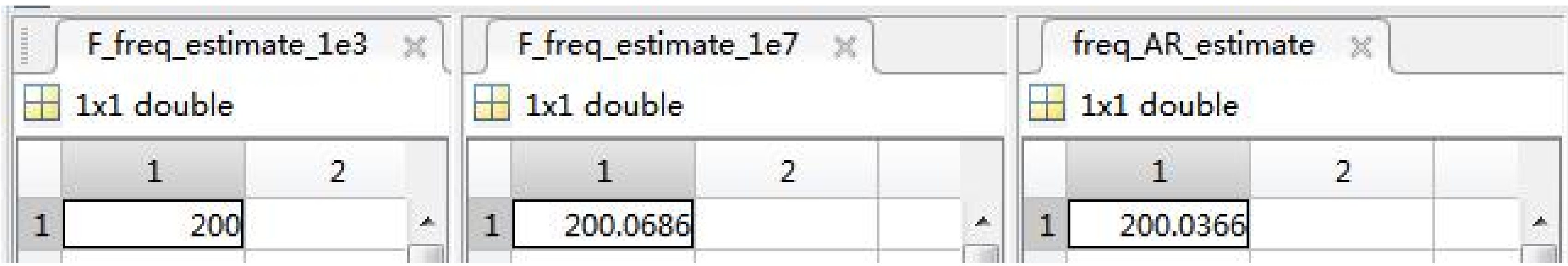
Frequency Estimation Problem

$$y(n) = x(n) + q(n)\ ,\ n = 0,1,2,\ldots,N-1$$

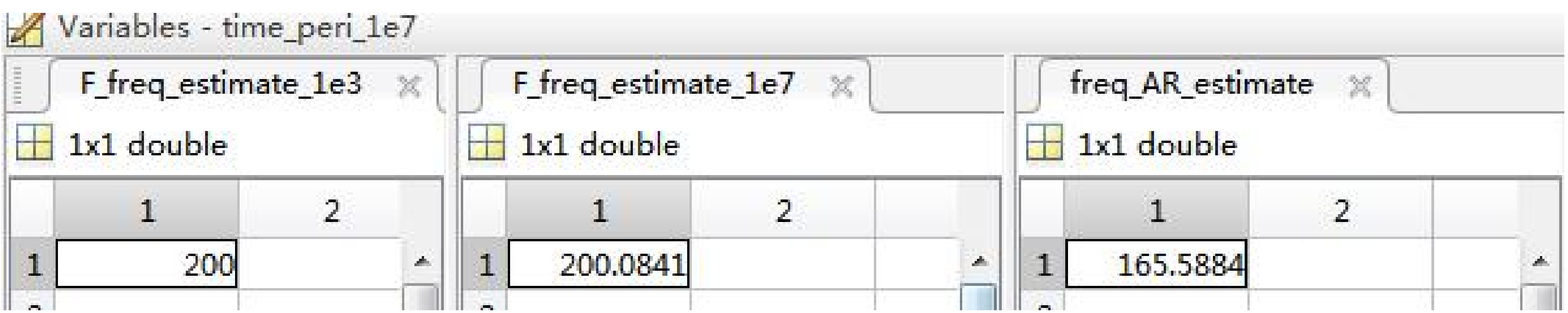
$$x(n) = A\cos(\omega n + \theta)$$

$$x'(n) = A\,\{\cos(\omega n + \theta) + j\sin(\omega n + \theta)\} = Ae^{j(\omega n + \theta)}$$

Frequency 200.07Hz, SNR = 40dB



SNR = 0dB



Frequency Estimation Problem

$$y(n) = x(n) + q(n)\ ,\ n = 0,1,2,\dots,N-1$$

real: $x(n) = Acos(\omega n + \theta)$

complex: $x(n) = Ae^{j(\omega n + \theta)}$

- Application
    - Microphone array
    - Radar target (DOA/point cloud) estimation

Frequency Estimation Problem

$$y(n) = x(n) + q(n)\ ,\ n = 0,1,2, \ldots, N-1$$

real: $\qquad x(n) = Acos(\omega n + \theta)$

complex: $\qquad x(n) = Ae^{j(\omega n + \theta)}$

- In array model, in each $x(n)$, there exist much more than 1 sample. Then we have:

$$\boldsymbol{Y}^k = \begin{bmatrix} y^k(0) \\ y^k(1) \\ \vdots \\ y^k(N-1) \end{bmatrix} \text{for } k = 1,2, \ldots, K$$

How to make the estimation more accurate?

Frequency Estimation Problem

$$y(n) = x(n) + q(n)\ , n = 0,1,2, \ldots, N-1$$

real: $$x(n) = Acos(\omega n + \theta)$$

complex: $$x(n) = Ae^{j(\omega n + \theta)}$$

- Can we detect whether there is a signal? Or says, seperate

$$y(n) = x(n) + q(n)$$
$$y(n) = q(n)$$

The array

$$y(n,t) = x(n,t) + q(n,t)$$
$$n = 0,1,2,\dots,N-1$$
$$t = 1,2,\dots,T$$

Where N is the number of sensors / microphones / antennas. T is the number of samples / snapshots.

real: $x(n,t) = A(t)cos(\omega n + \theta(t))$

complex: $x(n,t) = A(t)e^{j(\omega n+\theta(t))}$

- An array of size N receive signal for T times.

The array – detection problem

$$y(n,t) = x(n,t) + q(n,t)$$

complex: $x(n,t) = A(t)e^{j\theta(t)}e^{j\omega n}$

- For each time t, the amplitude $A(t)$ and the phase $\theta(t)$ of the signal is different, but the frequency ω does not change.
- The target: detect whether there is a signal?

$$y(n) = x(n) + q(n)$$
$$y(n) = q(n)$$

The array – detection method

$$y(n,t) = x(n,t) + q(n,t)$$

complex: $x(n,t) = A(t)e^{j\theta(t)}e^{j\omega n} = \alpha(t)\, e^{j\omega n}$

- In matrix form: $\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{Q}$

$$\boldsymbol{X} = \begin{bmatrix} x(0,0) & \cdots & x(0,T-1) \\ \vdots & \ddots & \vdots \\ x(N-1,0) & \cdots & x(N-1,T-1) \end{bmatrix}$$

- Note that

$$\boldsymbol{X} = \boldsymbol{u} * \boldsymbol{v}^T$$

$$\boldsymbol{u} = [e^{j\omega 0} \quad e^{j\omega 1} \quad \ldots \quad e^{j\omega(N-1)}]^T$$

$$\boldsymbol{v} = [\alpha(0) \quad \alpha(1) \quad \ldots \quad \alpha(T-1)]^T$$

The array – detection method

$$y(n,t) = x(n,t) + q(n,t)$$

complex: $x(n,t) = A(t)e^{j\theta(t)}e^{j\omega n} = \alpha(t)\, e^{j\omega n}$

- In matrix form: $\boldsymbol{Y} = \boldsymbol{u}\boldsymbol{v}^T + \boldsymbol{Q}$
- Then we can get the covariance matrix of $\boldsymbol{Y}$:

$$\boldsymbol{C} = \boldsymbol{Y}\boldsymbol{Y}^H = \boldsymbol{X}\boldsymbol{X}^H + \boldsymbol{Q}\boldsymbol{X}^H + \boldsymbol{X}\boldsymbol{Q}^H + \boldsymbol{Q}\boldsymbol{Q}^H$$

- $\boldsymbol{X}$ $(x(n,t))$ and $\boldsymbol{Q}$ $(q(n,t))$ independent:

$$E(\boldsymbol{Q}\boldsymbol{X}^H) = E(\boldsymbol{X}\boldsymbol{Q}^H) = \mathbf{0} \text{ (when } T \to \infty)$$

- $\boldsymbol{Q}^H\boldsymbol{Q}$ is full rank noise, and

$$\boldsymbol{X}\boldsymbol{X}^H = \boldsymbol{u}\boldsymbol{v}^T\boldsymbol{v}^*\boldsymbol{u}^H$$

- $\boldsymbol{v}^*$ is the conjugate, $\boldsymbol{v}^T$ is transpose, $\boldsymbol{u}^H$ is conjugate transpose

The array – detection method

$$y(n,t) = x(n,t) + q(n,t)$$

complex: $x(n,t) = A(t)e^{j\theta(t)}e^{j\omega n} = \alpha(t)\, e^{j\omega n}$

- ovariance matrix of $\boldsymbol{Y}$:

$$\boldsymbol{C} = \boldsymbol{Y}\boldsymbol{Y}^H = \boldsymbol{u}\boldsymbol{v}^T\boldsymbol{v}^*\boldsymbol{u}^H + \boldsymbol{Q}\boldsymbol{Q}^H$$

$$\boldsymbol{u} = [e^{j\omega 0} \quad e^{j\omega 1} \quad \dots \quad e^{j\omega(N-1)}]^T$$

$$\boldsymbol{v} = [\alpha(0) \quad \alpha(1) \quad \dots \quad \alpha(T-1)]^T$$

$\boldsymbol{v}^T\boldsymbol{v}^* = a$ is a number, refers the signal ‘strength’

$$\boldsymbol{C} = \boldsymbol{Y}\boldsymbol{Y}^H = a\,\boldsymbol{u}\boldsymbol{u}^H + \boldsymbol{Q}\boldsymbol{Q}^H$$

The array – detection method

$$y(n,t) = x(n,t) + q(n,t)$$

complex: $x(n,t) = A(t)e^{j\theta(t)}e^{j\omega n} = \alpha(t)\, e^{j\omega n}$

$$\boldsymbol{C} = \boldsymbol{Y}\boldsymbol{Y}^H = a\, \boldsymbol{u}\boldsymbol{u}^H + \boldsymbol{Q}\boldsymbol{Q}^H$$

- Now we do the eigenvalue decomposition of $\boldsymbol{C}$:

$$\boldsymbol{C} = \boldsymbol{U\Sigma U^H}$$

Then

$$\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_N]$$

$$\boldsymbol{u}_1 = \boldsymbol{u} + noise\ ;\ \boldsymbol{u}_n = noise \text{ for } n \neq 1$$

$$\boldsymbol{\Sigma} = \mathbf{diag}([\alpha_1, \alpha_2, \dots, \alpha_N])$$

$\alpha_1 = a + noise$ is **large**

$\alpha_n = noise$ is **small** for $n \neq 1$

The array – detection method

- Noise only:

$$y(n,t) = q(n,t)$$
$$\boldsymbol{C} = \boldsymbol{Y}\boldsymbol{Y}^H = \boldsymbol{Q}\boldsymbol{Q}^H$$

- Now we do the eigenvalue decomposition of $\boldsymbol{C}$:

$$\boldsymbol{C} = \boldsymbol{U\Sigma U^H}$$

Then

$$\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N]$$

$$\boldsymbol{u}_n = noise \text{ for all } n$$
$$\boldsymbol{\Sigma} = \mathbf{diag}([\alpha_1, \alpha_2, \ldots, \alpha_N])$$

$$\alpha_n = noise \text{ is } \textbf{small} \text{ for all } n$$

The array – detection method

code4.m*

```matlab
clear
clc

N = 8;
T_snapshot = 100;
F_sam_freq=1000;     %采样频率
F_freq = 200.07;
omega = F_freq/F_sam_freq*2*pi
SNR = 10;
Amplitude_A_mean = 0;
Amplitude_A_std = 1;
n_array = [0:N-1]';
t_array = [0:T_snapshot-1];
```

The array – detection method

```
Theta = rand(1)*2*pi;
Amplitude_At = Amplitude_A_std*randn(1,T_snapshot)+Amplitude_A_mean;
Theta_t = rand(1,T_snapshot)*2*pi;
Frequnecy_t = exp(1j*omega*n_array);
X_signal = Frequnecy_t * (Amplitude_At.*exp(1j*Theta_t));
X_reshape = reshape(X_signal, N*T_snapshot, 1);
X_signal_power = X_reshape'*X_reshape/(N*T_snapshot);
Noise_sigma2 = X_signal_power / [10^(SNR/10)];
Noise = sqrt(Noise_sigma2) * randn(N, T_snapshot) .* exp(1j*rand(N, T_snapshot)*2*pi);
Y_receive = X_signal + Noise;

%% detection method
Y_cov_matrix = Y_receive*Y_receive';
[SigNoise_Vectors,SigNoise_Values] = eig(Y_cov_matrix);

Noise_cov_matrix = Noise*Noise';
[Noise_Vectors,Noise_Values] = eig(Noise_cov_matrix);
```

The array – detection method

SigNoise_Values

8x8 double

	1	2	3	4	5	6	7	8
1	5.9121	0	0	0	0	0	0	0
2	0	7.6077	0	0	0	0	0	0
3	0	0	8.1600	0	0	0	0	0
4	0	0	0	9.9052	0	0	0	0
5	0	0	0	0	10.2992	0	0	0
6	0	0	0	0	0	11.9626	0	0
7	0	0	0	0	0	0	13.6964	0
8	0	0	0	0	0	0	0	785.5127

Noise_Values

8x8 double

	1	2	3	4	5	6	7	8
1	5.8884	0	0	0	0	0	0	0
2	0	6.5446	0	0	0	0	0	0
3	0	0	7.9414	0	0	0	0	0
4	0	0	0	9.4648	0	0	0	0
5	0	0	0	0	10.0606	0	0	0
6	0	0	0	0	0	11.2955	0	0
7	0	0	0	0	0	0	12.6686	0
8	0	0	0	0	0	0	0	14.4646

注意，Matlab的eig函数的奇异值是从小到大排列的，和数学公式的写法有些许不一样

The array – detection method

$$y(n,t) = x(n,t) + q(n,t)$$

complex: $x(n,t) = A(t)e^{j\theta(t)}e^{j\omega n} = \alpha(t)\, e^{j\omega n}$

$$\boldsymbol{C} = \boldsymbol{Y}\boldsymbol{Y}^H = a\,\boldsymbol{u}\boldsymbol{u}^H + \boldsymbol{Q}\boldsymbol{Q}^H$$

- Now we do the eigenvalue decomposition of $\boldsymbol{C}$:

$$\boldsymbol{C} = \boldsymbol{U\Sigma U^H}$$

$\alpha_1 = a + noise$ is **large**

$\alpha_n = noise$ is **small** for $n \neq 1$

- If α_1/α_2 or α_1/α_N is large, for example, > 10, then we can judge that there is signal; otherwise, there exists noise only

More

- How about?:

$$y(n,t) = x_1(n,t) + x_2(n,t) + q(n,t)$$

That is, there are two signals?

Or

$$y(n,t) = \sum_{i=1}^{K} x_i(n,t) + q(n,t)$$

That is, there are K signals?

K known?

K unknown?

Try yourself