EECM30136:

高等通訊系統模擬與實驗

Advanced Communication System Simulations and Experiments

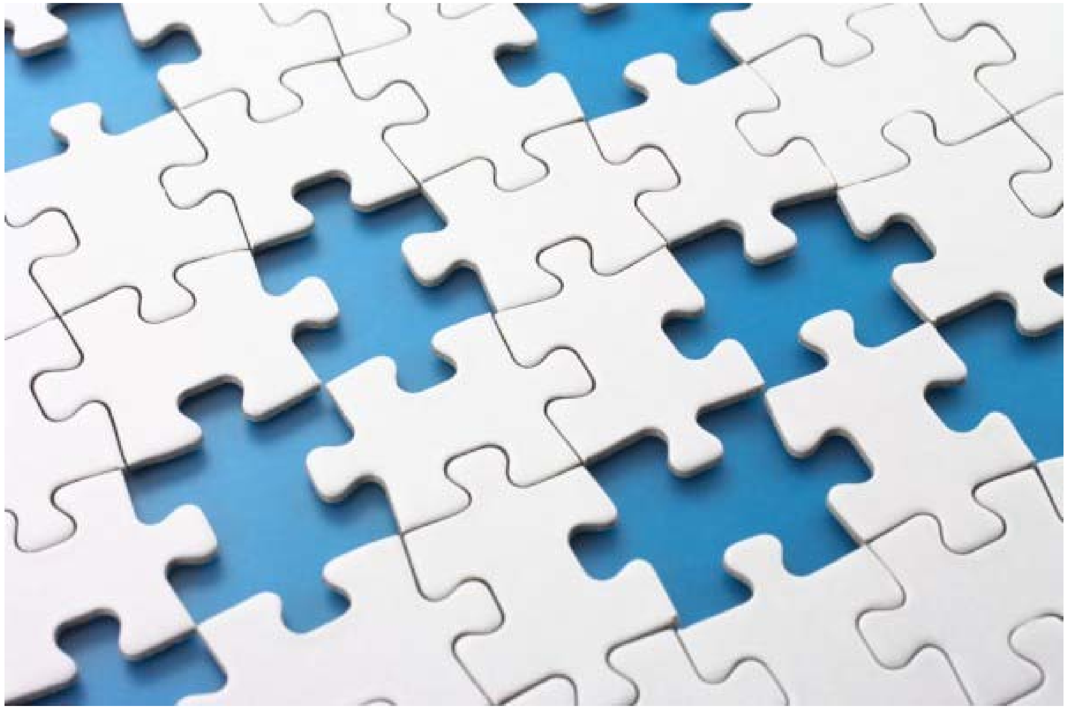
This course provides a series of computer experiments that let you learn how digital data are ***actually*** *transmitted and received in wireless systems*.

- How can best describe your knowledge about communications?

<一知半解>
<支離破碎>

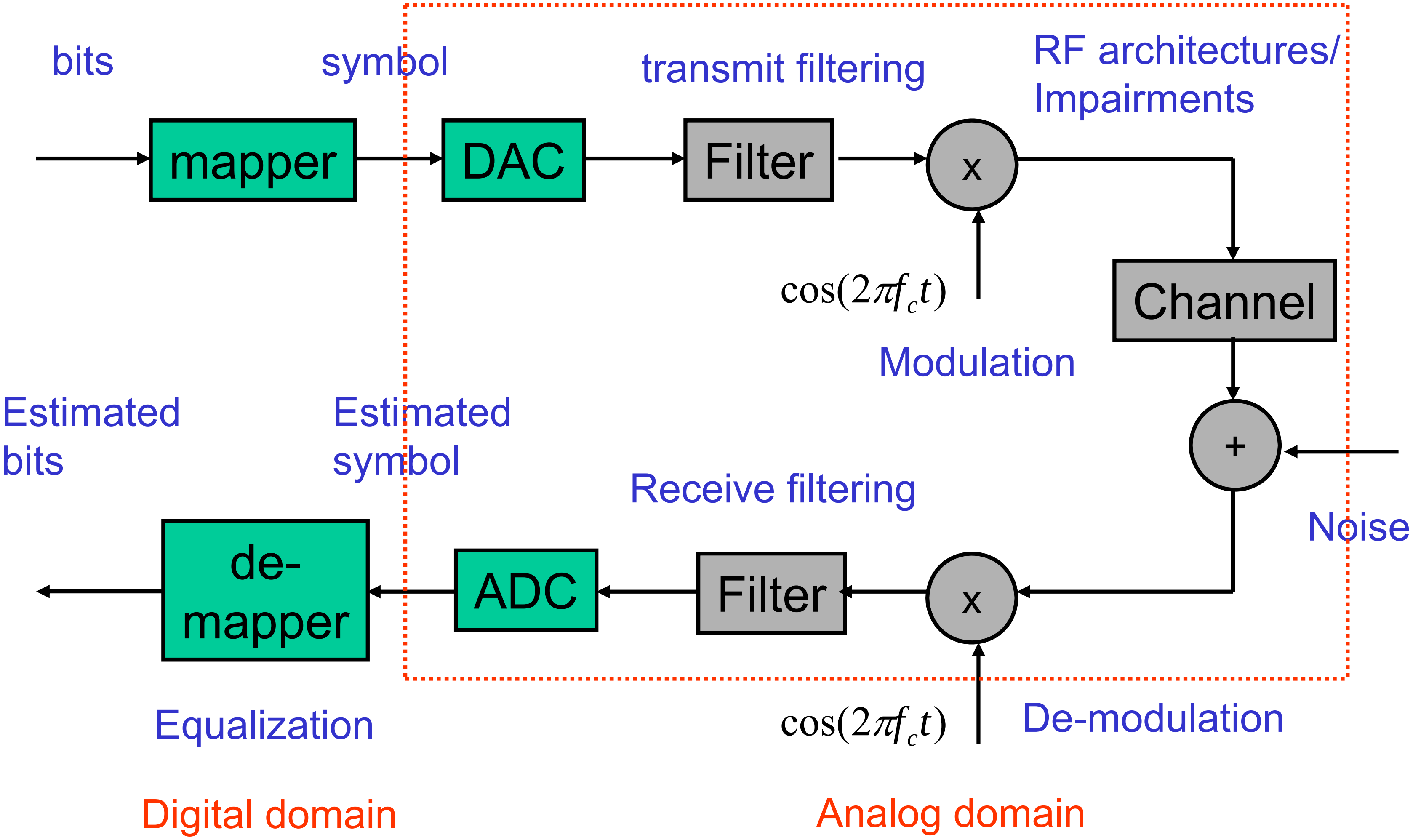
- Do you know?
  - What is “matched filter”?
  - Why signal/channel is “complex”?
  - What is the difference between a “digital” and “analog” filter?
  - Why do we need “modulation”?
  - What transceiver “structures” we have?
  - …
  - How a “bit” is actually transmitted and received?

- Missing puzzles:



- The purpose of this course:
  - Let you fully understand how communication works
  - Provide you simulation skills

- Digital communication system:



- Three domains: digital, analog, and RF

- In the course, you are supposed to implement/design some typical communication systems with computer programs.
- Programming language – Matlab
- Why simulations?
  - The first step in communication system design/realization
- Type of simulations
  - Floating point simulations: to evaluate the performance of a behavior model
  - Fixed point simulations: to evaluate the performance of a structure model
- In this course, we mainly focus on the behavior model of communication system.

- Grading policy:
  - Quizs: 10%
  - In-class practice: 20%
  - Homework (report): 20%
  - Midterm/final: 50% (online test)
- **Instructor**: 吳文榕， e-mail: wrwu@nycu.edu.tw
- **TA**:羅威淇 : wickylo.ee11@nycu.edu.tw，蘇玠文 : jwsu.ee12@nycu.edu.tw
- Text book: None
- Lecture notes can be downloaded from e3.
- References
  - User manual of Matlab
  - Text books for *Signals and Systems* and *Principles of Communication Systems*

- **Course outline:**
  - Lab 1 MATLAB introduction
  - Lab 2 Signals and the Fourier transform (*)
  - Lab 3 LTI systems and their responses (*)
  - Lab 4 Analog modulation (*)
  - Lab 5 Digital modulation (*)
  - Lab 6 Sampling and rate conversion (*)
  - Lab 7 Transmit filtering/up conversion I (*)
  - Lab 8 Transmit filtering/up conversion II
  - Lab 9 Receive filtering/down conversion
  - Lab 10 RF impairments
  - Lab 11 Equalization (*)
  - Lab 12 Constant envelop modulation (*)
  - Lab 13 MIMO transmission
  - Lab 14 Fixed-point implementation
  - Lab 15 Testing

(*): in-class quiz

Format of Report

- Problem
  - Problem
  - Background, theories, or ideas
- Simulation results and discussions
  - Simulations results to explain the behaviors of the algorithm or method investigated (table or figures)
  - Discussions
- References
- Appendix (codes)

Grading Policy

Correctness 50%

correctness, reasonableness

Completeness 20%

Simulation procedure, parameter
finished items

Discussion 15%

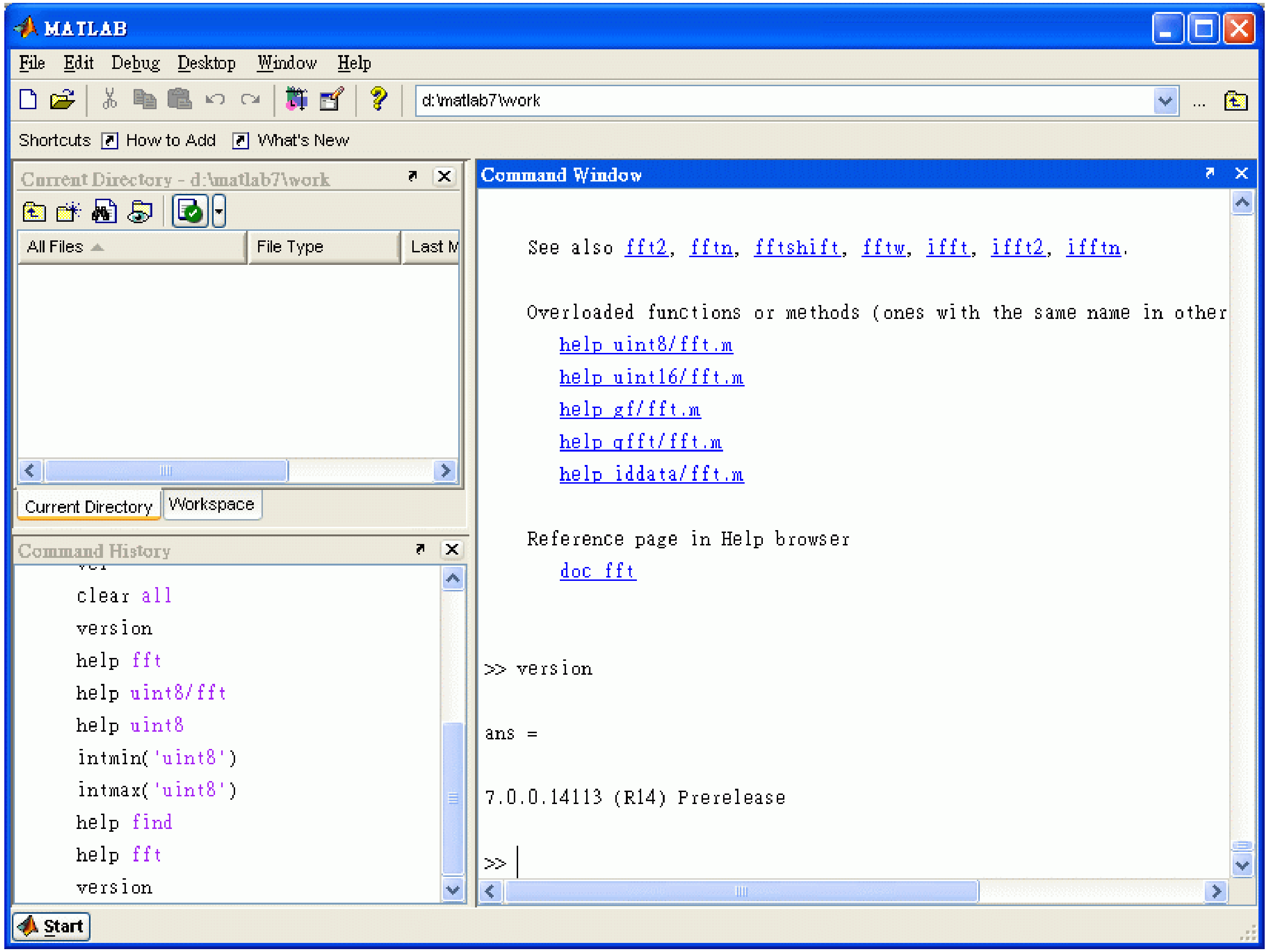
Result discussions, property interpretation

Readability 15%

Appearance, organization, data/figure

Lab. 1 MATLAB Introduction

- Initiation:



```
ver
clear all
version
help fft
help uint8/fft
help uint8
intmin('uint8')
intmax('uint8')
help find
help fft
version
```

```
    See also fft2, fftn, fftshift, fftw, ifft, ifft2, ifftn.

    Overloaded functions or methods (ones with the same name in other
       help uint8/fft.m
       help uint16/fft.m
       help gf/fft.m
       help qfft/fft.m
       help iddata/fft.m

    Reference page in Help browser
       doc fft


>> version

ans =

7.0.0.14113 (R14) Prerelease

>>
```

- MATLAB is an interpreter not a compiler language
  - It interprets a command whenever you want to execute it.
- General mathematical expressions:

```
>> 2*10^2+2
   ans=
         202
```

- If you do not want to show the result on screen, just put “;” after your command.
- You can use “%” to add comments.

```
>> 2*10^2+2; % This is a test
```

- Vector/matrix definition:

```
>> x=[1 2 3 4]; y=[1;1;1;1];

>> x=[1 2;3 4]; y=[3 2;1 1]
```

- Matrix:
  - All the matrices are stored in a column-by-column fashion
  - a(i,j) accesses the ijth component of matrix a
  - We can use an index range to access submatrix of a matrix. e.g., b=a(2:3, 5:6);
  - For a whole column of row, we use a(1,:) and a(:,3) or a(:,end)
  - b=diag(a)
  - b=reshape(a, 2,4) % a must have 8 elements.
- Useful matrices:

```
zeros(n,m)
ones(n,m)
eye(n,m)
rand(n,m)
randn(n,m)
```

- It is better to define the size of a vector/matrix before its use.

```
>> x=zeros(1,100);
>> y=zeros(1000,1000);
```

- Mathematic functions:

```
>> x=abs(y)
>> x=log(y); x=log10(y)
>> x=sin(2*pi*0.1*[1:0.1:30])
>> x=exp(y)
```

```
>> y=max(x); y=max([3 1 2 5]);
>> y=min(x)
>> y=sort(x)
>> y=x.^2
```

```
>> y=inv(x);
>> y=eig(x)
>> y=(x)
>>[q,r]=qr(A)
```

* Functions (toolbox): There are many functions that you can use.

- Signal processing tool box
- Communication tool box

- Complex numbers: 1+i or 1+j

```
>> x=[1+j; 1-j]; y=[-2+j; 2-j];
>> xr=real(x)
>> xi=imag(y)
```

- Mathematic operations:

1. (.’), (.^), (’), (^)
2 (+), (-)
3. (.*), (./), (.\), (*), (/), (\)
4. (+), (-)
5. (:)

```
>> x=[1 2;3 4]; y=[3 2;1 1]
>> x*y
>> z=[x(2,1:2); y(1,:)]
>> x.*y
>> x^2
>> x.^2
```

- Scalar expansion:

```
>> a = [1 2; 3 4];
>> b = a+1;
>> c = 1./a
```

- If you want to find the size of a matrix use size(x).

- Loops:

```
>> for i=1:10
     x=i^2;
   end
```

```
>> x=0;
   while x<10
     x=x+1;
   end
```

- Conditional statements:

```
>> x=-1;
   If x<0
     y=1;
   elseif x==0
     y=0;
   else
     y=-1;
   end
```

- Relational operations:

==, ~=, <, <= >, >=

- Logical operators: &, |, ~

If x==1 & y==0

- If you have a large amount of commands, you can put them in a file (xxx.m file). Just type the file name in the command window, Matlab will execute the commands in the file.

- Practice 1:
  - Let an input sequence have positive and negative elements.
  - Using the loop command to find the summations of its positive and negative elements.

- Some useful commands:

```
>> pwd
>> cd
>> who
```

- Command “clear” will clear all the variables in the memory.
- You can save the variables in the memory to the disk.
  - save filename
  - save filename x,y,z
  - load filename
- You can use “help” to find out the usage of a command, or “lookfor” for a specific word in the command.

- 2D plot :

```
>> x=0: .1 : 4*pi
>> y=cos(x);
>> plot(y)
```

- Multiple curves in a figure:

```
>> x=linspace(0, 2*pi);
>> plot(x, sin(x), x, cos(x), x, sin(x)+cos(x))
```

- You can also use different markers for different curves.

```
>> x=linspace(0, 2*pi);
>> plot(x, sin(x), ‘o’, x, cos(x), ‘x’, x, sin(x)+cos(x), ‘*’)
```

- For a matrix, “plot” plots the columns of the matrix.

```
>> x=peaks;   % 49x49 matrix
>> plot(x)
```

- Thus, you can have

```
>> x=linspace(0, 2*pi);
>> plot([sin(x’), cos(x’), sin(x’)+cos(x’)])
```

- For a complex vector, plot(z) is equivalent plot(real(z), imag(z)).
- You can define the maximum and minimum values of the x- and y-axis.

```
axis([xmin, xmax, ymin, ymax])
```

- You can also plot two or four subplots in a plot.

```
x = 0:0.1:4*pi;
subplot(2, 2, 1); plot(x, sin(x));
subplot(2, 2, 2); plot(x, cos(x));
subplot(2, 2, 3); plot(x, sin(x).*exp(-x/5));
subplot(2, 2, 4); plot(x, x.^2);
```

* Grid and box:

```
>> grid on
>> grid off
>> box on
>> box off
```

- Input from keyboard:

```
x=input('Please input the parameter?')
```

- Output to screen:

```
disp('There is an error!')
disp(['The result is ' num2str(x)])
```

- 3-D plot: “mesh” and “surf”
  - Plot a matrix (as the high of the z-axis).

```
x = linspace(-2, 2, 25);
y = linspace(-2, 2, 25);
[xx, yy] = meshgrid(x, y);
zz = xx.*exp(-xx.^2-yy.^2);
mesh(xx, yy, zz);
```

- Function:
  - Save as an individual file.
  - Call as needed.
  - It is recommended that the file name is the same as that of the function name.

```
function [s,p] = spf(x)
s = sum(x);
p= prod(x);
```

- Debug:
  - Use “keyboard” command in the program. When the program is executed to the position, it will stop (K>).
  - You can check the values of variables.
  - Use “return” to resume the execution.
- You can use the debugger provided in the Matlab editor.
  - Set/clear break points
  - Step execution

- Practice 2:
  - Write a program to calculate the mean and the variance of a set of scores (cannot use statistic functions).
  - The number of the scores is inputted from the keyboard.
  - The scores are also input from the keyboard.
  - The result is output to the screen.
  - Convert this program into a function.

- Homework: Write a program to verify the central limit theorem.
  - Generate a series of random numbers (with uniform distribution) with rand(1,n), and add them to obtain a new random number.
  - Repeat this process for m times.
  - Check the Gaussionality?
  - Use hist(x) to plot the histogram.
  - Kurtosis: $E\{x^4\}/(E\{x^2\})^2-3 \rightarrow 0$

- Reading assignment:
  - Signals
  - Fourier transform

Lab. 2 Signals

- Signal: a function of time
  - Continuous
  - Sampled
  - Discrete-time
- Continuous signal:
  - e.g. sinusoidal $\cos(2\pi ft)$

Lowest frequency : $f = 0$

Highest freqeuncy: $f = \infty$

- Sampled signal:

$$\cos(2\pi fnT)$$

T : samping period

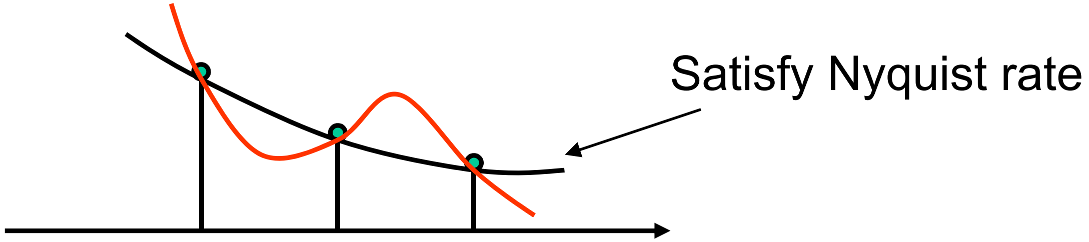
- Discrete-time sequence:

$$\cos(2\pi fn)$$

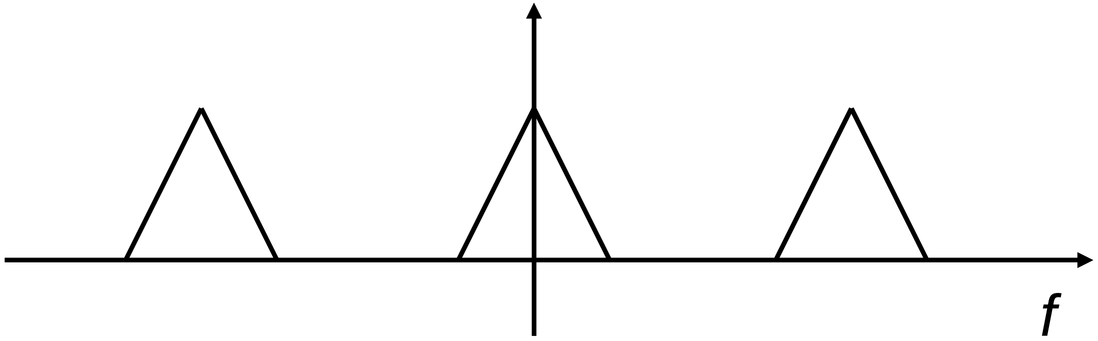
Lowest frequency : $f = 0$

Highest freqeuncy: $f = ?$

- Note that a discrete-time signal cannot define a unique continuous-time signal. This is what the sampling theory tries to tell us.



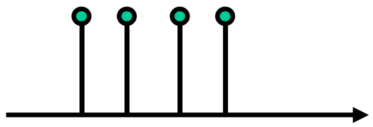
- This is also the reason why the spectrum of a discrete-time signal is periodic.



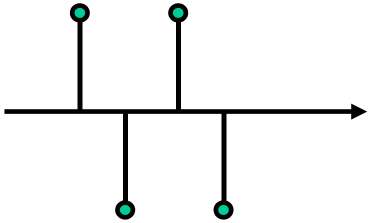
- Continuous and discrete frequency:

$$\cos(2\pi ft) \text{ vs. } \cos(2\pi fn)$$

- Discrete signal:
  - A sequence $\cos(2\pi fn)$



$f = 0$



The maximum frequency:

$f = \frac{1}{N} = \frac{1}{2}$

$* f = \frac{1}{2} \rightarrow \omega = \pi, \quad f = -\frac{1}{2} \rightarrow \omega = -\pi$

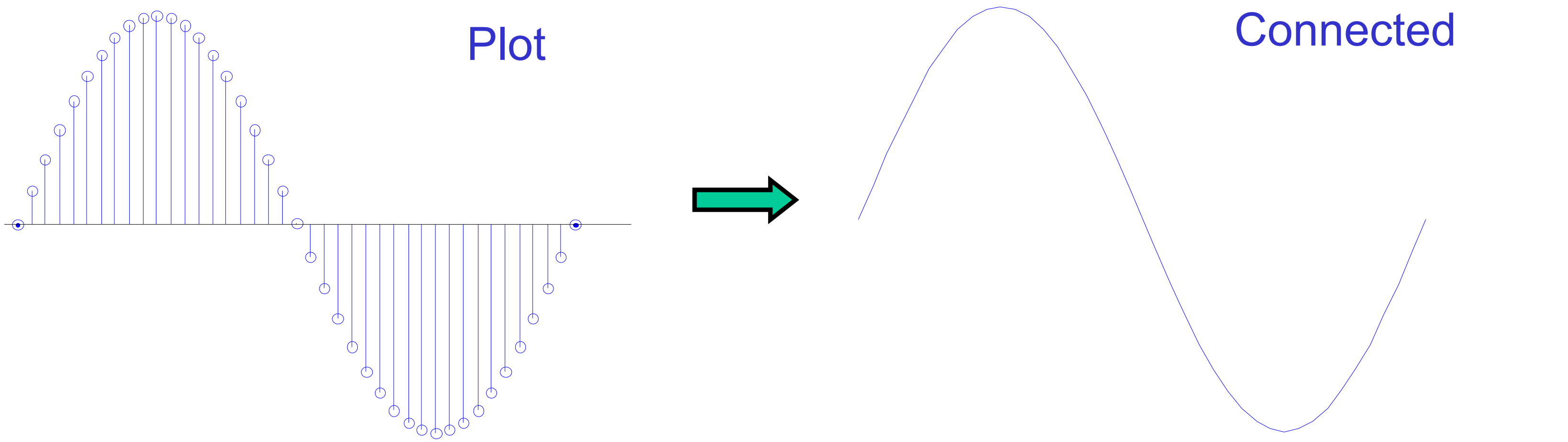
- Note that .

$$\cos(2\pi fnT) = \cos(2\pi \frac{f}{f_s} n) = \cos(2\pi f_d n)$$

* f_d: digital frequency

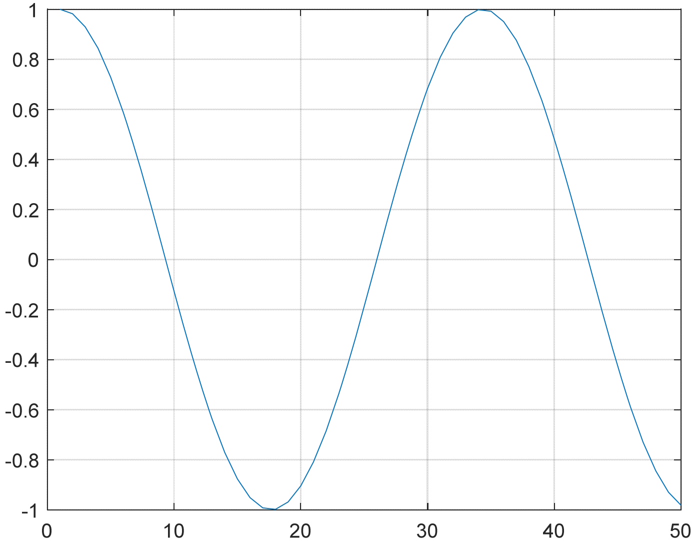
- The frequency of a discrete-time signal can be seen as that of a normalized sampled signal.
- Also, $\cos(2\pi(f \pm N)n) = \cos(2\pi fn \pm 2\pi Nn) = \cos(2\pi fn)$. Thus, the effective frequency range of discrete-time signal is [-1/2,1/2].

- Note that with computers, we can only deal with discrete-time signals.
- That means we can only plot discrete-time signals.

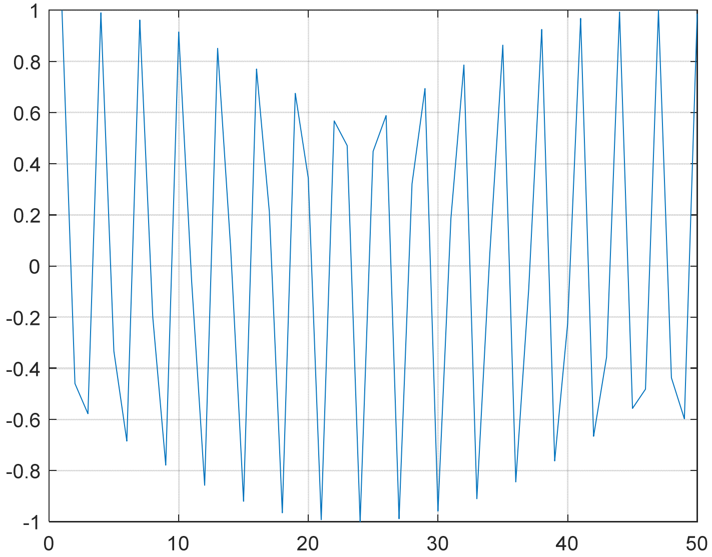


- Practice 1:
  - Generate a sinusoidal signal with length 50 and f=0.03.
  - Gradually increase its frequency to 0.4 and observe the waveforms.
  - Check the equivalence of f, N-f, and N+f.

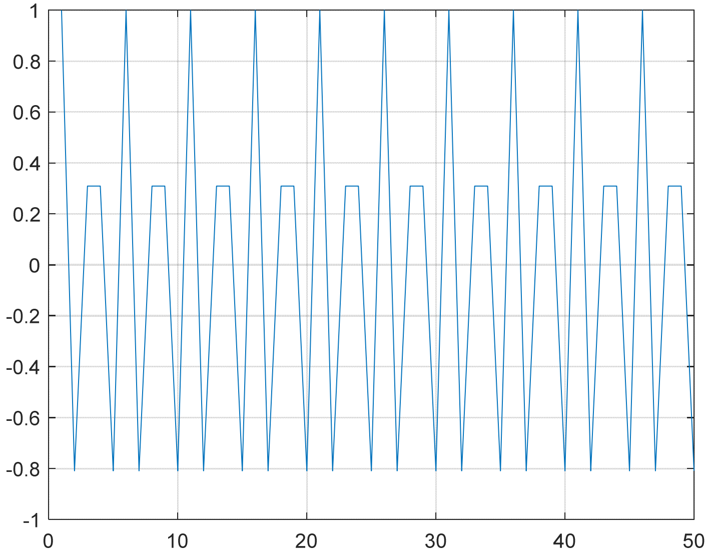
- Results:



f=0.03

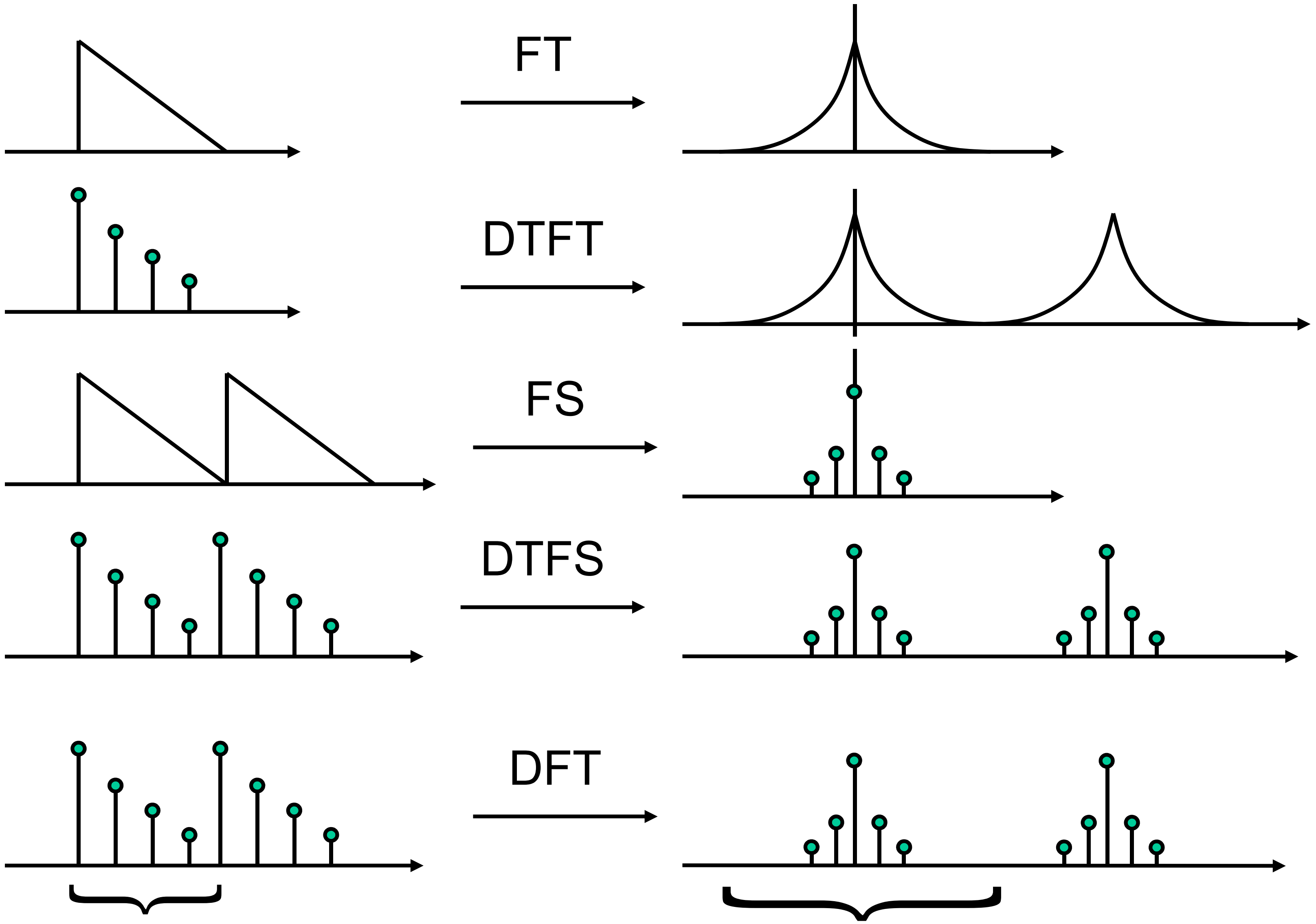


f=0.326



f=0.4

- Frequency domain representation:



FT

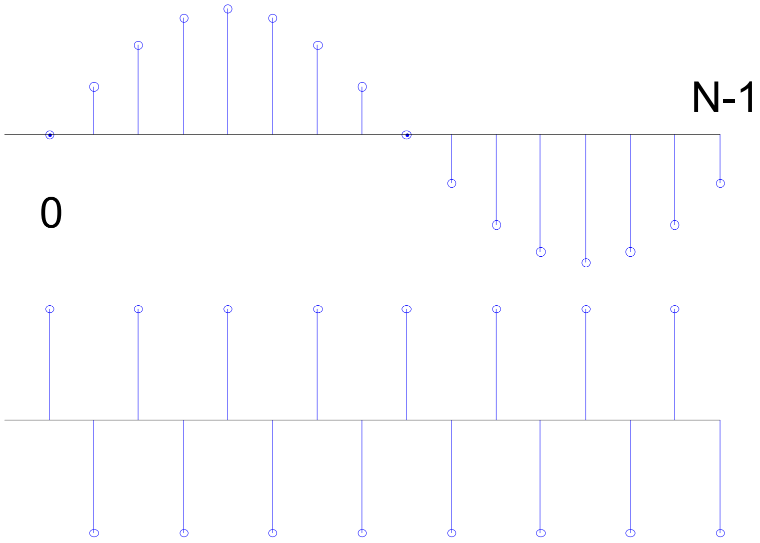
DTFT

FS

DTFS

DFT

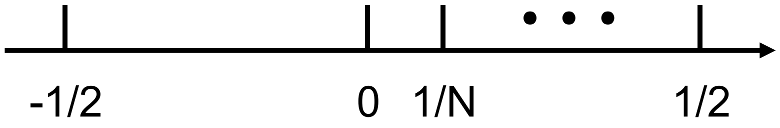
- Discrete-time sinusoidal signal:
  - $f = \frac{i}{N}, \quad i = 1, 2, \cdots, N-1$
  - $f \neq \frac{i}{N}$
- Fundamental frequency for the signal with period N.



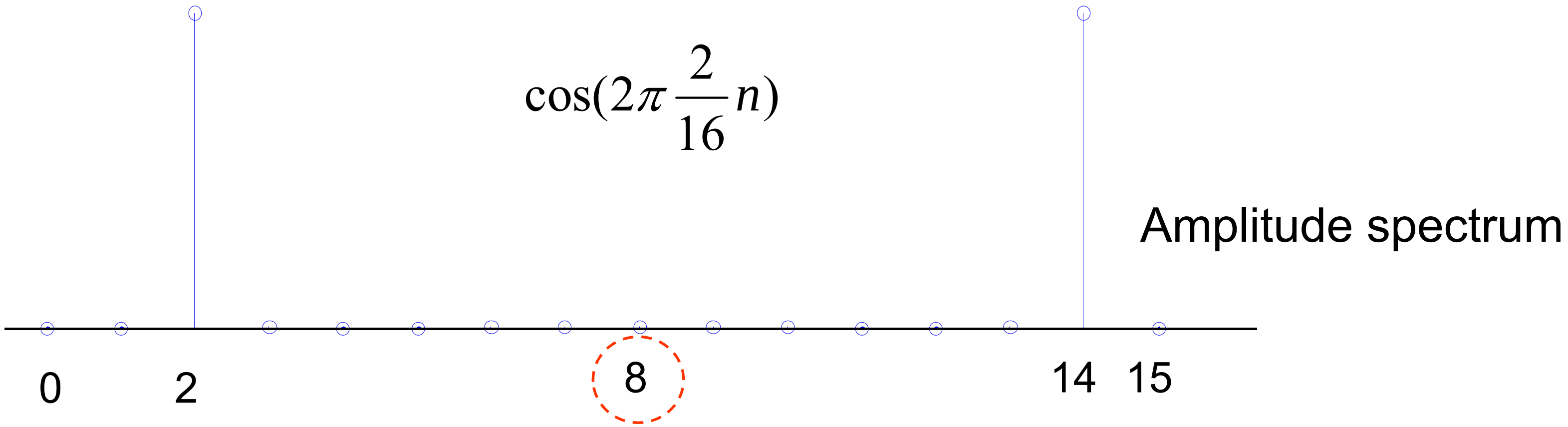
$$f_0 = \frac{1}{N}$$

$$f_0 = \frac{1}{2}$$

- Frequencies:



- DFT: $\cos(2\pi \frac{i}{N} n) \rightarrow \text{DFT}$



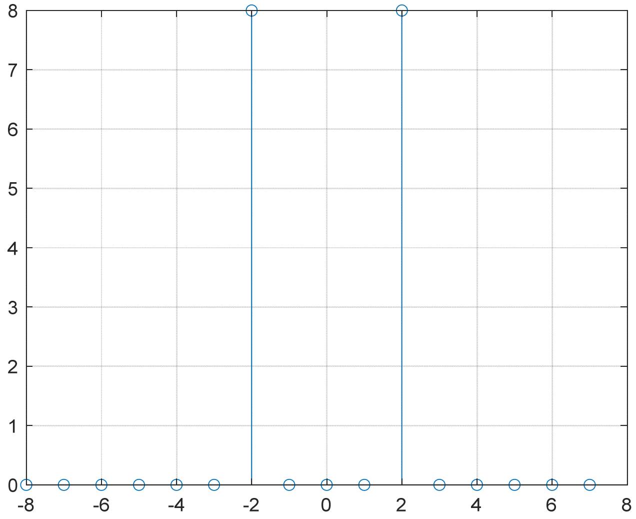
- For real signals, we have two impulses for a sinusoidal since

$$\cos(2\pi fn) = \frac{e^{j2\pi fn} + e^{-j2\pi fn}}{2}$$

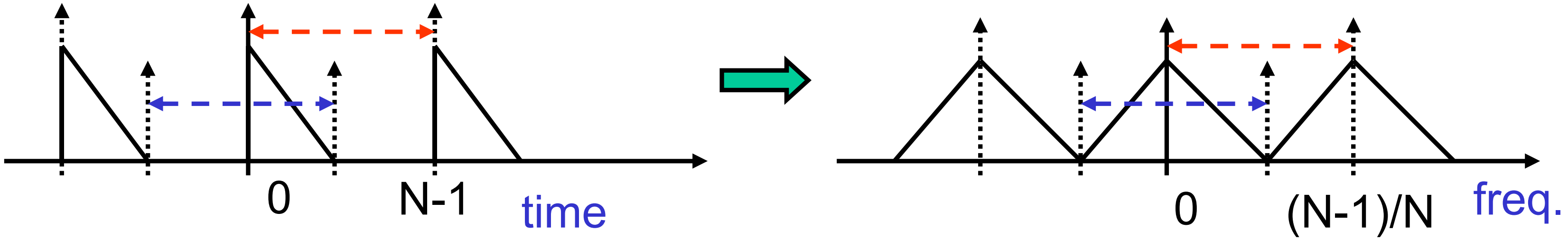
- Practice 2:
  - Generate sinusoidal signals with frequency i/N where N=16 and i=2 (length is also 16).
  - Observe its DFT spectrum, and the relationship to its time domain signal.
  - Also use fftshift(.) to see the spectrum between [-1/2,1/2].

* Fast Fourier transform: fft(x)
* Plot(abs(fft(x)))
* Discrete signal plot: stem(x)

- Result:

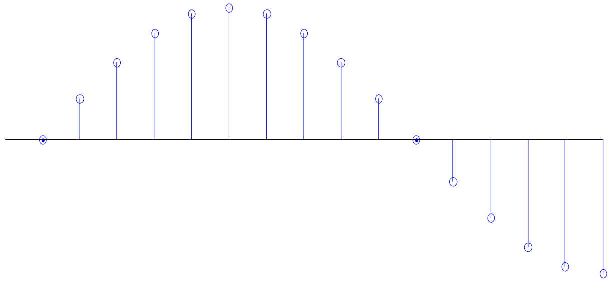


- As mentioned, in Matlab, DFT is conducted with the command “fft”, and IDFT with “ifft”.
- When we use DFT, we assume that the signal is periodic.



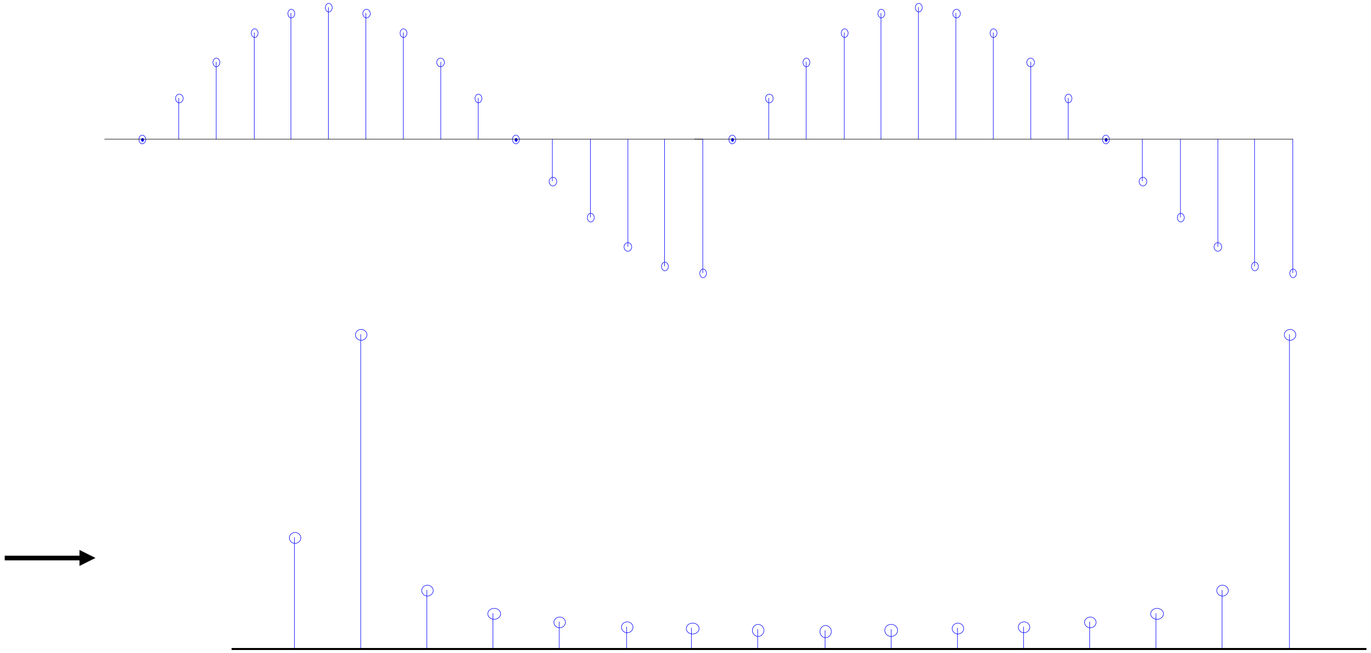
- The command “fft” shows the spectrum period of [0,(N-1)/N] while “ifft” shows the time-domain period of [0,N-1].
- If we want to see the spectrum period of [-1/2, (N-1)/2N] and the time-domain period of [-N/2,N/2-1], we can use the command of “fftshift” to rotate the result of “fft” or “ifft”.

- What about the DFT size is N and $f \neq \frac{i}{N}$?

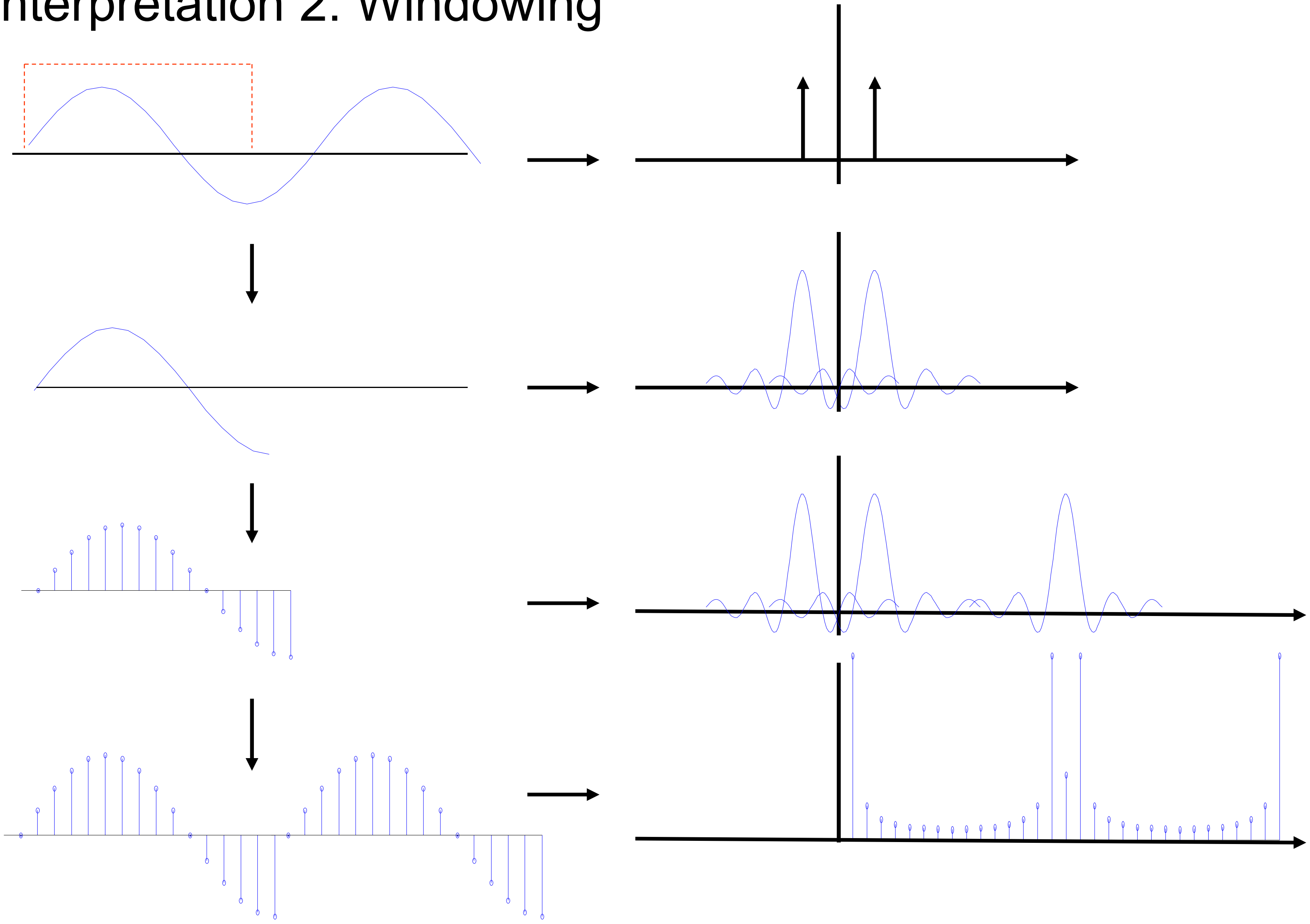


$$f = \frac{0.8}{16}$$

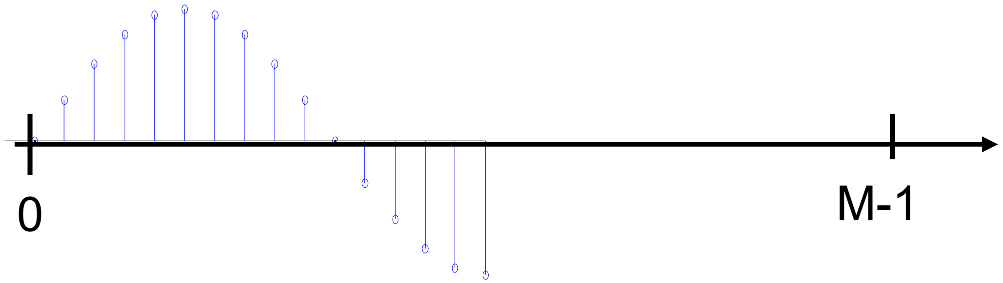
- Interpretation 1: not a sinusoidal signal anymore.



- Interpretation 2: Windowing

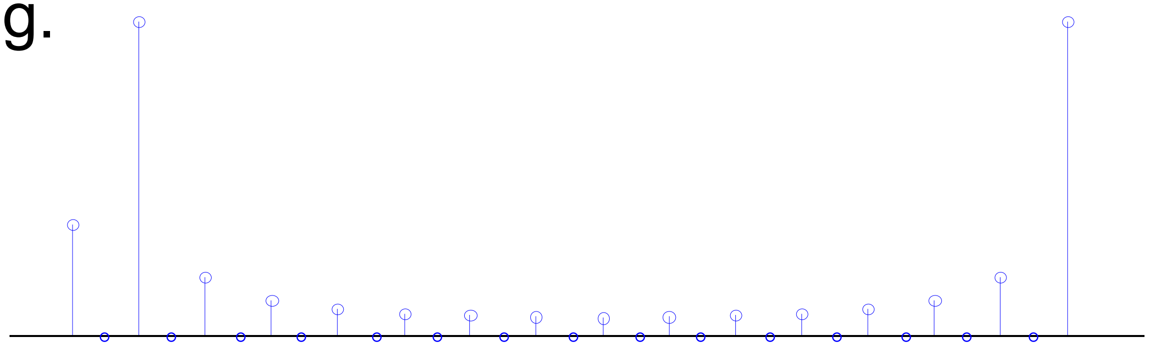


- What about add more zero points in DFT?



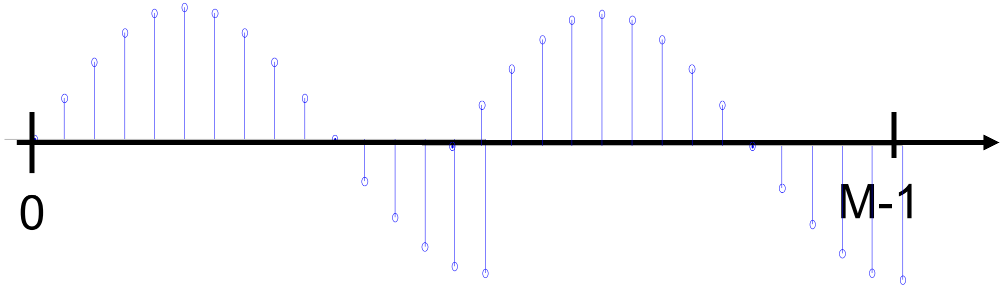
$M > N$

- It is seen that the fundamental frequency becomes 1/M, and M frequencies can be obtained in DFT domain.
- Ex. M=2N. With only original N frequencies (half or them is zero), we can have a time domain signal without zero padding.

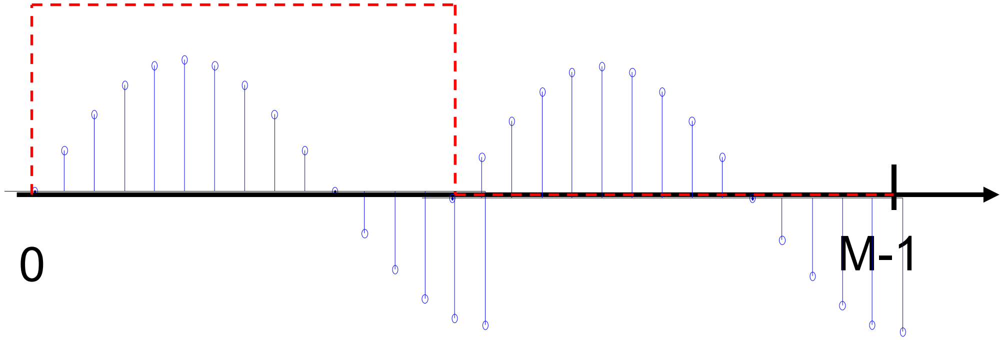


$M = 2 \times 16$

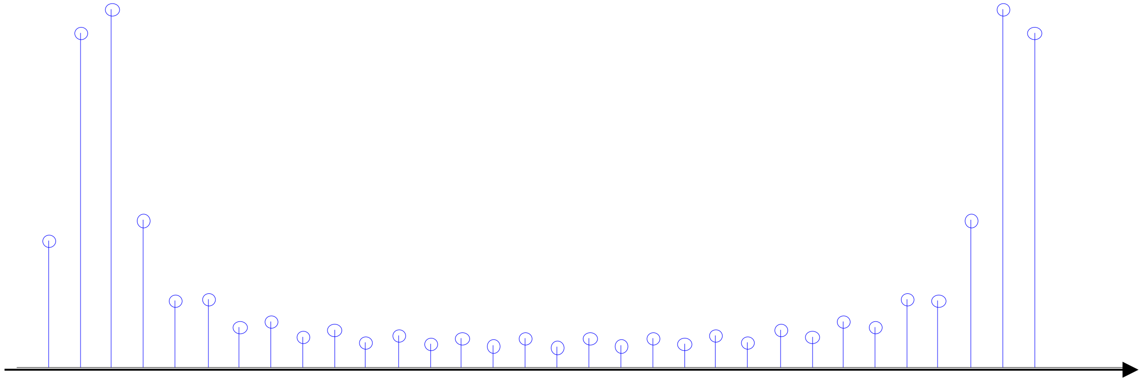
- That is



- To obtain the original zero padding signal, a window has to be used.



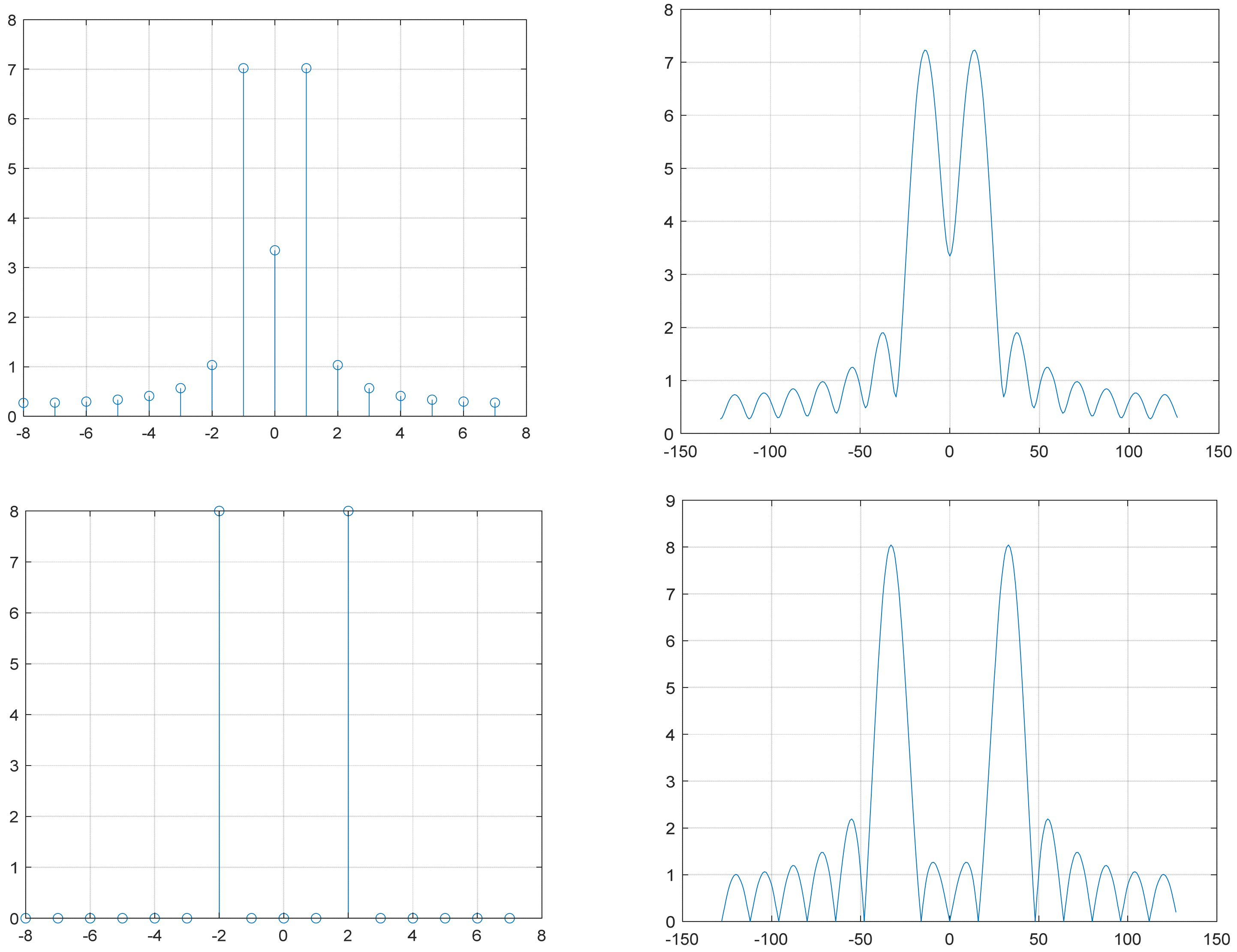
- This results interpolation in frequencies with zero value.



- DFT can also be seen as a sampling of the spectrum of DTFT. Thus, the spectrum remains the same; only does the density become higher.
- Practice 3:
  - Generate a sinusoidal signal with frequency 0.8/16 (length is 16).
  - Observe its DFT spectrum.
  - Pad zeros to have a signal length of 256 and observe its DFT spectrum again.
  - Generate a sinusoidal signal with frequency 2/16, and redo the above operations.

- As we can see, as the length of the DFT increases, the sampling frequency in the DTFT domain increases.
- As the length goes to infinity, the spectrum obtained with DFT approaches to that of DTFT.

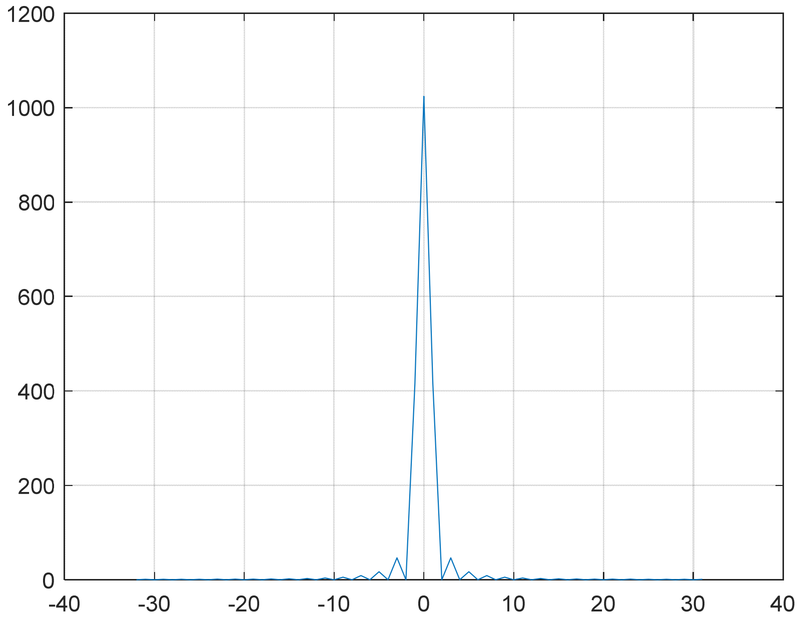
- Results:

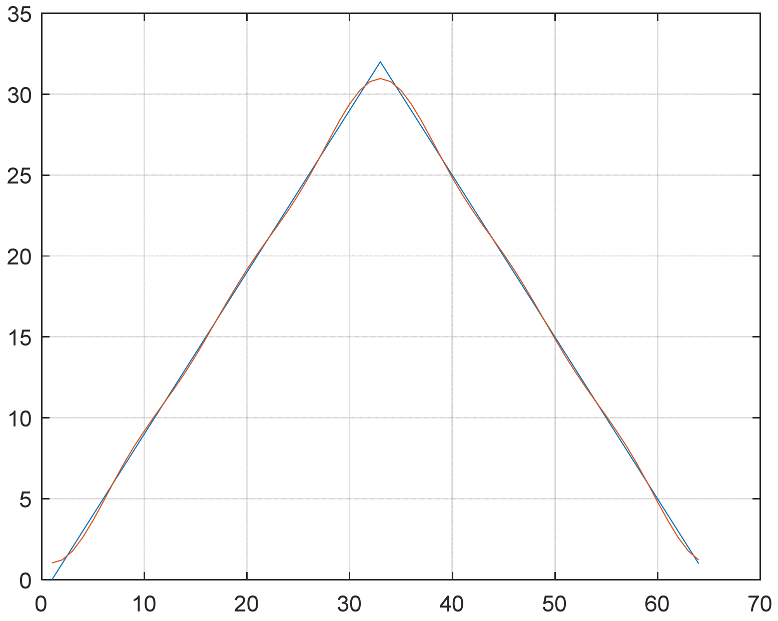


- A real discrete-time signal of length N can be represented with a summations of N/2-1 complex sinusoidals, one DC signal, and one real sinusoidal with frequency of 1/2.
  - Use DFT to obtain its spectrum.
  - Find the amplitude and phase of each sinusoidal (N/2).
  - With the sinusoidals, we can construct its time domain signal (IDFT).
- Using the representation, we can conduct compression by ignoring the frequency components of small amplitudes.
  - Transform the signal to the frequency domain.
  - Window the frequency domain signal.
  - Transform back to the time domain.

- Practice 4:
  - Generate a triangular with length 64 (peak value is 32).
  - Transform the pulse into DFT domain and observe its spectrum.
  - Conduct compression with the signal. Let 2M+1 be the number of reserved sinusoidal components.
  - Try different M values and observe the relationship of its reconstructed time domain signal and the number of the sinusoidal reserved.

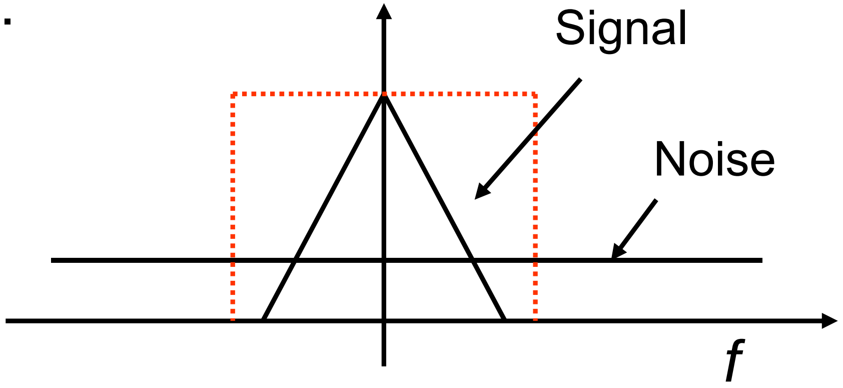
- Results:





M=5, data: 64→ 1+4x2=9

- If we know the spectrum locations of the desired and un-desired signal, we can then extract out the desired signal.
  - Put a windowing function in the DFT domain to remove undesired signal. This is equivalent to conduct an filtering operation.



* Power spectrum density (PSD)

- Indicator for processed signal quality: mean-squared-error (MSE) and signal-to-noise-ratio (SNR):

$$s(n)+v(n)\rightarrow y(n)=\overline{s}(n)+\overline{v}(n)$$

$$e(n)=\overline{s}(n)-s(n)+\overline{v}(n)=y(n)-s(n)$$

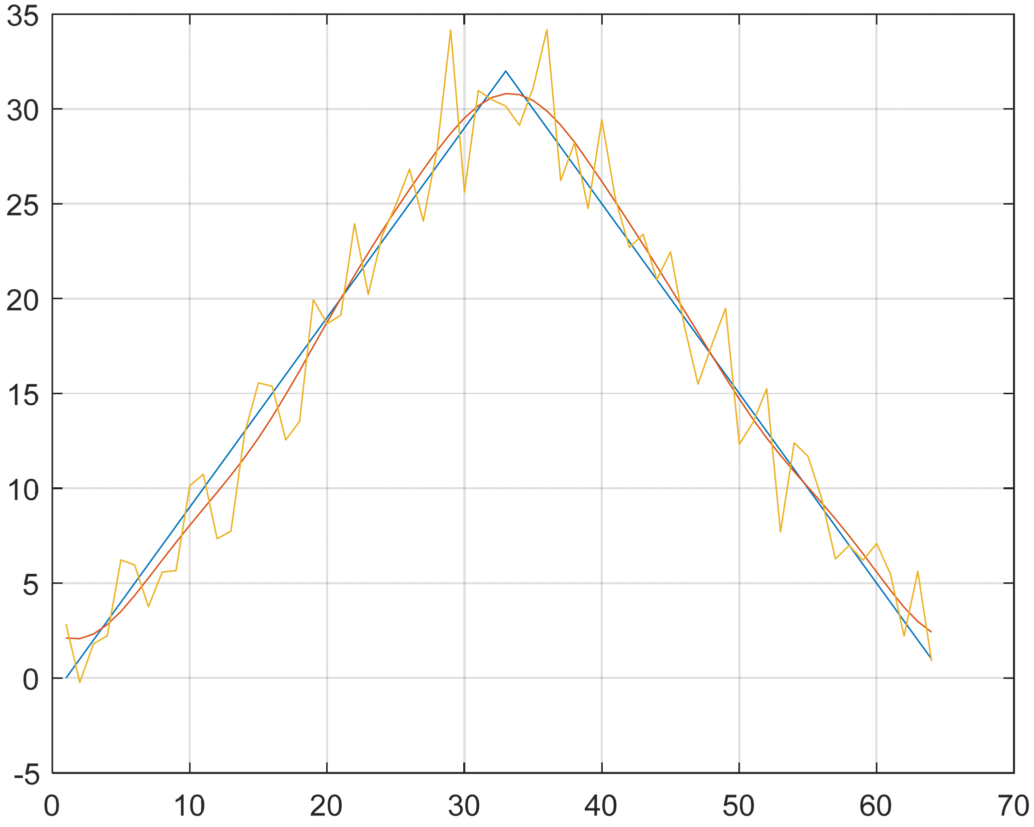
$$\text{MSE}=E\left\{\,\left|e(n)\right|^2\right\},\quad \text{SNR(dB)}=10\log 10\frac{E\left\{\,\left|\overline{s}(n)\right|^2\right\}}{E\left\{\,\left|e(n)\right|^2\right\}}$$

- Note that we may have to adjust signal output delay and filter gain to obtain the maximum SNR or minimum MSE (MMSE,).
- If a signal is random and infinite, we cannot describe it by energy spectrum. We can do that with power spectrum density (PSD).
- The total power is the variance of the random signal.

- Practice 5:
  - Add Gaussian noise (with variance of 5) to the triangular waveform used in Practice 4.
  - Recover the waveform with a frequency domain windowing operation (M=5).
  - Calculated MSE.

* Add Gaussian noise: randn(n,m)

- Results:



MSE=0.66

- How to add noise in a communication system?
  - Set a target SNR (dB)
  - Calculate the standard deviation of the noise
  - Generate the noise sequence and add it to the signal

$$\mathrm{SNR} = 10\log 10\left(\frac{\sigma_s^2}{\sigma_v^2}\right) \Rightarrow \sigma_v^2 = \sigma_s^2 \times 10^{-\frac{\mathrm{SNR}}{10}}$$

* $s(n) + v(n)$

σ_s^2 : signal power (variance)

σ_v^2 : noise power (variance)

e.g. :

$s = 1000;$

* BPSK

$x = \operatorname{sgn}(randn(1, s));$

$v = \sqrt{\sigma_v^2} \times randn(1, s);$

$y = x + v;$

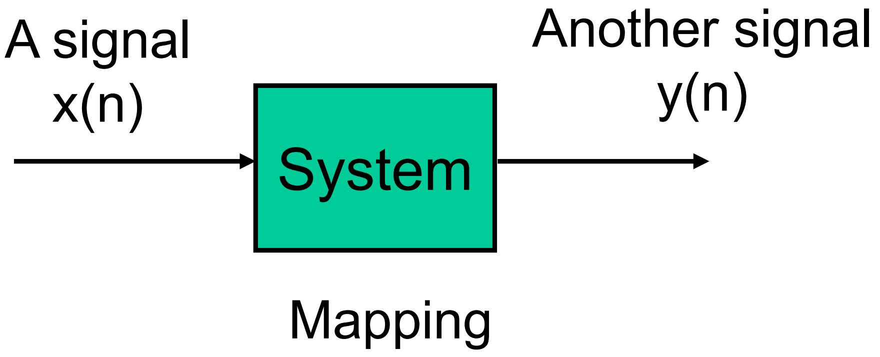
- For complex signal, the noise is also complex. As a result, the calculated variance has to be divided by two for the generation of real or imaginary noise.

- Homework 1:
  - A sinusoidal signal has a frequency of 1MHz. It is sampled with a frequency of 4MHz. Find the discrete frequency of the sampled signal.
  - Plot its spectrum with a DFT of size 256.
  - If the signal is sampled with 1.5MHz, find the frequency of the sampled signal.
  - Plot its spectrum with a DFT of size 256.
- Homework 2:
  - Generate a triangular signal (with length 128, peak 64) and add Gaussian noise to yield an SNR of 15dB.
  - Using the frequency windowing technique to conduct a filtering operation (conducting 128-point DFT).
  - Generate 100 noise sequences and add to the signal.
  - Find an optimum windowing function such that the averaged MSE of the filtered signal is minimized.

- Reading assignments:
  - Systems
  - Transfer functions (z-transform)

Lab. 3 Systems

- System: mapping of a signal



- Classification of systems:
  - Memoryless/with memory
  - Linear/nonlinear
  - Time invariant/time variant
  - Causal/noncausal

- Memoryless/with memory
  - Memoryless:

$$y(n) = f\left[x(n)\right]$$

  - With memory:

$$y(n) = f\left[\cdots, x(n+1), x(n), x(n-1), \cdots\right]$$

- Linear/nonlinear:
  - The mapping satisfies the supperposition principle or not.

$$\begin{array}{l} x_1(n) \to y_1(n) \\ x_2(n) \to y_2(n) \end{array} \quad ➡ \quad ax_1(n) + bx_2(n) \to ay_1(n) + bx_2(n)$$

- Time invariant/time variant:
  - The mapping function is variant with time or not

$$y(n) = f\left[\cdots, x(n+1), x(n), x(n-1), \cdots\right]$$

$$y(n) = f_n\left[\cdots, x(n+1), x(n), x(n-1), \cdots\right]$$

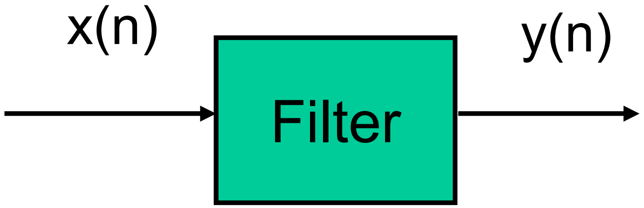
- Causal/noncausal:
  - Causal

$$y(n) = f\left[x(n), \underbrace{x(n-1), \cdots}_{Past}\right]$$

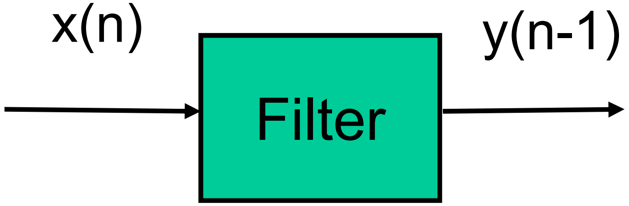
  - Noncausal

$$y(n) = f\left[\underbrace{\cdots, x(n+1)}_{Future}, x(n), \underbrace{x(n-1), \cdots}_{Past}\right]$$

- All the real-world systems are causal.
  - How do you know the future input?
- However, the man-made systems can be noncausal.
  - Put delays at the output.



$$y(n) = f\left[\underbrace{x(n+1)}_{Future}, x(n), \underbrace{x(n-1)}_{Past}\right]$$



$$y(n-1) = f\left[\underbrace{x(n)}_{Future}, x(n-1), \underbrace{x(n-2)}_{Past}\right]$$

Delay

Reference time

- LTI systems: linear and time-invariant
  - The output is a linear combination of the input values
  - Can be causal or noncausal

$$
\begin{aligned}
y(n) &= f\left[\cdots, x(n+1), x(n), x(n-1), \cdots\right] \\
&= \cdots + \underbrace{h(-1)}_{}x(n+1) + \underbrace{h(0)}_{}x(n) + \underbrace{h(1)}_{}x(n-1) + \cdots
\end{aligned}
$$

  - e.g.

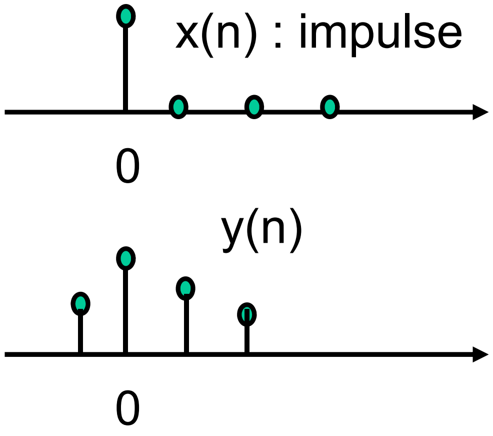
$$y(n) = h(-1)x(n+1) + h(0)x(n) + h(1)x(n-1) + h(2)x(n-2)$$

$$y(-1) = h(-1)x(0) = h(-1)$$

$$y(0) = h(0)x(0) = h(0)$$

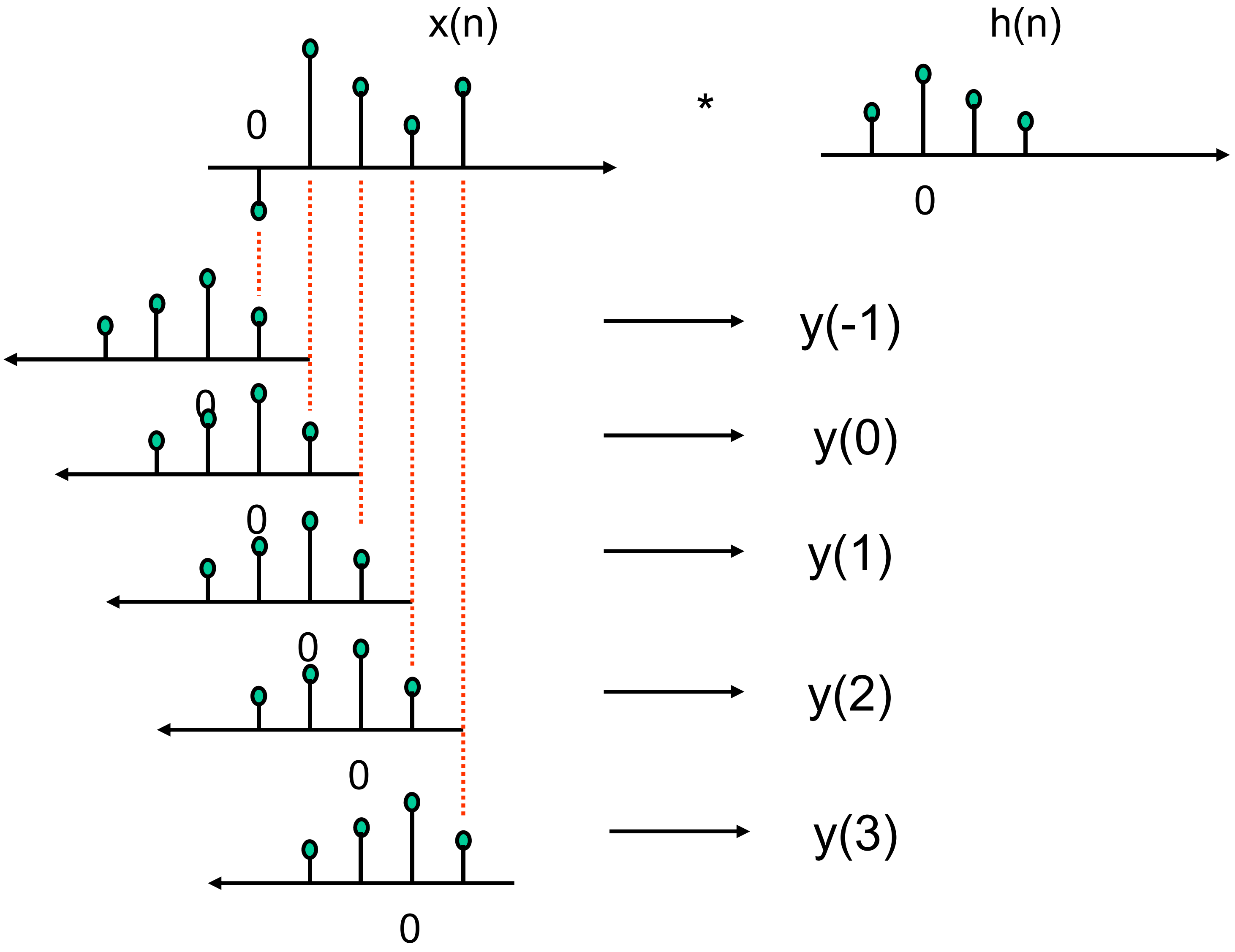
$$y(1) = h(1)x(0) = h(1)$$

$$y(2) = h(2)x(0) = h(2)$$



➡ Impulse response

- Convolution: y(n)=x(n)*h(n)



- Finite impulse response (FIR) and infinite impulse response (IIR) systems.
  - e.g.

* Difference equation

FIR: $$y(n) = a(0)x(n) + a(1)x(n-1) + a(2)x(n-2)$$

IIR: $$y(n) = b(1)\boxed{y(n-1)} + a(0)x(n) + a(1)x(n-1)$$

- Z-transform of an FIR system:

$$\begin{aligned} Y(z) &= a(0)X(z) + a(1)X(z)z^{-1} + a(2)X(z)z^{-2} \\ &= \left[a(0) + a(1)z^{-1} + a(2)z^{-2}\right]X(z) \end{aligned}$$

- The transfer function is then

$$\frac{Y(z)}{X(z)} \triangleq H(z) = a(0) + a(1)z^{-1} + a(2)z^{-2}$$

- Z-transform of an IIR system:

$$Y(z) = b(1)Y(z)z^{-1} + a(0)X(z) + a(1)X(z)z^{-1}$$
$$\Rightarrow \left[1 - b(1)z^{-1}\right]Y(z) = \left[a(0) + a(1)z^{-1}\right]X(z)$$

- The transfer function is then

$$\frac{Y(z)}{X(z)} \triangleq H(z) = \frac{a(0) + a(1)z^{-1}}{1 - b(1)z^{-1}}$$

- General form of transfer function

$$H(z) = \frac{a(0) + a(1)z^{-1} + a(2)z^{-2} + \cdots a(N)z^{-N}}{1 + b(1)z^{-1} + b(2)z^{-2} + \cdots b(M)z^{-M}}$$

zero

$$= \frac{(1 - a_1 z^{-1})(1 - a_2 z^{-1})\cdots(1 - a_N z^{-1})}{(1 - b_1 z^{-1})(1 - b_2 z^{-1})\cdots(1 - b_M z^{-1})}$$

Pole

- Transfer function ↔ difference equation
  - Transfer function: for analysis
  - Difference equation: for implementation

- Practice 1:
  - Given a signal (S3P1.mat) and the impulse response ([1 2 3 4 -2 -1]) of an FIR system, conduct the convolution operation (cannot use conv(.) command).
- Practice 2:
  - Given a signal (S3P2.mat) and a difference equation of an IIR system (see below), find the output of the system (cannot use filter(.) command).

$$y(n) = 0.5y(n-1) + y(n-2) + x(n) + 2x(n-1) + 3x(n-2)$$

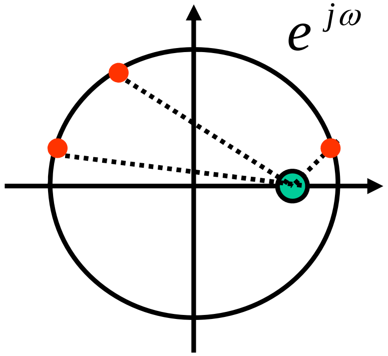
- Frequency response of systems:
  - We can know the characteristics of the frequency response of a system from the positions of TF zeros and poles.

$$H(z) = 1 - a_1 z^{-1} \rightarrow H(e^{j\omega}) = 1 - a_1 e^{-j\omega}$$

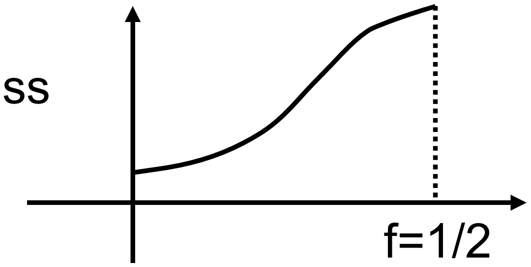
$$\left|H(e^{j\omega})\right| = \left|e^{-j\omega}(e^{j\omega} - a_1)\right| = \left|e^{j\omega} - a_1\right|$$

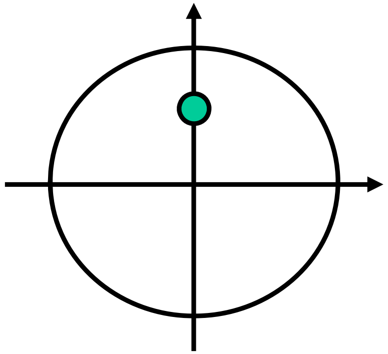
Zero:

* Frequency response

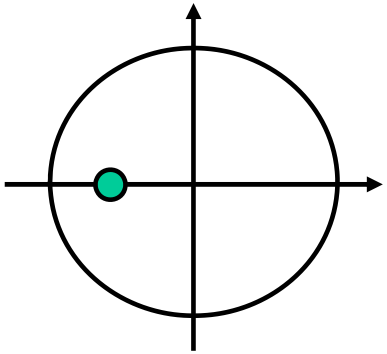


Highpass

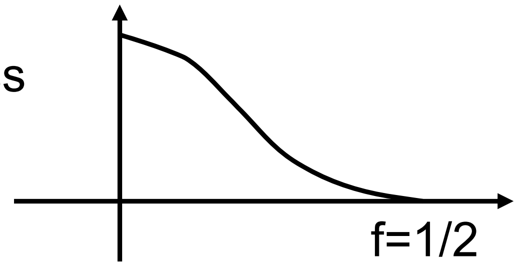




???



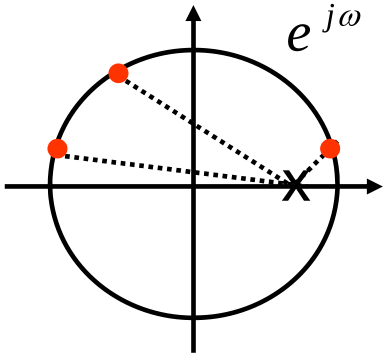
Lowpass



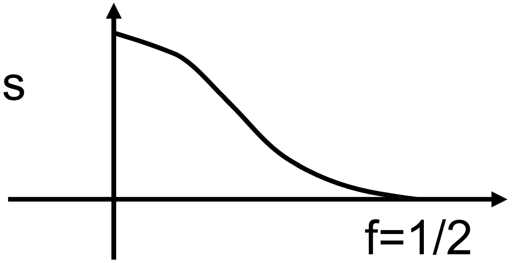
- For a pole:

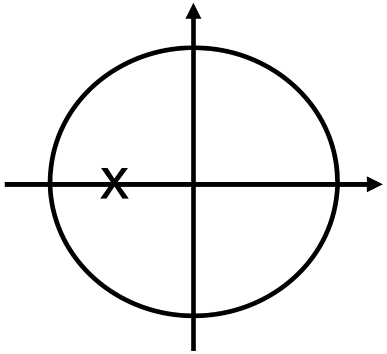
$$H(z) = \frac{1}{1 - b_1 z^{-1}} \rightarrow H(e^{j\omega}) = \frac{1}{1 - b_1 e^{-j\omega}}$$

$$\left|H(e^{j\omega})\right| = \frac{1}{\left|1 - b_1 e^{-j\omega}\right|} = \frac{1}{\left|e^{j\omega} - b_1\right|}$$

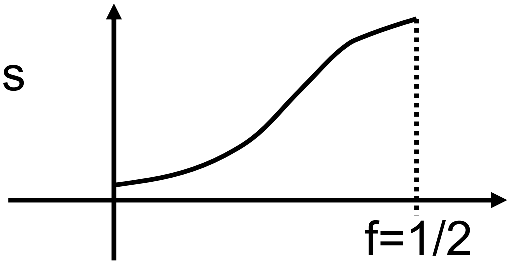


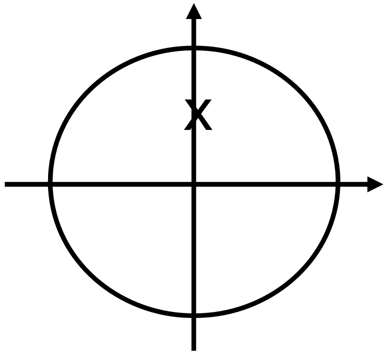
Lowpass





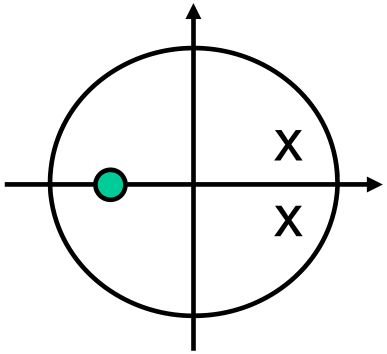
Highpass





???

- So, we can design a filter (removes undesirable frequency components) by placing zeros and poles in appropriate z-plane location.
- For example, if we want to design a lowpass filter.



$$H(z) = \frac{1+0.9z^{-1}}{\left(1-(0.8+0.3j)z^{-1}\right)\left(1-(0.8-0.3j)z^{-1}\right)}$$

$$= \frac{1+0.9z^{-1}}{1-1.6z^{-1}+0.73z^{-2}}$$

- We then have the difference equation as

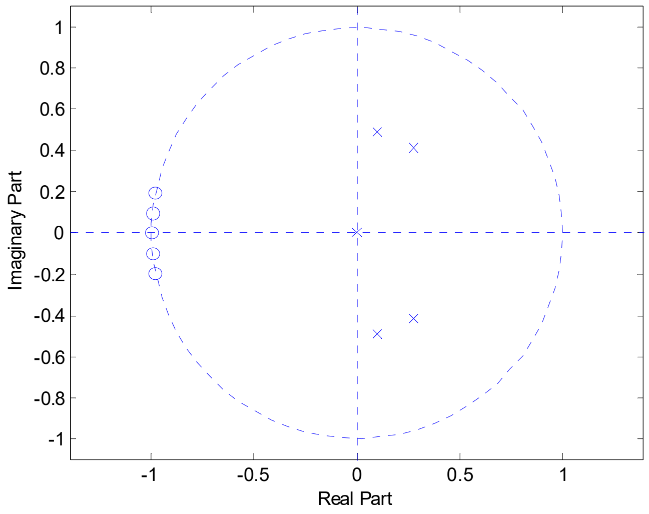
$$\frac{Y(z)}{X(z)} = \frac{1+0.9z^{-1}}{1-1.6z^{-1}+0.73z^{-2}} \Leftrightarrow$$

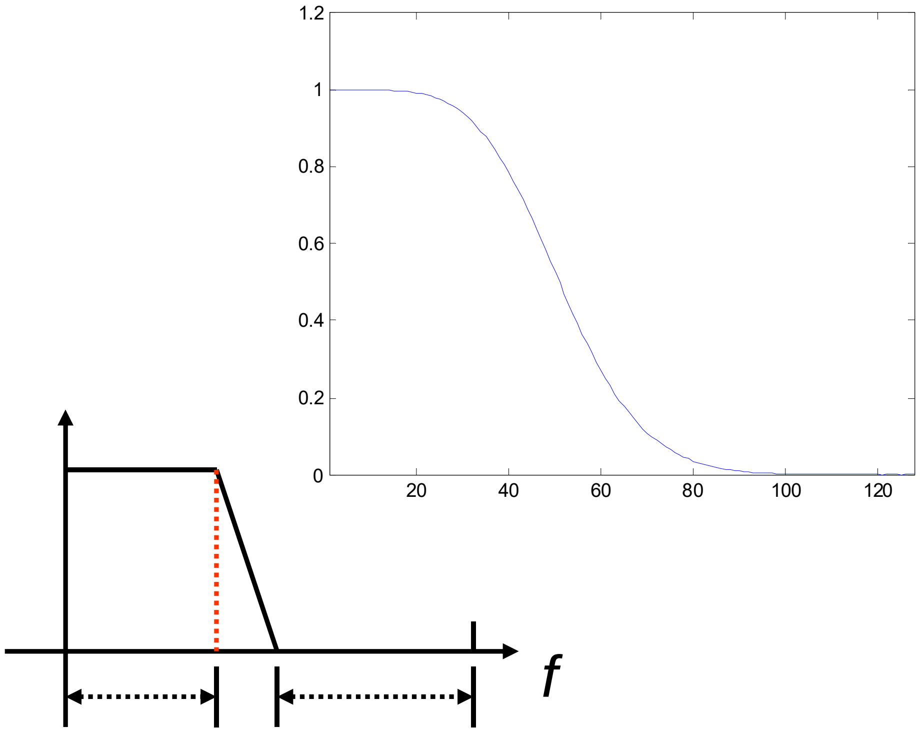
$$Y(z) - 1.6Y(z)z^{-1} + 0.73Y(z)z^{-2} = X(z) + 0.9X(z)z^{-1} \Leftrightarrow$$

$$y(n) - 1.6y(n-1) + 0.73y(n-2) = x(n) + 0.9x(n-1) \Leftrightarrow$$

$$y(n) = 1.6y(n-1) - 0.73y(n-2) + x(n) + 0.9x(n-1)$$

- Design guidelines:
  - Since zeros are simpler to control, place zeros on the unit circle around the stopband.
  - If the passband is not flat enough, put poles on a circle (with a radius smaller than one) around the passband.





Passband Stopband

- Filter response:
  - In many filter designs, only amplitude response is considered (not distorted in the passband).
  - It is desired that the phase response is not distorted either.
- If both response are not distorted, the input and output (in the passband only differs by a delay.
- The delay means the filter has a linear-phase response.

$$\sin(\omega n) \leftrightarrow \sin(\omega(n-N)) = \sin(\omega n - \underbrace{N\omega}_{phase}) = \sin(\omega n + \theta(\omega))$$

$$\theta(\omega) = -N\omega \rightarrow \text{Linear phase}$$

- How to determine the delay of a filter?

$$\hat{N} = \frac{\theta(\omega_c) - \theta(0)}{\omega_c - 0} \rightarrow \text{Slope of the phase response}$$

- How to evaluated the performance of a filter?
  - A signal pass a filter may cause delay and attenuation/ magnification. We do not consider the signal is distorted.
  - So, when we use MSE or SNR as a performance index. As mentioned, we may need to conduct gain normalization and delay adjustment (group delay).
  - Sometime, it may require some trial-and-errors to obtain the maximum values.
  - For a long sequence, you can only calculate the signal and noise power in the stationary region (avoiding transient).
  - You can also go to the frequency domain and calculate the power of desired signals and that of undesired ones.

$$H(e^{j0}) = G\frac{a(0)+a(1)+a(2)+\cdots a(N)}{1+b(1)+b(2)+\cdots b(M)}$$

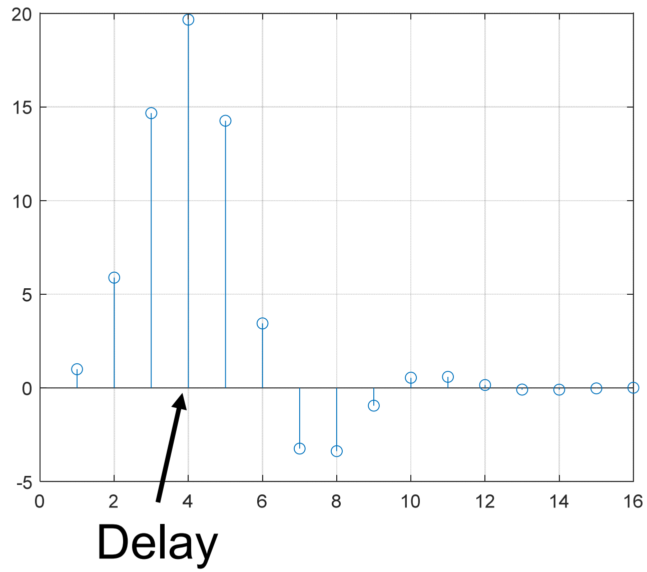
- $H(e^{j0})$: filter DC gain
- G: adjustable parameter

Practice 3:

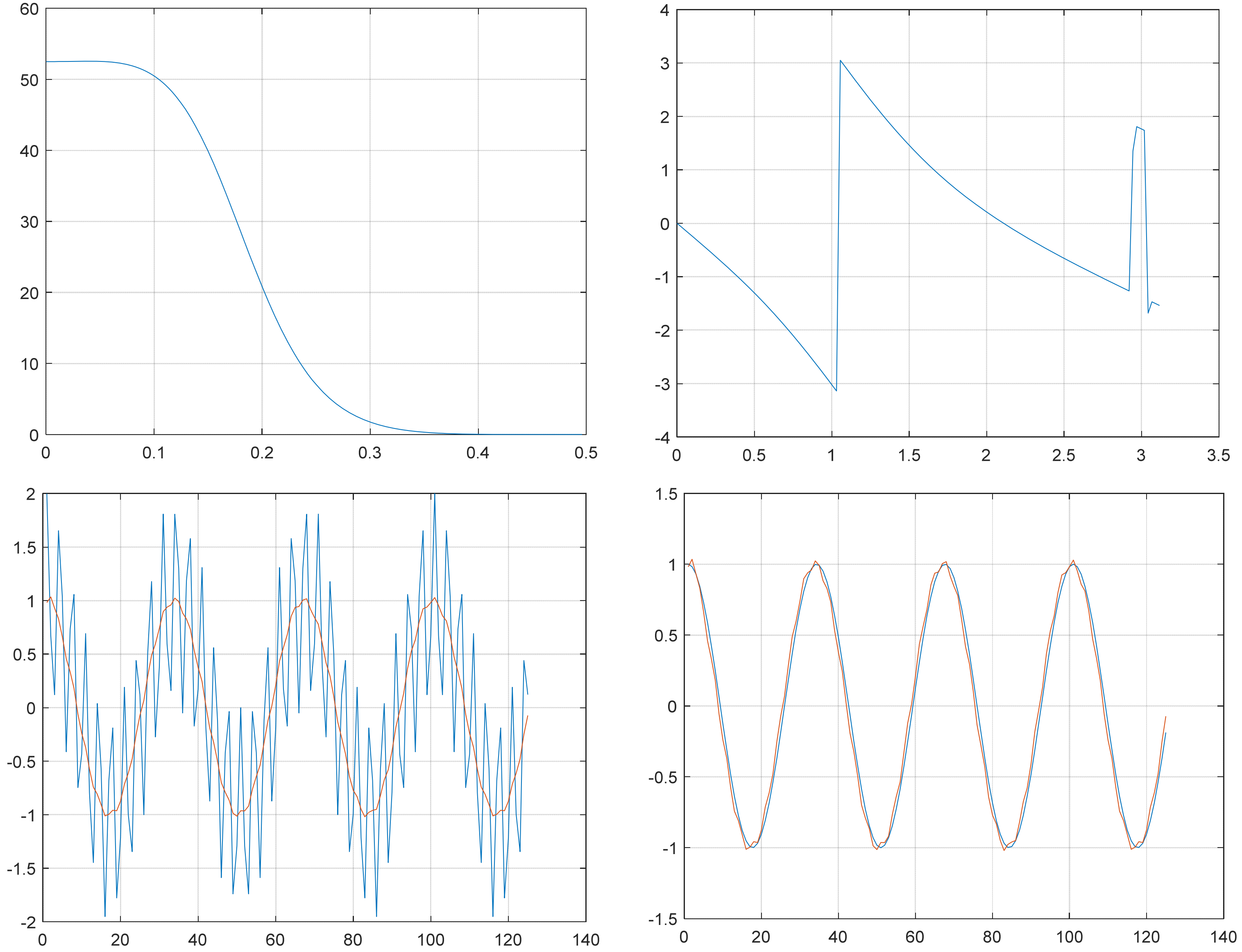
- Generate two sinusoidal signals with frequencies of 0.03 and 0.3 (length 128).
- Construct a lowpass filter with the zero-pole-placing method (given below).
- Plot the frequency response of the filter (find the delay).
- Plot the impulse responds of the filter.
- Plot the input and output signals.
- Plot the original and the filtered signals.

Zeros: -1, $\pm 31\pi/32$, $\pm 30\pi/32$
Poles: $\pm 5\pi/16$, $\pm 6\pi/16$

* Roots to polynomials: poly([x,y,z])
* Frequency response: freqz([b,a,n])
→ Only see frequency from 0 to 1/2
* Pole-zero plot: zplane(x,y)

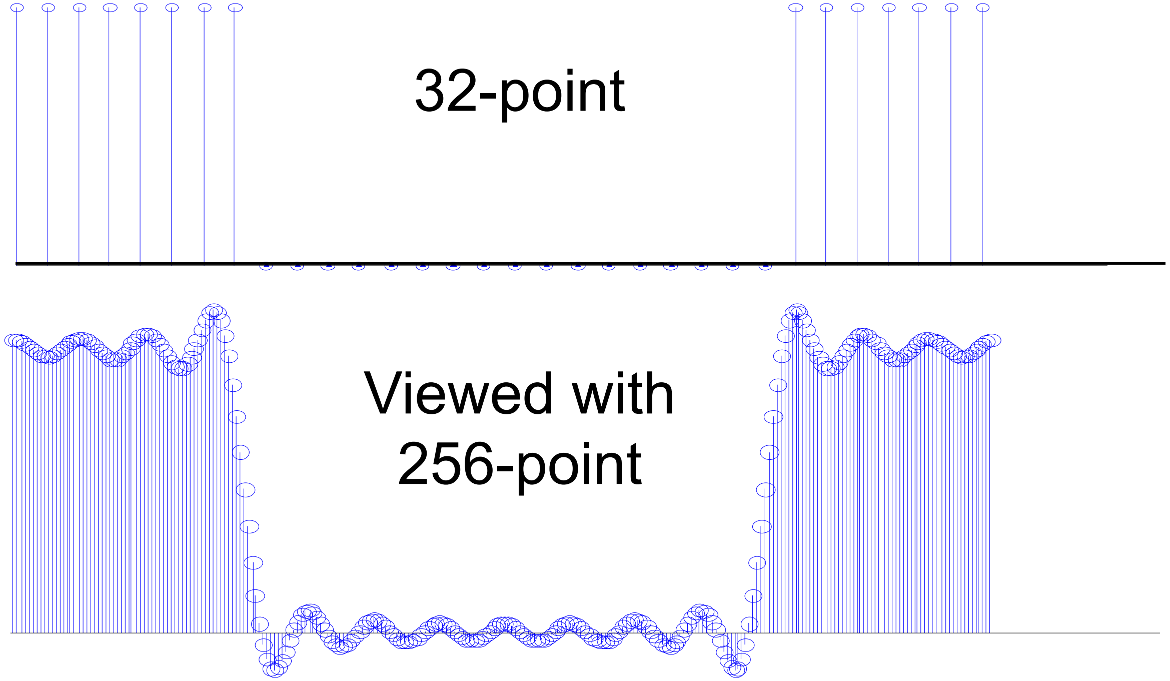


- Results:

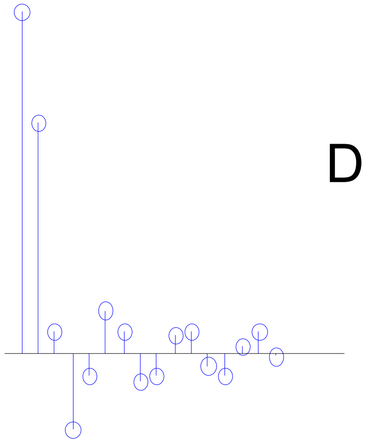


60
50
40
30
20
10
0
0
0.1
0.2
0.3
0.4
0.5
4
3
2
1
0
-1
-2
-3
-4
0
0.5
1
1.5
2
2.5
3
3.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
20
40
60
80
100
120
140
1.5
1
0.5
0
-0.5
-1
-1.5
0
20
40
60
80
100
120
140

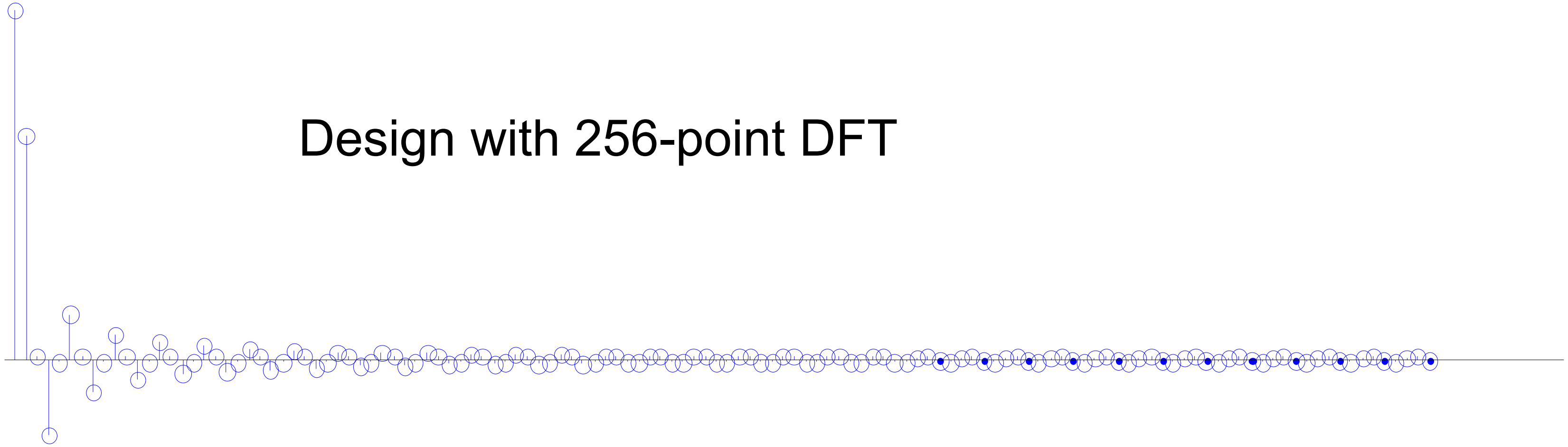
- You can also use DFT to design an FIR filter
  - Specify the frequency response of the desired filter.
  - Transform the response to the time domain.
- How does the size of the DFT influence the filter response?
  - The larger the size, the longer the time-domain response.
  - The spectrum is more smooth (close to the desired).



- Time domain response (right half):



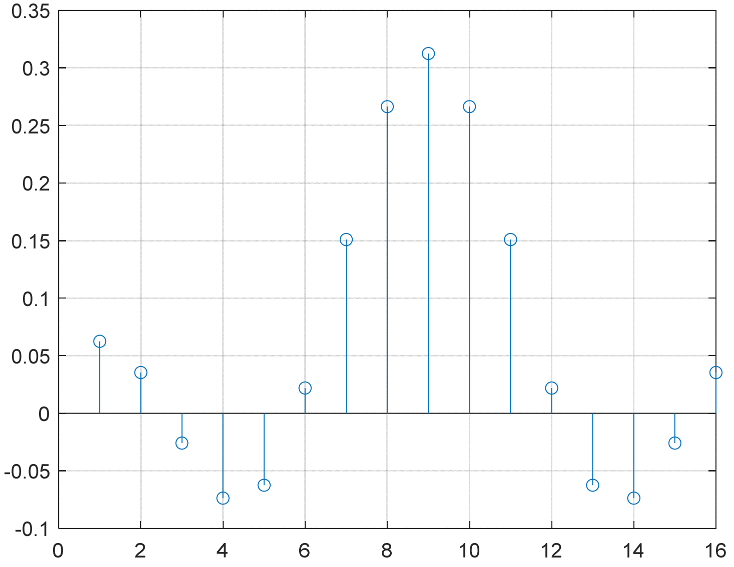
Design with 32-point DFT

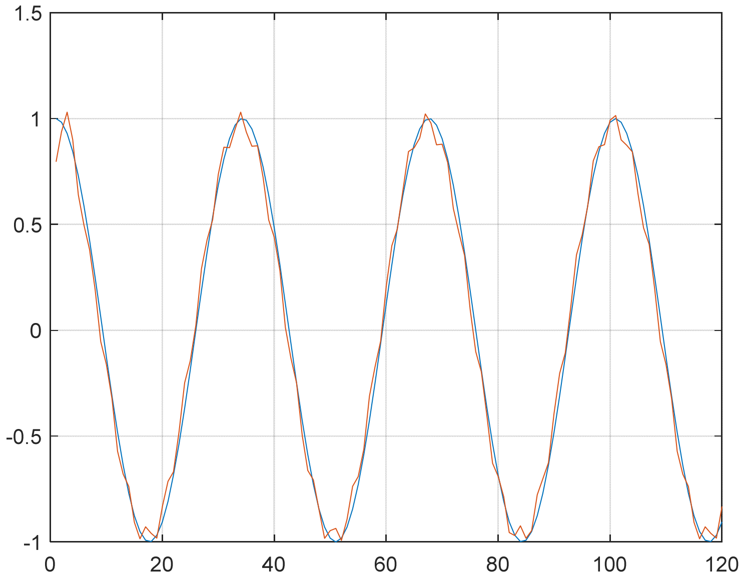


Design with 256-point DFT

- Practice 4:
  - With the sinusoidal signals in Practice 3, design a lowpass filter by the DFT method (FFT size=16).
  - Find the impulse response.
  - Find the delay of the filter.
  - Plot the original and filtered signals.

- Results:





- Homework:
  - Given two signals x and y (S3HW.mat) where y=x+w and w is noise. Using the DFT method to design an FIR filter that can filter out the noise (you can only plot the steady state result).

- Reading assignments:
  - Real modulations
  - Complex modulations

Lab. 4 Analog Modulation

- Modulation: the process of varying one or more properties of a high frequency periodic waveform, called the carrier signal, with respect to a modulating signal.
- For example:

$$y(t) = x(t)\cos(2\pi f_c t)$$

* x(t): modulating signal
* $\cos(2\pi f_c t)$: carrier
* x(t) modulates $\cos(2\pi fct)$

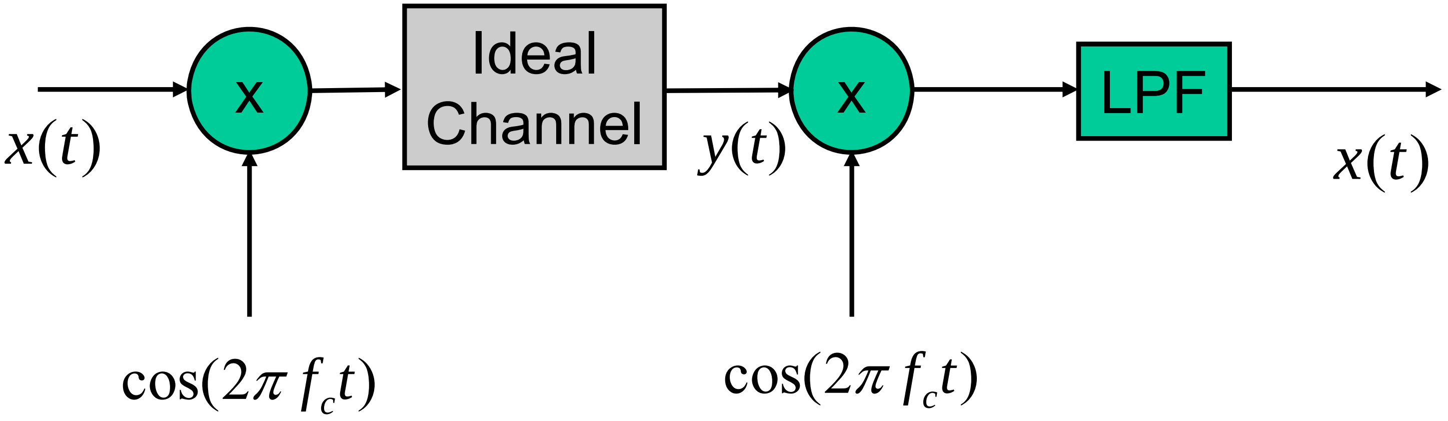
- Why modulation?
  - For transmission
  - For multiple access

- Demodulation (real):

$$y(t) = x(t)\cos(2\pi f_c t)$$

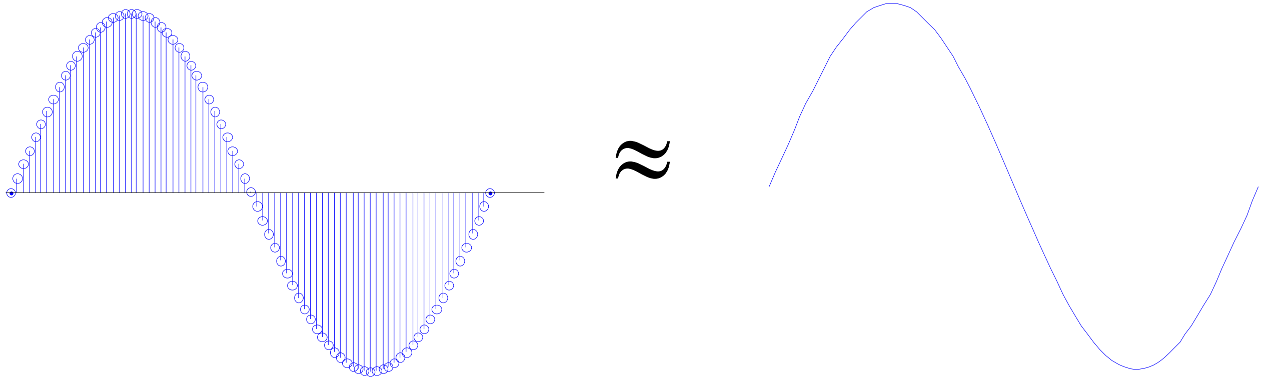
$$z(t) = y(t)\cos(2\pi f_c t) = x(t)\cos^2(2\pi f_c t) = x(t)\left(\frac{1}{2} + \cancel{\frac{\cos(4\pi f_c t)}{2}}\right)$$

$$\xrightarrow{\textit{Lowpass}} x(t)$$



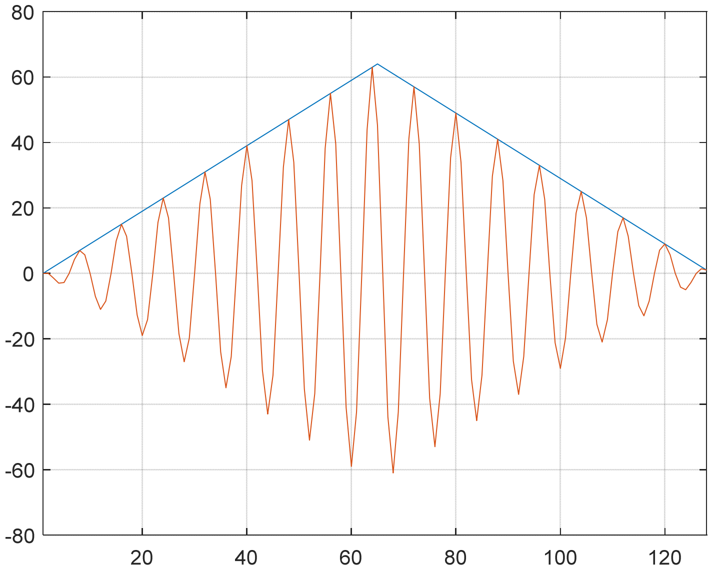
* What will happen if the carrier in the receiver has a phase offset?

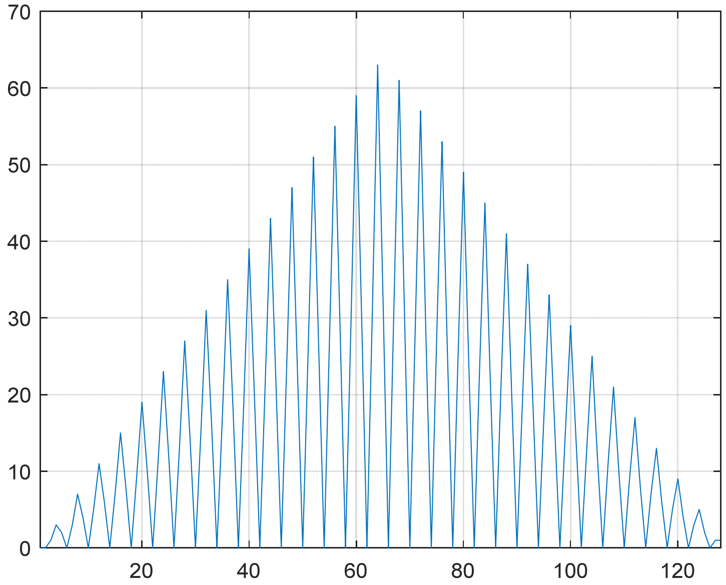
- A continuous signal can be approximated with a discrete time signal with a high sampling frequency (what we can do with computers).

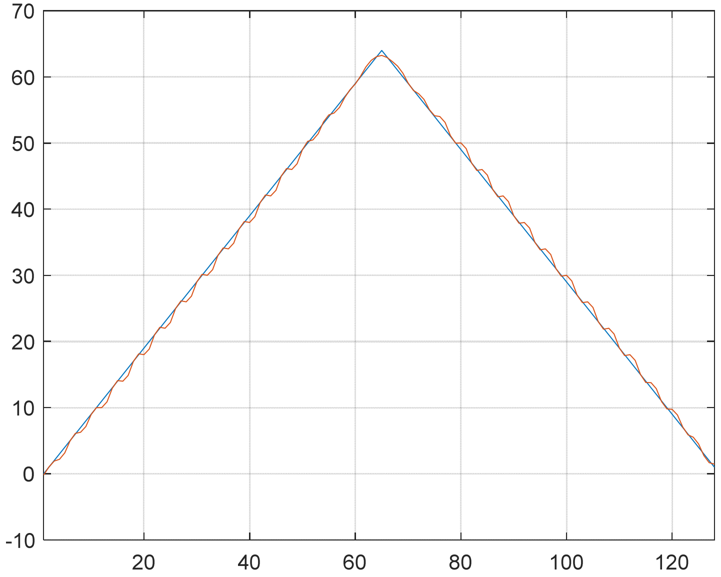


- Practice 1:
  - Generate a triangular pulse (length 128, peak value 64) and a sinusoidal carrier with frequency of 1/8.
  - Design a LPF with the DFT method (length 32).
  - Conduct the modulation and demodulation.
  - See modulated, demodulated, and filtered signals.
  - Compare the original and received pulses.

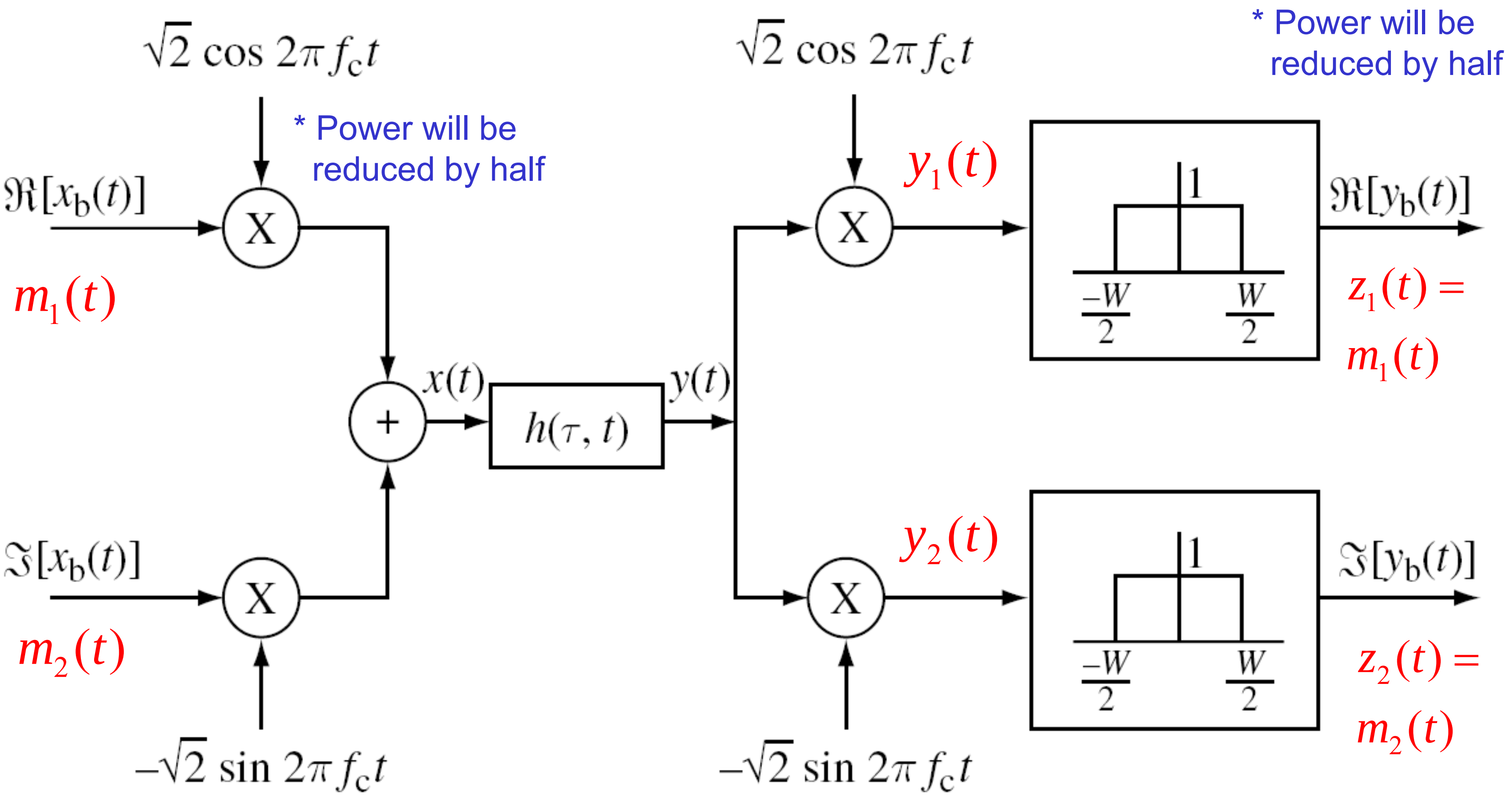
- Results:


80
60
40
20
0
-20
-40
-60
-80
20
40
60
80
100
120


70
60
50
40
30
20
10
0
20
40
60
80
100
120


70
60
50
40
30
20
10
0
-10
20
40
60
80
100
120

- Modulation with two carries (complex):



* Power will be reduced by half

* Power will be reduced by half

$$x(t) = \sqrt{2}\left[m_1(t)\cos(2\pi f_c t) - m_2(t)\sin(2\pi f_c t)\right]$$

- Let y(t)=x(t), then

$$
\begin{aligned}
y_1(t) &= 2y(t)\cos(2\pi f_c t) \\
&= 2\left[m_1(t)\cos(2\pi f_c t) - m_2(t)\sin(2\pi f_c t)\right]\cos(2\pi f_c t) \\
&= m_1(t)\left[1+\cancel{\cos(4\pi f_c t)}\right] - m_2(t)\cancel{\sin(4\pi f_c t)} \\
&= m_1(t) \quad \text{(after LPF)}
\end{aligned}
$$

$$
\begin{aligned}
y_2(t) &= -2y(t)\sin(2\pi f_c t) \\
&= -2\left[m_1(t)\cos(2\pi f_c t) - m_2(t)\sin(2\pi f_c t)\right]\sin(2\pi f_c t) \\
&= -m_1(t)\cancel{\sin(4\pi f_c t)} + m_2(t)\left[1-\cancel{\cos(4\pi f_c t)}\right] \\
&= m_2(t) \quad \text{(after LPF)}
\end{aligned}
$$

- The two signals can be transmitted, simultaneously, and received without mutual interference.

- Construct a complex transmit signal as $m(t) = m_1(t)+j\, m_2(t)$, and a complex carrier as $c(t)=\cos(2\pi f_c t)+j\,\sin(2\pi f_c \mathrm{t})$.
- The transmitted signal can then be obtained as

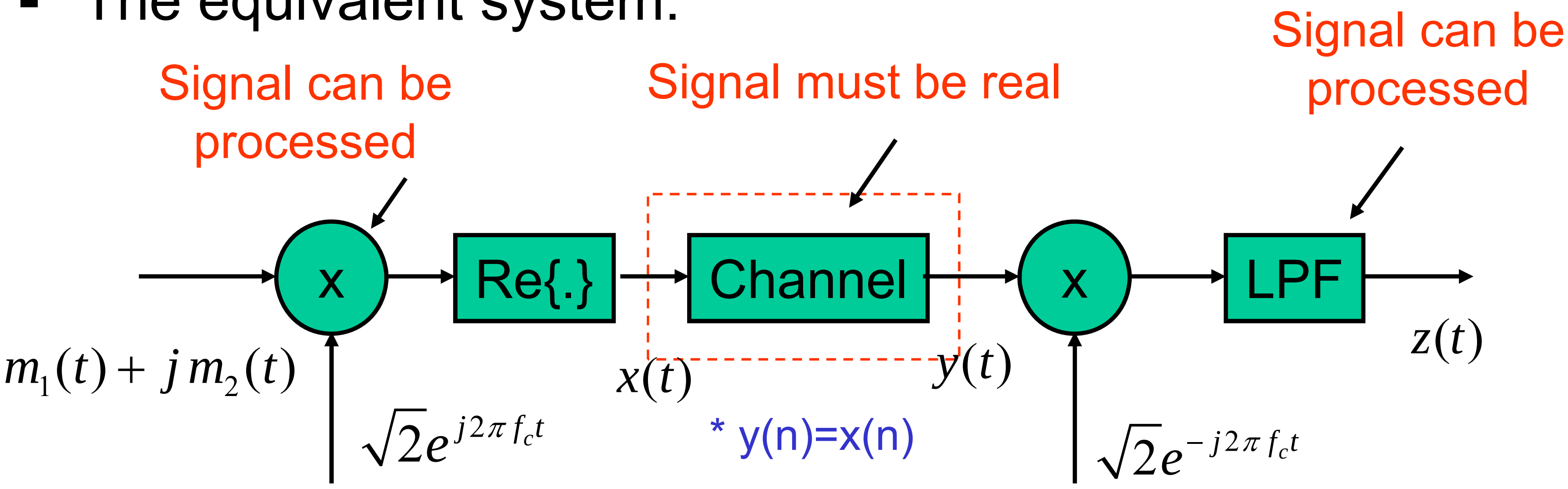
$$\begin{aligned} x(t) &= \sqrt{2}\,\mathrm{Re}\left[m(t)c(t)\right] \\ &= \sqrt{2}\left[m_1(t)\cos(2\pi f_c t) - m_2(t)\sin(2\pi f_c t)\right] \end{aligned}$$

- Then, the whole system can be represented by an equivalent system with complex signal representation.
- Note that

$$e^{j2\pi f_c t} = \cos(2\pi f_c t) + j\sin(2\pi f_c t)$$

(To see the story of how complex number is found, see the Youtube vedio in https://www.youtube.com/watch?v=DwHpAyKB1-g&t=3s)

- The equivalent system:



Signal can be processed

Signal must be real

Signal can be processed

x

Re{.}

Channel

x

LPF

$m_1(t) + j\,m_2(t)$

$\sqrt{2}e^{j2\pi f_c t}$

$x(t)$

$y(t)$

* y(n)=x(n)

$\sqrt{2}e^{-j2\pi f_c t}$

$z(t)$

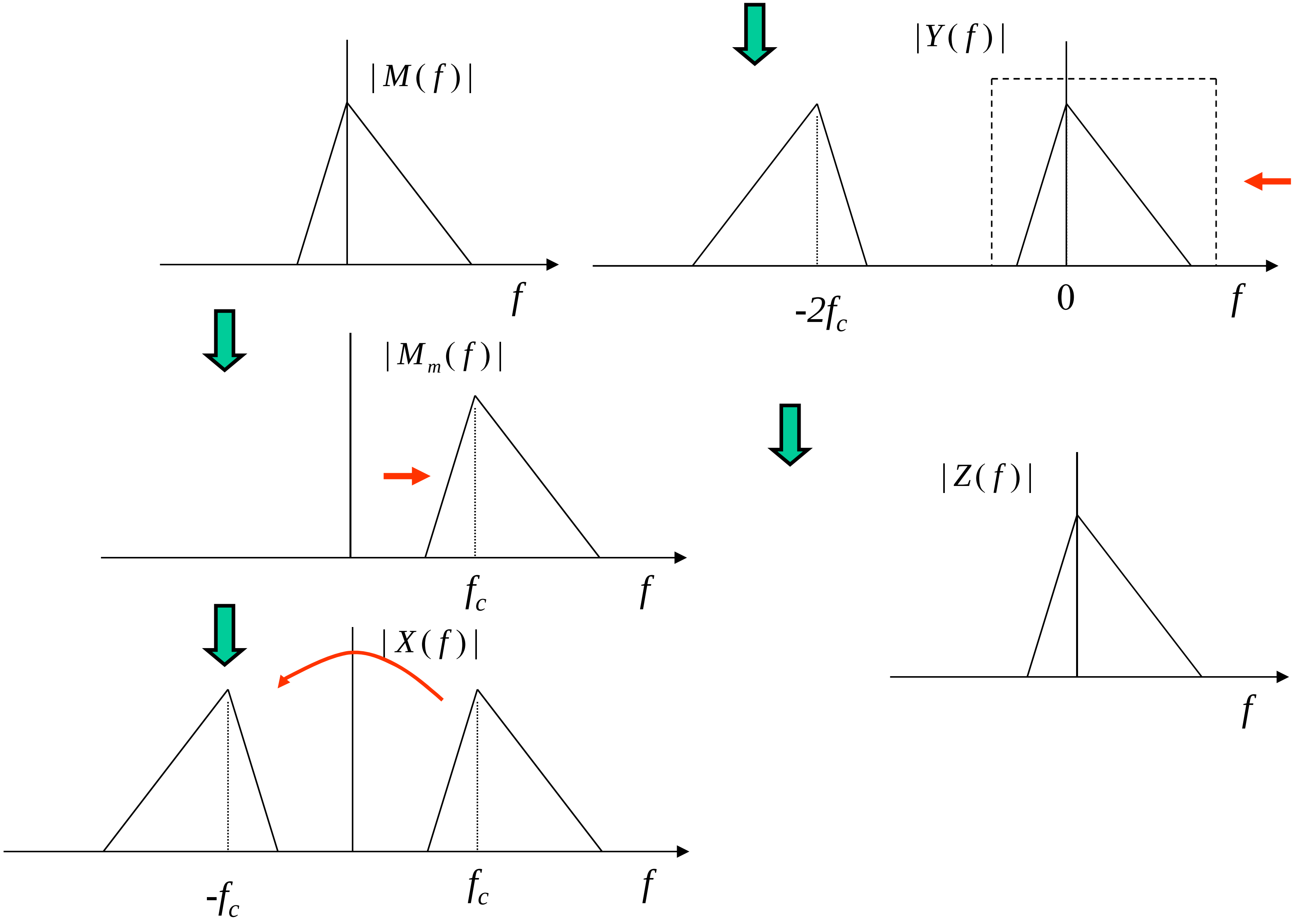
$$x(t) = \sqrt{2}\left[m_1(t)\cos(2\pi f_c t) - m_2(t)\sin(2\pi f_c t)\right]$$

$$\begin{aligned} y(t)\sqrt{2}e^{-j2\pi f_c t} &= x(t)\sqrt{2}\left(\cos(2\pi f_c t) - j\sin(2\pi f_c t)\right) \\ &= \sqrt{2}x(t)\cos(2\pi f_c t) - j\sqrt{2}x(t)\sin(2\pi f_c t) \end{aligned}$$

$$z(t) = m_1(t) + j\,m_2(t)$$

- Note that complex signal cannot be transmitted, but it can be constructed and processed at tx and rx sides!

- Spectrum properties:



- Consider the channel response is a delay, i.e, y(t)=x(t-Δt).

* The delay is much smaller than the symbol time T_s

$$y(t) = \sqrt{2}\left[m_1(t-\Delta t)\cos(2\pi f_c(t-\Delta t)) - m_2(t-\Delta t)\sin(2\pi f_c(t-\Delta t))\right]$$

$$= \sqrt{2}\left[m_1(t)\cos(2\pi f_c t+\theta) - m_2(t)\sin(2\pi f_c t+\theta)\right]$$

* $m_1(t-\Delta t) \approx m_1(t)$

$m_2(t-\Delta t) \approx m_2(t)$

- Then,

$$y_1(t) = 2\left[m_1(t)\cos(2\pi f_c t+\theta) - m_2(t)\sin(2\pi f_c t+\theta)\right]\cos(2\pi f_c t)$$

$$= m_1(t)\left[\cancel{\cos(4\pi f_c t+\theta)} + \cos(\theta)\right] - m_2(t)\left[\cancel{\sin(4\pi f_c t+\theta)} + \sin(\theta)\right]$$

$$y_2(t) = -2\left[m_1(t)\cos(2\pi f_c t+\theta) - m_2(t)\sin(2\pi f_c t+\theta)\right]\sin(2\pi f_c t)$$

$$= -m_1(t)\left[\cancel{\sin(4\pi f_c t+\theta)} - \sin(\theta)\right] + m_2(t)\left[\cancel{-\cos(4\pi f_c t+\theta)} + \cos(\theta)\right]$$

- As we can see, the output is no longer m_1(t) and m_2(t).

- The output of the demodulator, $z(t)=z_1(t)+jz_2(t)$, is

$$z_1(t) = m_1(t)\cos(\theta) - m_2(t)\sin(\theta)$$
$$z_2(t) = m_1(t)\sin(\theta) + m_2(t)\cos(\theta)$$

- If we let

$$h(0) = e^{j\theta} = \cos(\theta) + j\sin(\theta)$$

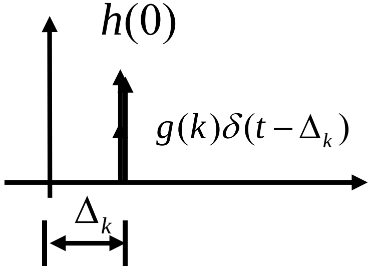
- The complex output can be represented as $z(t)=m(t)h(0)$.

$$z(t) = (m_1(t) + jm_2(t))(\cos(\theta) + j\sin(\theta)) = e^{j\theta}(m_1(t) + jm_2(t))$$

- In general, the channel has a multi-path response. There are two types of delays. The first type is that paths are close and can be merged into one coefficient:

$$h(0) = \sum_k g(k)e^{j\theta_k}, \quad \theta_k = 2\pi f_c \Delta_k$$

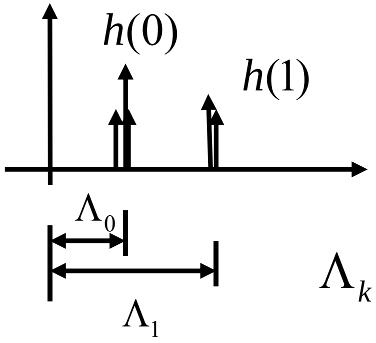
$\Delta_k \ll T_s$ (T_s: symbol time)



- The second type of delays cannot be merged into one coefficient (far apart). The channel response can be represented as:

$$h(t)=\sum_k h(k)\delta(t-\Lambda_k)$$

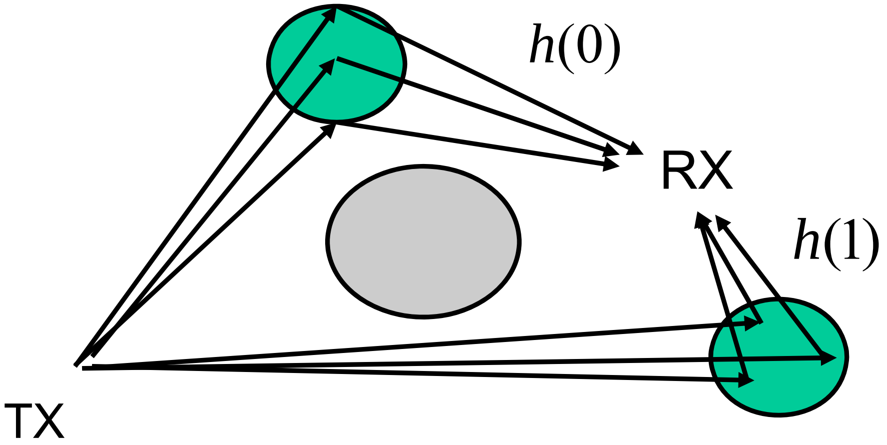
h(*k*): complex



$$\Lambda_k \approx T_s$$

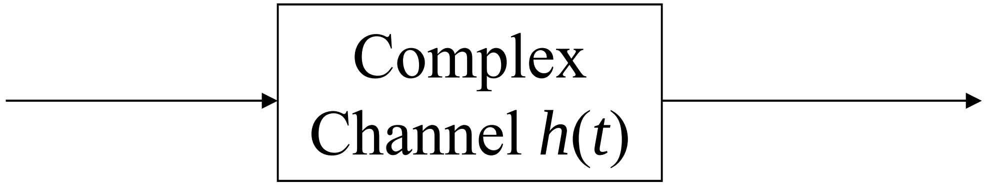
- Then

$$\boxed{z(t)=m(t)*h(t)}$$

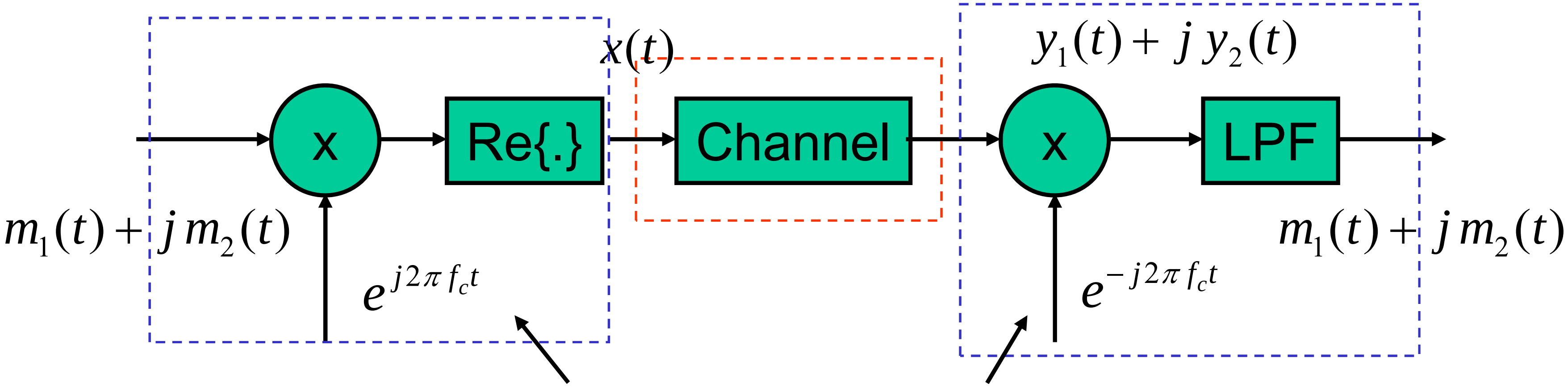


- Equivalent complex baseband representation (without involving modulation/demodulation):

$$m(t) = m_1(t) + j\,m_2(t) \qquad\qquad z(t) = z_1(t) + j\,z_2(t)$$

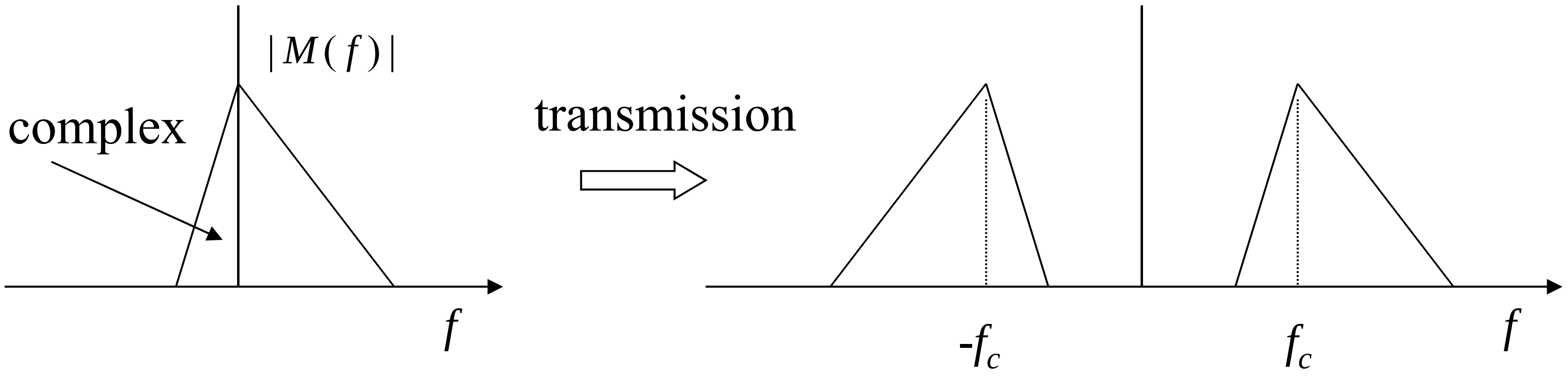


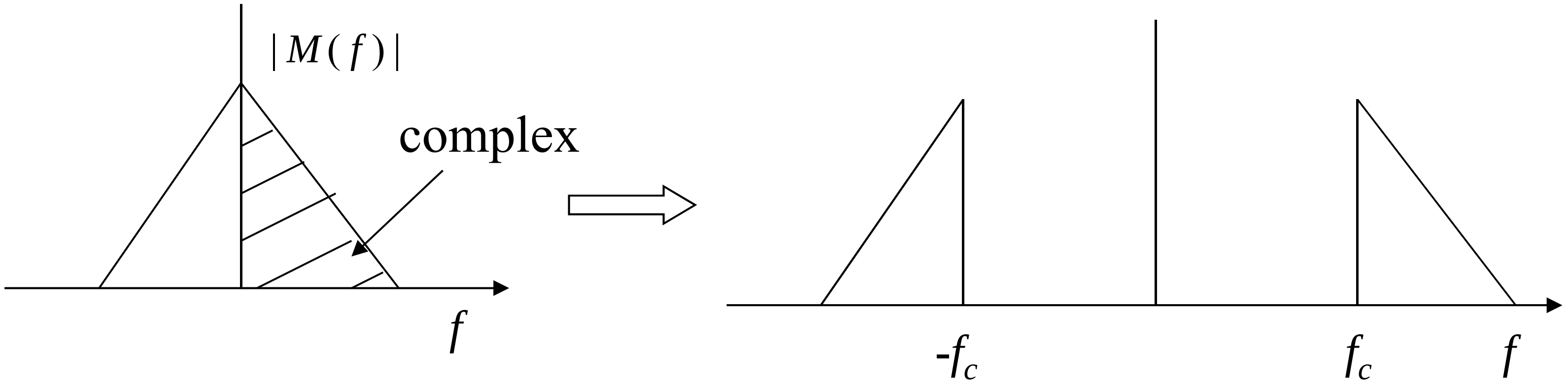
$$z(t) = m(t) * h(t)$$



We can do complex processing

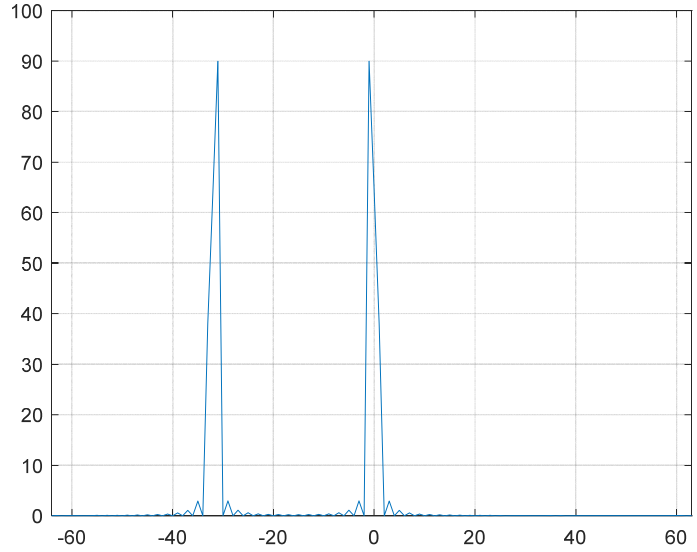
- Types of complex baseband signals:
  - Formed by two real signals
  - Extracted from a real signal



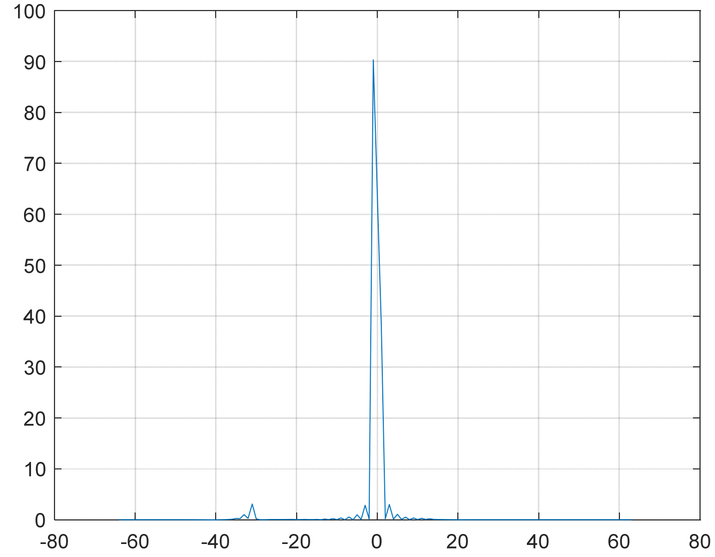


- Practice 2:
  - Simulate a complex modulation system with real operations. Let I-branch transmit a triangular pulse and Q-branch a period of sinusoidal signal ($\sin(2\pi fn)$).
    - Carrier frequency: 1/8
    - Triangular pulse: same as that in Practice 1 (peak value is one).
    - Signal length: 128
    - Sinusoidal signal: $f = 1/128$
    - Use the LPF designed in Practice 1.
  - Plot the original and recovered signals.
- Practice 3:
  - Simulate the complex modulation system in Practice 2 with complex operations.
  - Plot demodulated and filtered spectrums.
  - Plot the original and recovered signals.

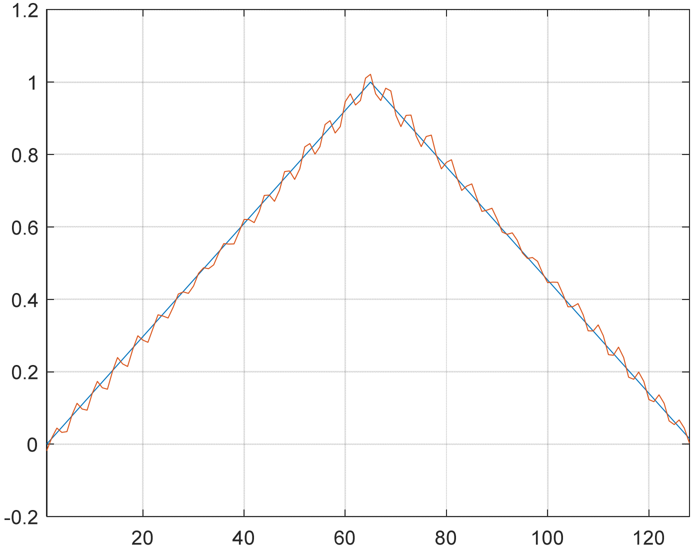
- Results:



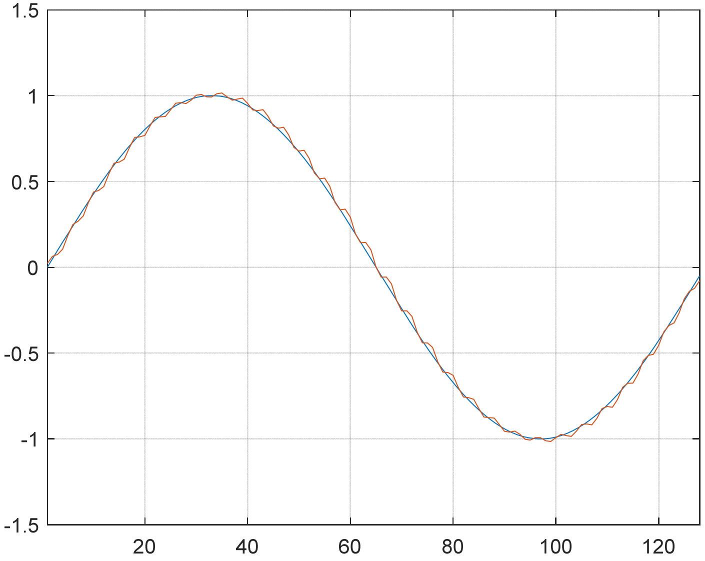
100
90
80
70
60
50
40
30
20
10
0
-60
-40
-20
0
20
40
60



100
90
80
70
60
50
40
30
20
10
0
-80
-60
-40
-20
0
20
40
60
80



1.2
1
0.8
0.6
0.4
0.2
0
-0.2
20
40
60
80
100
120



1.5
1
0.5
0
-0.5
-1
-1.5
20
40
60
80
100
120

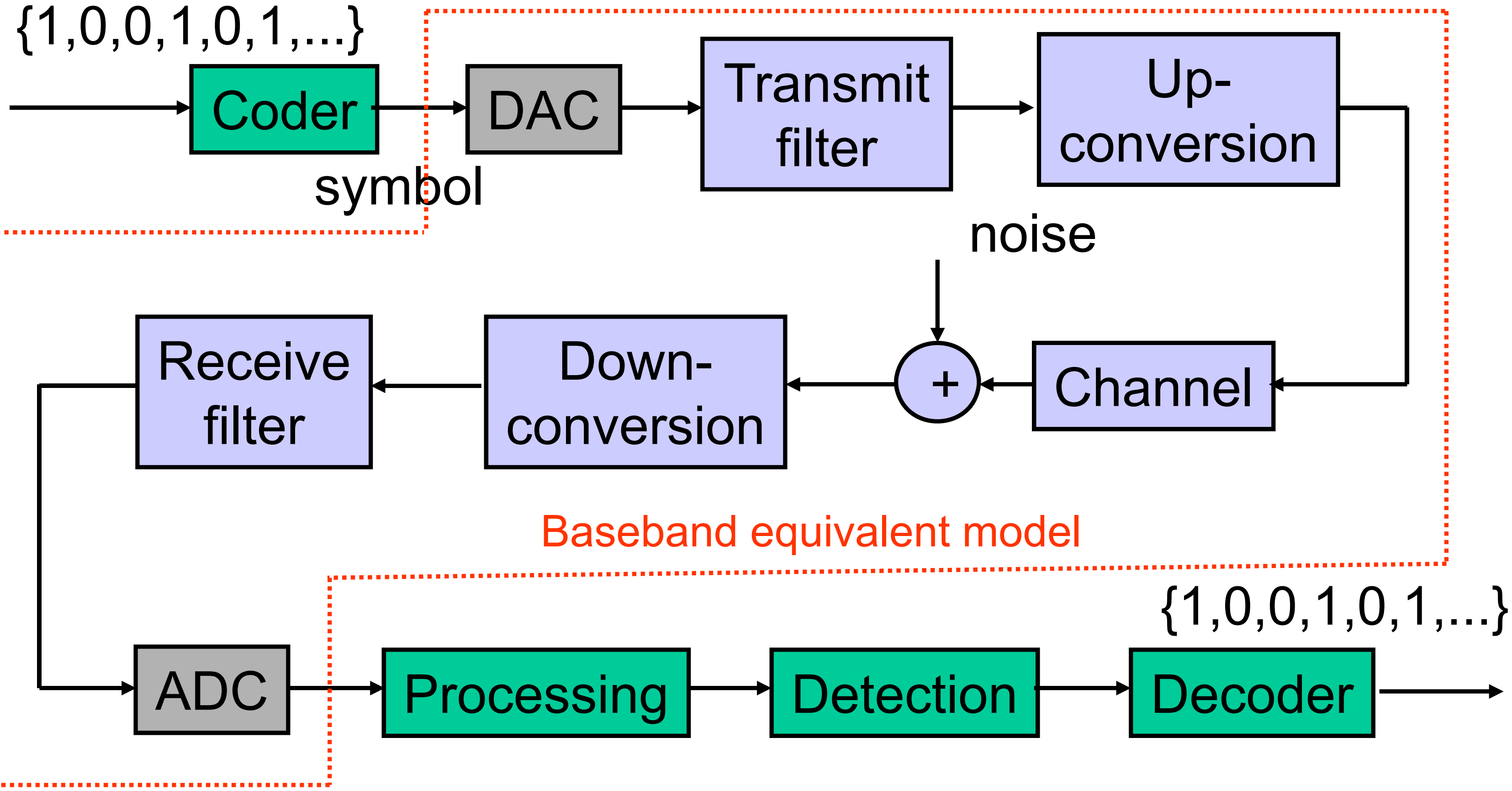
- Homework:
  - Use the result in Practice 3.
  - Assume there is a phase shift of 5° in the receiver carrier. Observe demodulated signals.
  - Compare the result with

$$z_1(t) = m_1(t)\cos(\theta) - m_2(t)\sin(\theta)$$
$$z_2(t) = m_1(t)\sin(\theta) + m_2(t)\cos(\theta)$$

- Reading assignment:
  - Digital modulation: PAM, QAM
  - AWGN
  - Error probability (Q-function)

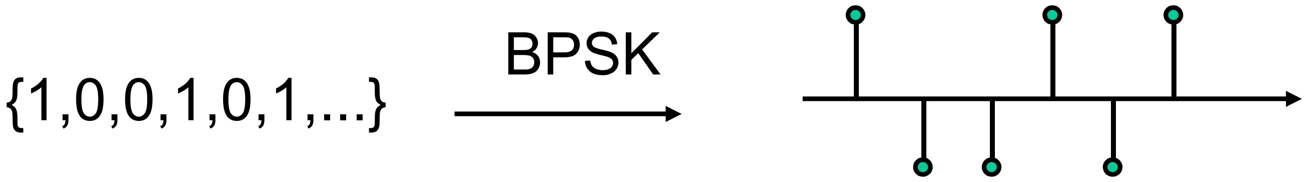
Lab. 5 Digital Modulation

- Digital modulation:



- Coder (symbol mapper):
  - Maps the input bits to a digital symbol (number).
- Digital-to-analog converter (DAC)
  - Convert digital sequence to analogy signal
- Transmit filtering (pulse shaping):
  - Maps digital symbols to an analog symbol (waveform).
- Receiver filtering (matched filtering)
  - Match transmit analog symbols (noise filtering)
- Analog-to-digital conversion (ADC):
  - Convert analogy waveform to digital sequence (sampling)
- Processing/detection
  - Process and detect transmit symbols
- Decoder:
  - Demaps the detected symbols to bits

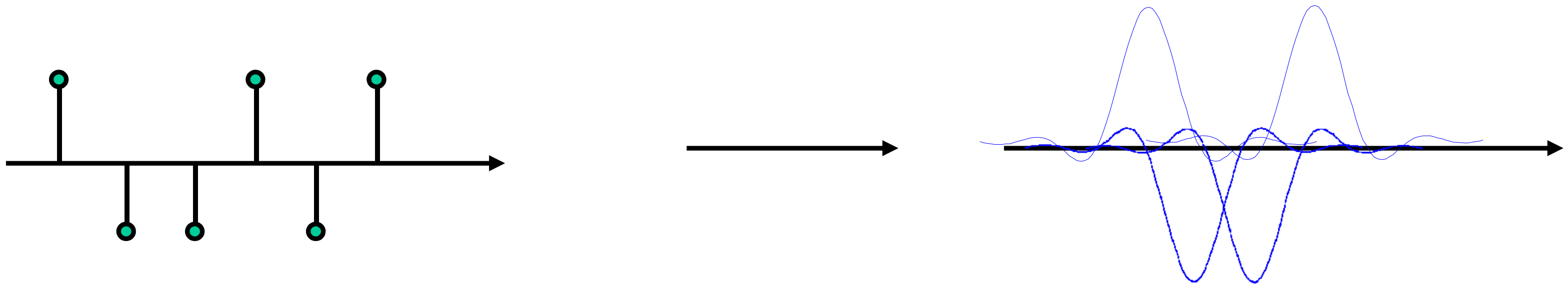
- Symbol mapping:



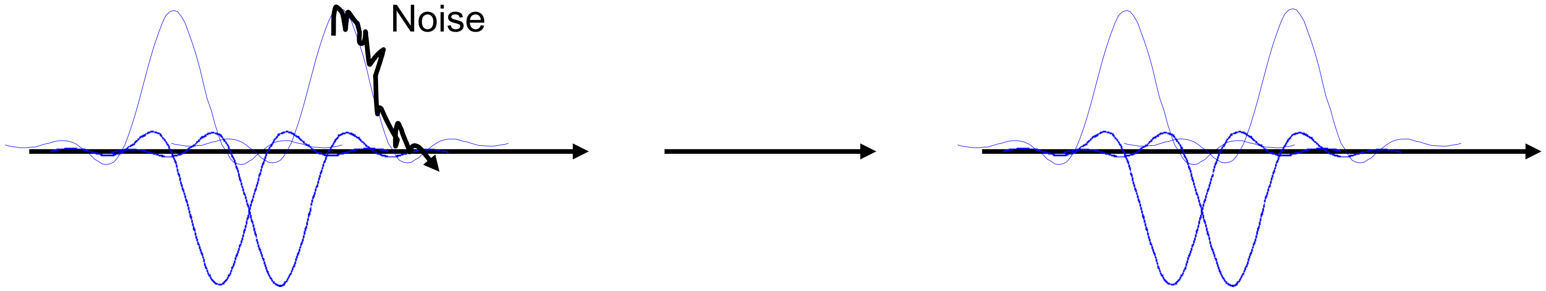
{1,0,0,1,0,1,...}

BPSK

- Transmit filtering:

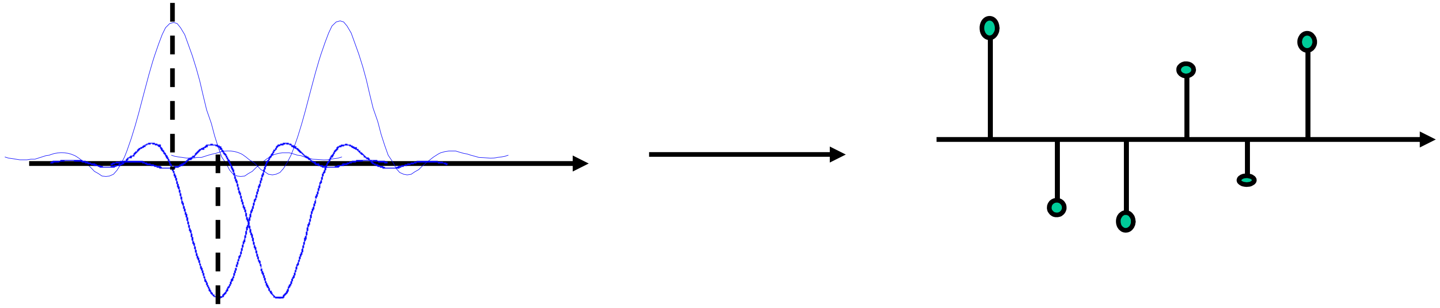


- Receive filtering:

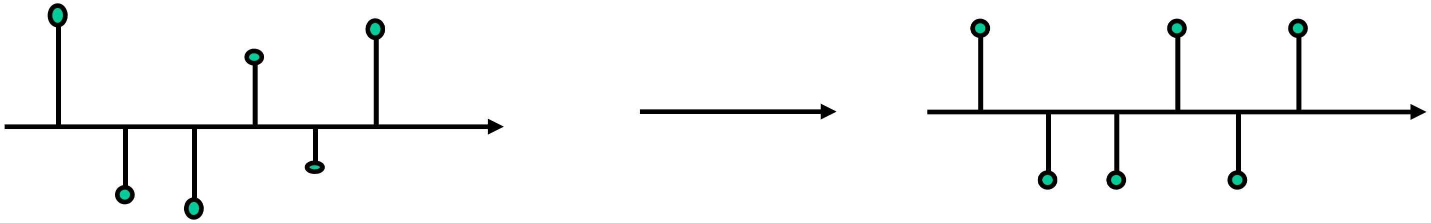


Noise

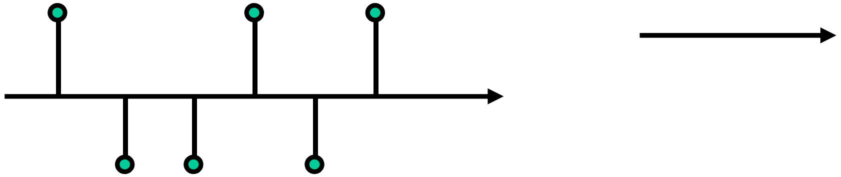
- Sampling:



- Detection:



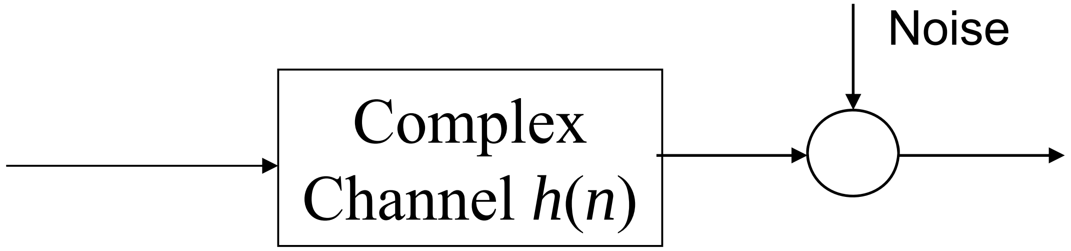
- Demapping:



{1,0,0,1,0,1,...}

- Digital equivalent baseband model:

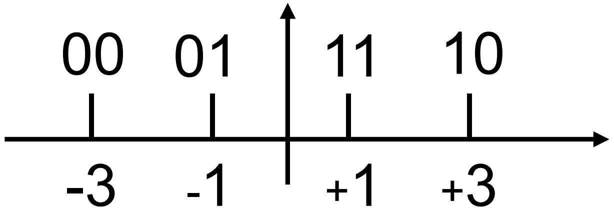
$$a(n) = a_I(n) + j\,a_Q(n) \qquad\qquad y(n) = y_I(n) + j\,y_Q(n)$$



$$y(n) = a(n) * h(n)$$

- Coder (symbol mapping):
  - PAM (±1,±3,±5,±7,..)
  - QAM (±1/±3/±5/±7,..+j ±1/±3/±5/±7)

- Symbol mapping:
  - Maps bits into symbols



- Direct mapping:
  - Ex: 4-PAM, [00 01 10 11] → [0, 1, 2, 3] → [-3, -1, 1, 3]
  - Bit error due to [-3,-1], [1,3] is one.
  - Bit errors due to [-1,1] is two.
- Gray mapping:
  - Neighboring symbols only differs by one bit
  - Minimize bit error rate (one bit error for all symbols)
  - 4-PAM: [00 01 11 10] → [0, 1, 2, 3] → [-3, -1, 1, 3]
- Let an N-bit binary and Gray mapped bit sequences be denoted by B(n) and G(n). Then,

$$G(n) = B(n+1) \text{ xor } B(n), \text{ in which } 0 \le n \le N-1 \text{ and } B(N) = 0$$

Ex: {0 0 1} → {0 1}, {0 1 0} → {1 1}

B(2) B(1) B(0) G(1) G(0) B(2) B(1) B(0) G(1) G(0)

- The formula is equivalent to

  $G(n) = B(n+1) + B(n)$, and ignore the carry (modulus)

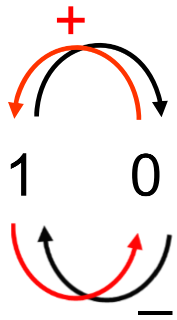
- Why is the equation true?
  - The problem arises from the carry problem (mapping for number n and n+1).

n   : x … x 0 1 1 1
n+1: x … x 1 0 0 0

When carrier occurs,
only will this bit differ.

n   : x … x x x 0
n+1: x … x x x 1

When no carry occurs,
only will this bit differ.



  - Also note that the mapping is one-to-one.

- From G(n) to B(n):

  $B(n) = B(n+1) - G(n)$, and use modulus operation

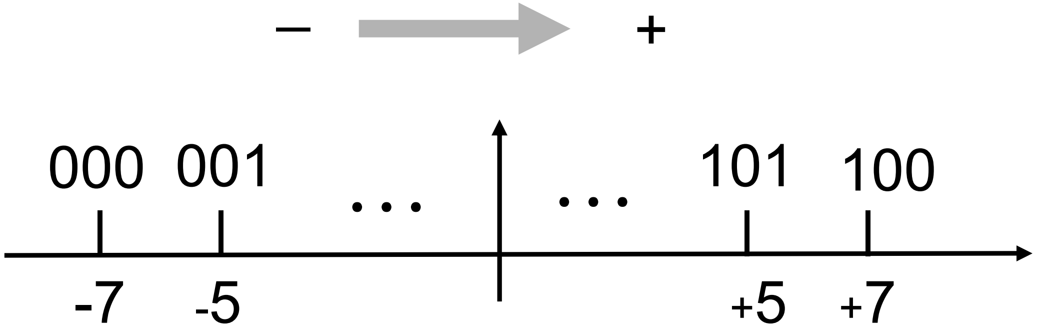
Ex: {0 0 1} → {0 1},   {0 1 0} → {1 1}
      {0 1} → {0 1}   {1 1} → {1 0}

B(1)=B(2)-G(1)=0-0=0
B(0)=B(1)-G(0)=0-1= -1=1

B(1)=B(2)-G(1)=0-1= -1=1
B(0)=B(1)-G(0)=1-1= 0

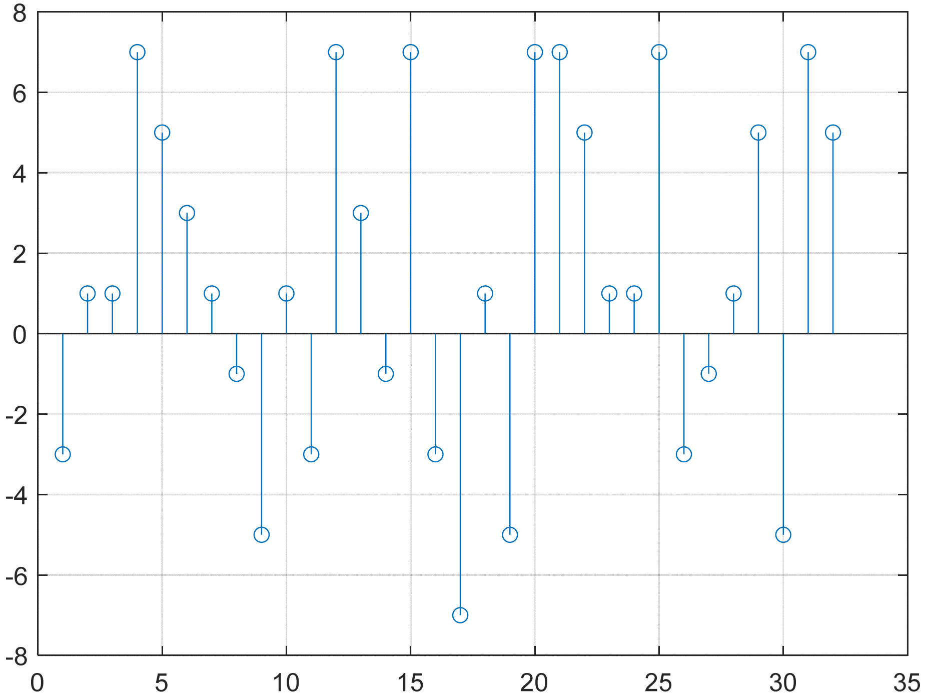
- How to conduct Gray mapping for N-PAM symbols?
  - Online calculation
  - By table lookup
- Online calculation:
  - Separate bits into segments (each corresponds to a digital symbol)
  - For each segment, conduct Gray-to-binary mapping
  - Convert binary bits into an integer (I) and calculate the symbol value ($2\times I\text{-}N+1$)
- By table lookup:
  - Using the Gray coded number as an index and construct a table (stores symbol values) offline
  - Each symbol value can then be online obtained with the table

- Practice 1:
  - Generate a 8-PAM Gray mapper.
  - Map a bit sequence (S5P1.mat) with the mapper.

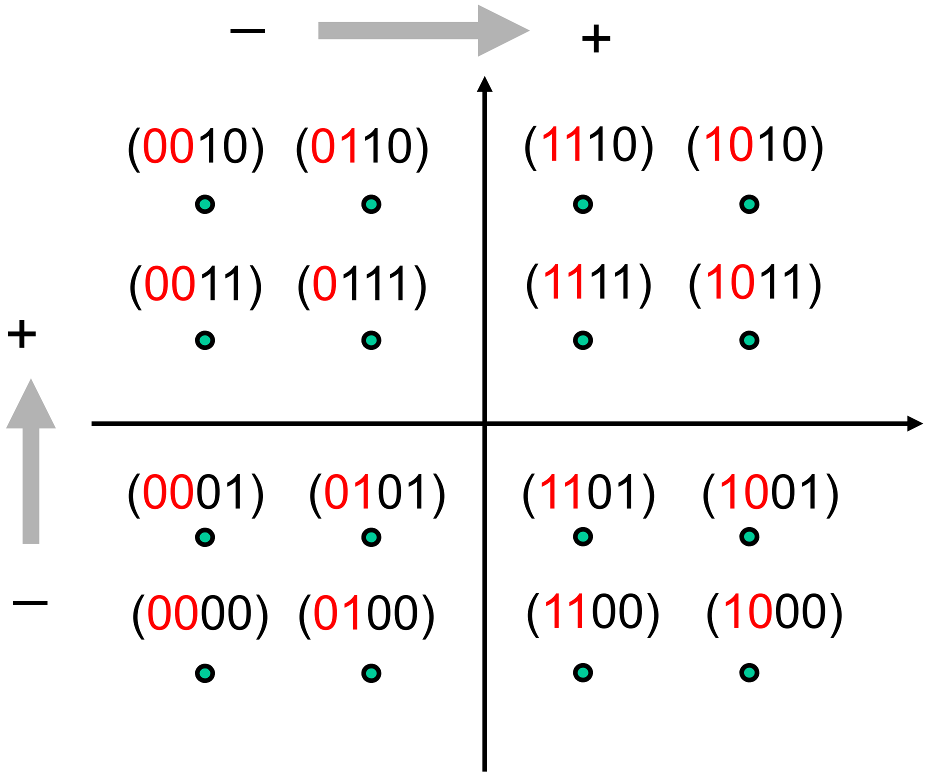


  - Generalize to have a N-PAM Gray mapper where N=2^n and n is an even number.

- Results:

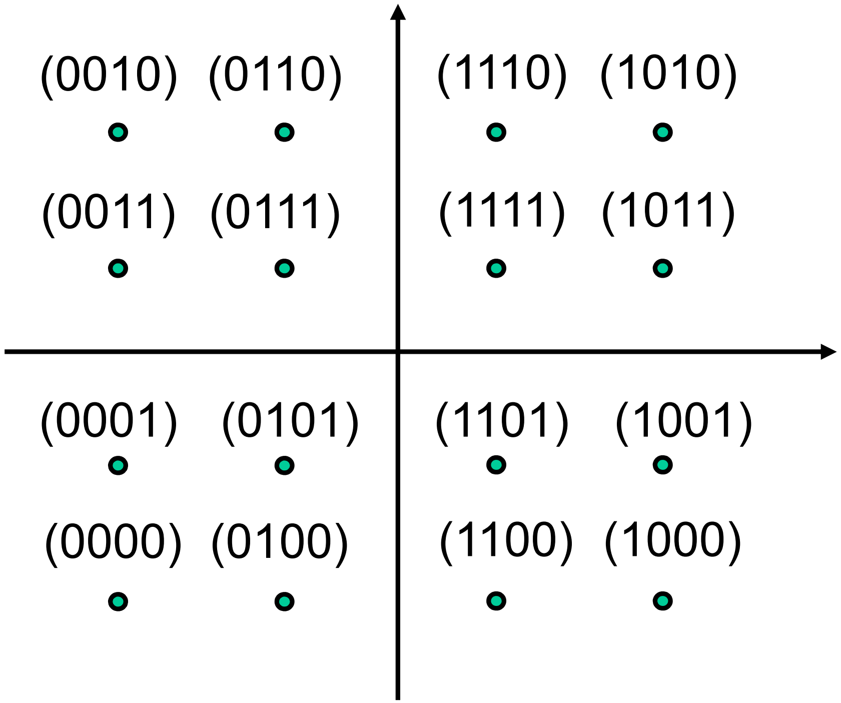


- QAM Gray mapping:
  - Apply one-dimensional Grapping in I and Q directions



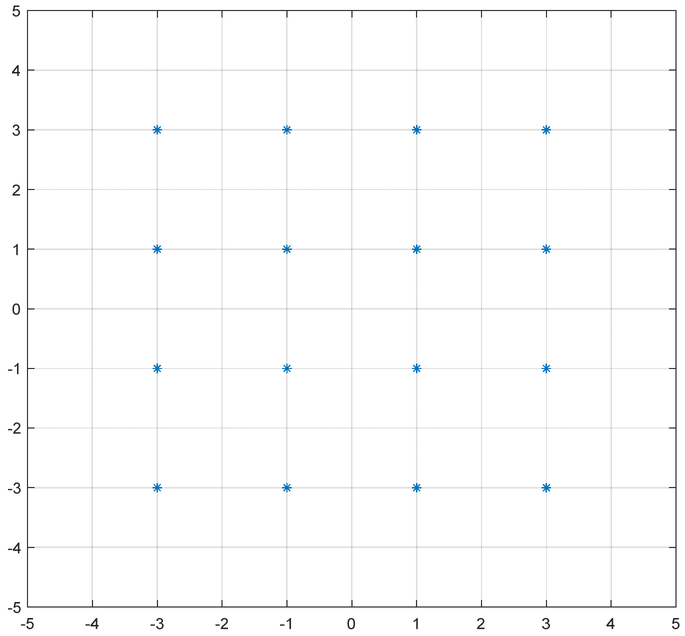
  - Note that there are more than one way to conduct the mapping.

- Practice 2:
  - Generate a 16-QAM Gray mapper
  - Map a bit sequence (S5P2.mat) with the mappers.

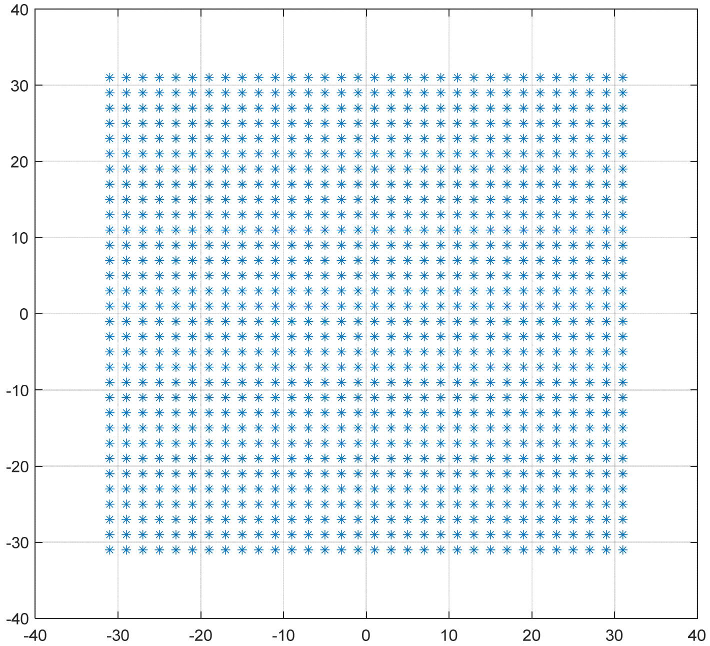


  - Generalize to have a N^2-QAM Gray mapper where $N=2^n$.

- Results:



16 QAM

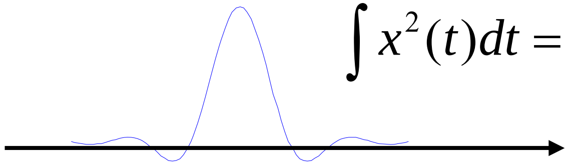


1024 QAM

- E_s/N_0 : symbol energy to noise spectral density ratio

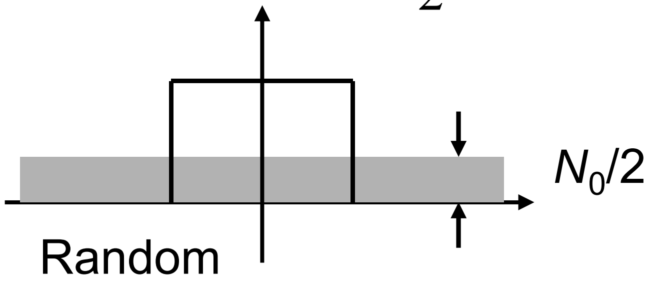
$$x(t) = \sqrt{E_s}\phi(t)$$

$$\int x^2(t)dt = E_s$$

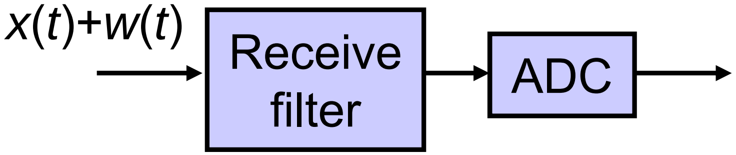


Deterministic

$$E\left\{w(t)w(s)\right\} = \frac{N_0}{2}\delta(t-s)$$

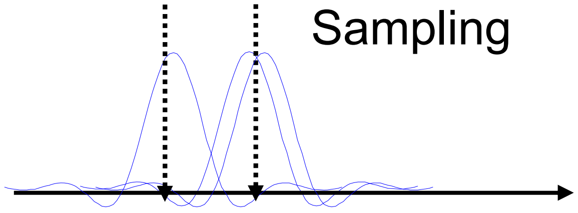


- At the ADC output:


$x(t)+w(t)$

Receive filter

ADC



$$\int\left[x(t)+w(t)\right]\phi(t)dt = \int\sqrt{E_s}\phi^2(t)dt + \int\phi(t)w(t)dt = \sqrt{E_s} + w(n)$$

Signal power: E_s

Noise power: $E\left\{w^2(n)\right\} = \frac{N_0}{2}$ * Evaluated in a sampling instant

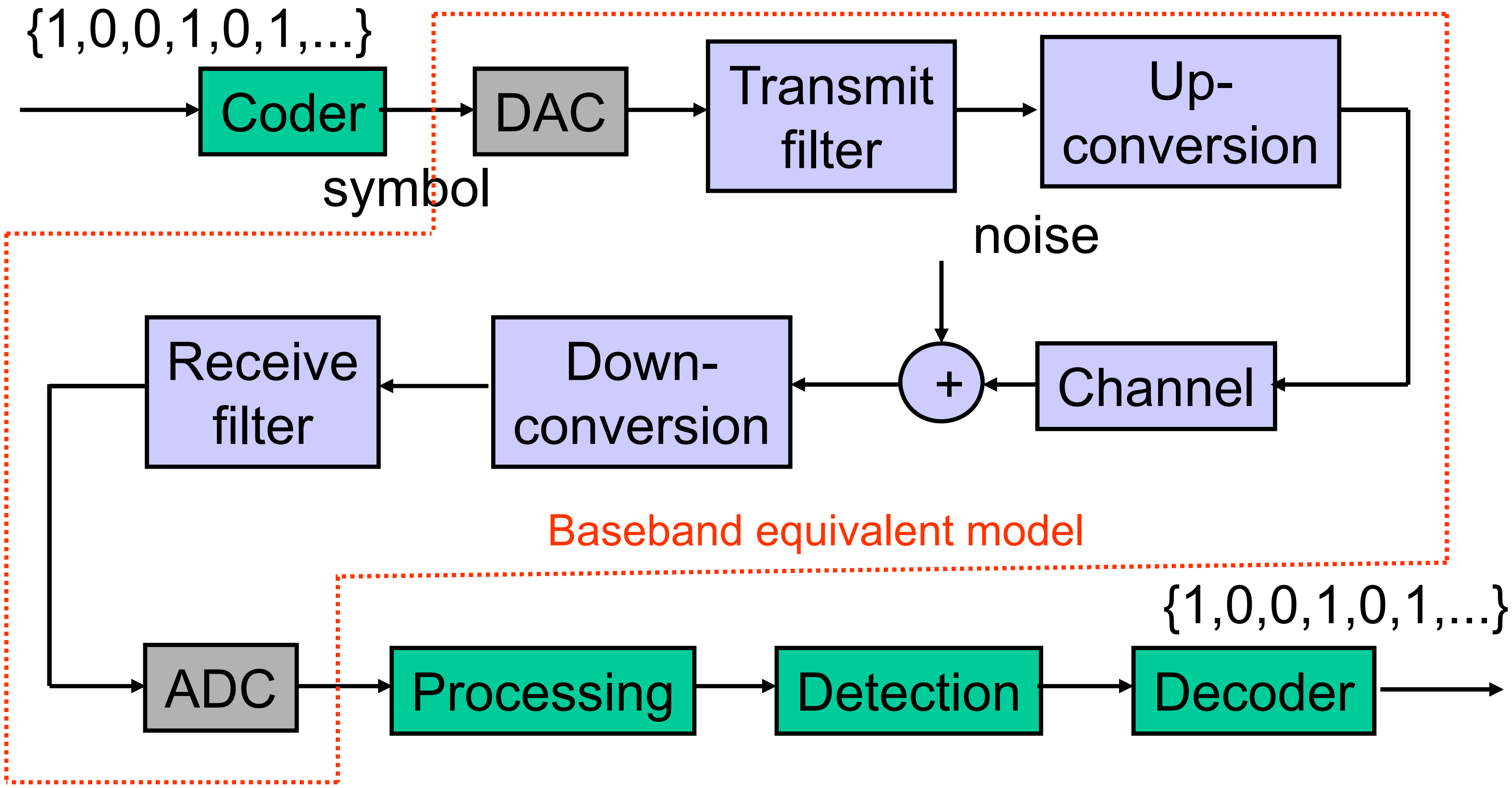
- Thus, the SNR (evaluated in the digital domain) is then $2E_s/N_0$. For complex signals, we then have $\mathrm{SNR}=E_s/N_0$.
- Now, if a symbol carry M bits information, then the energy consumed for one bit transmission is then $E_s=ME_b$. Then,

$$\mathrm{SNR}=\frac{E_s}{N_0}=\frac{ME_b}{N_0}\Rightarrow\frac{E_b}{N_0}=\frac{\mathrm{SNR}}{M}$$

* For BPSK modulation, $\mathrm{SNR}=E_b/N_0$

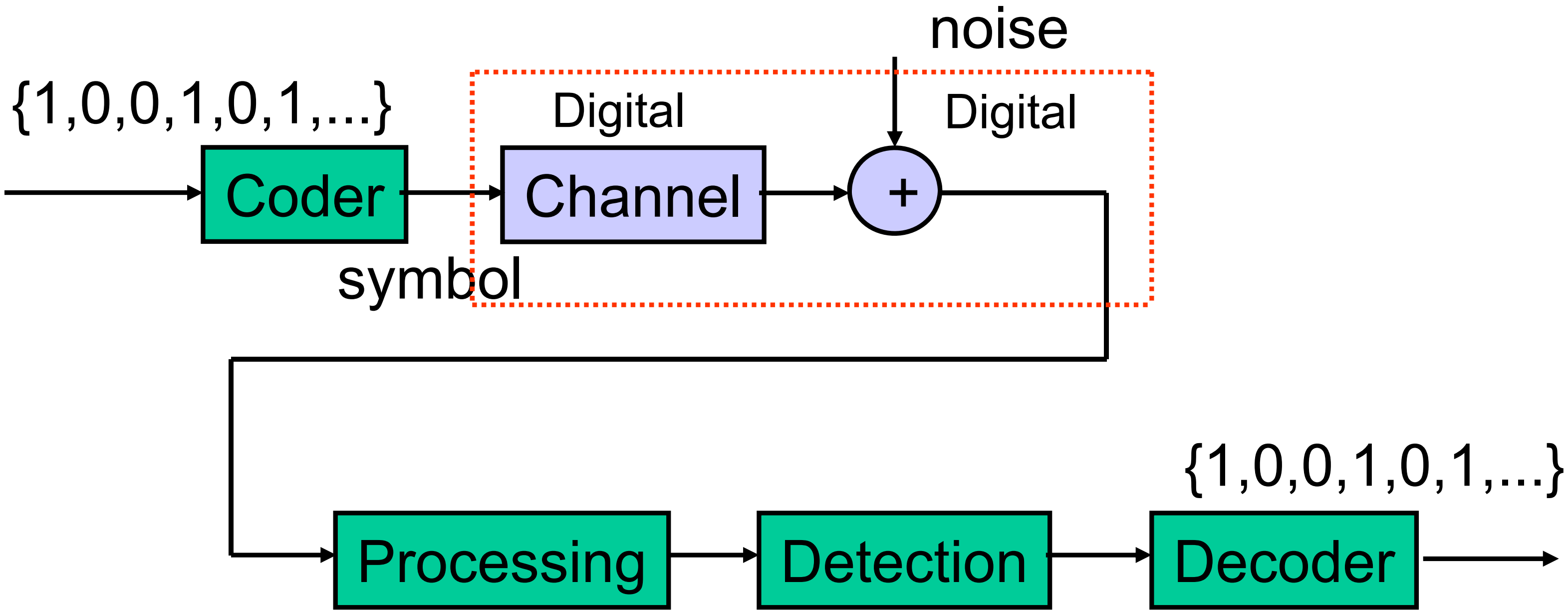
- E_b/N_0 is then a normalized measure indicating “SNR per bit”.

- We first consider the baseband equivalent system.



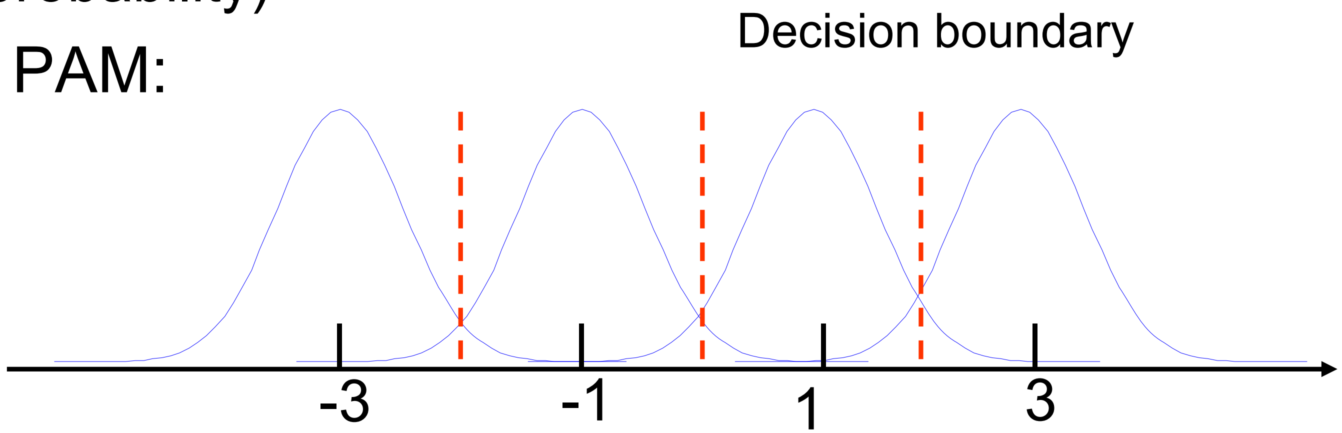
- Note that the operations of the digital systems can be exactly modeled. However, those of analog systems can only be approximately modeled.

- Thus, we conduct simulations all on the digital domain.

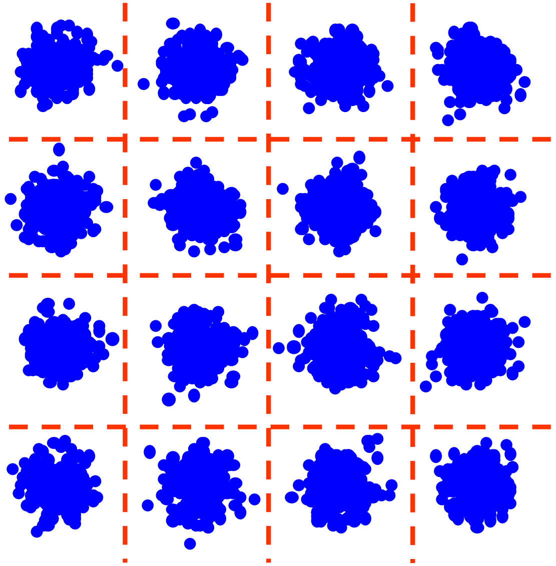


- This assumes that all the processing in analog domain is either perfect or can be absorbed into the channel and noise effects.

- Detection:
  - Minimum error probability (maximize the posteriori probability)
- For PAM:



- For QAM:



- Practice 3:
  - Given a 16-QAM symbol sequence (S5P3.mat), add Gaussian noise to have a SNR of 10 dB.
  - Check if your generation is right.
  - With the noisy sequence, conduct detection for the QAM symbols.
  - Calculate the symbol error rate.
  - Compare the simulated and theoretical results.

- Result:
  - SER=0.2151

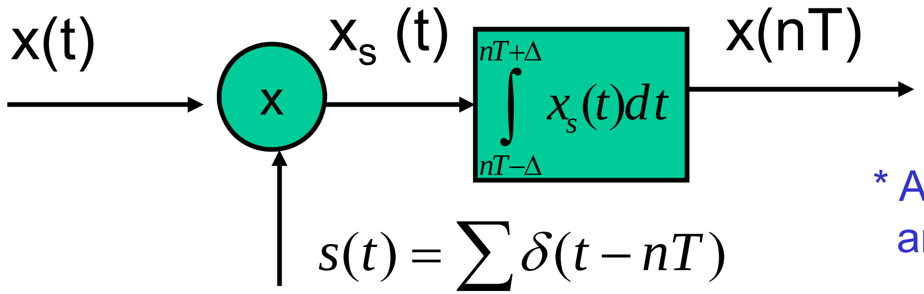
- Homework:
  - Simulate the symbol error rates (SERs) of the 16-QAM scheme with SNRs of 0dB, 2dB, 4dB … 18dB, etc such that you can plot a SER curve.
  - Calculate the theoretical SERs and also plot a curve.
  - Put these two curves in the same figure to see if your simulation results are OK.

Note: you may find that the result is different for different simulation.

- Reading assignment:
  - Sampling/reconstruction
  - Downsampling/upsampling

Lab. 6 Sampling and Rate Conversion

- Sampling:



$$s(t)=\sum_n \delta(t-nT)$$

* An impulse is an analog signal.

$$x_s(t)=x(t)\sum_n \delta(t-nT)=\sum_n x(nT)\delta(t-nT)$$

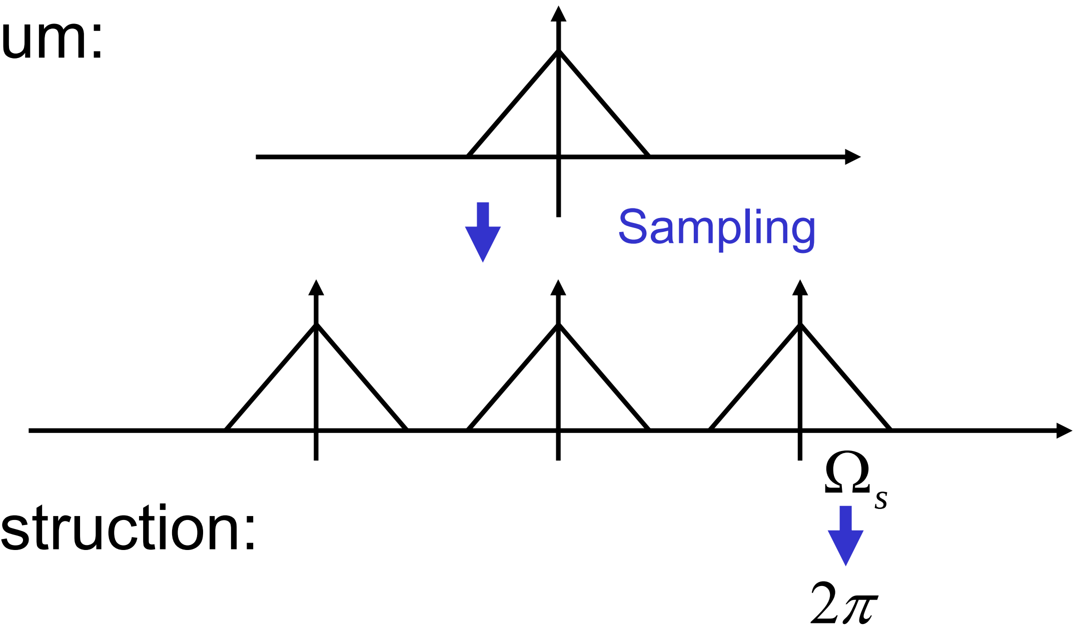
- The Fourier transform of an impulse train is still an impulse train.

$$S(j\Omega)=\frac{2\pi}{T}\sum_k \delta(\Omega-k\Omega_s)$$

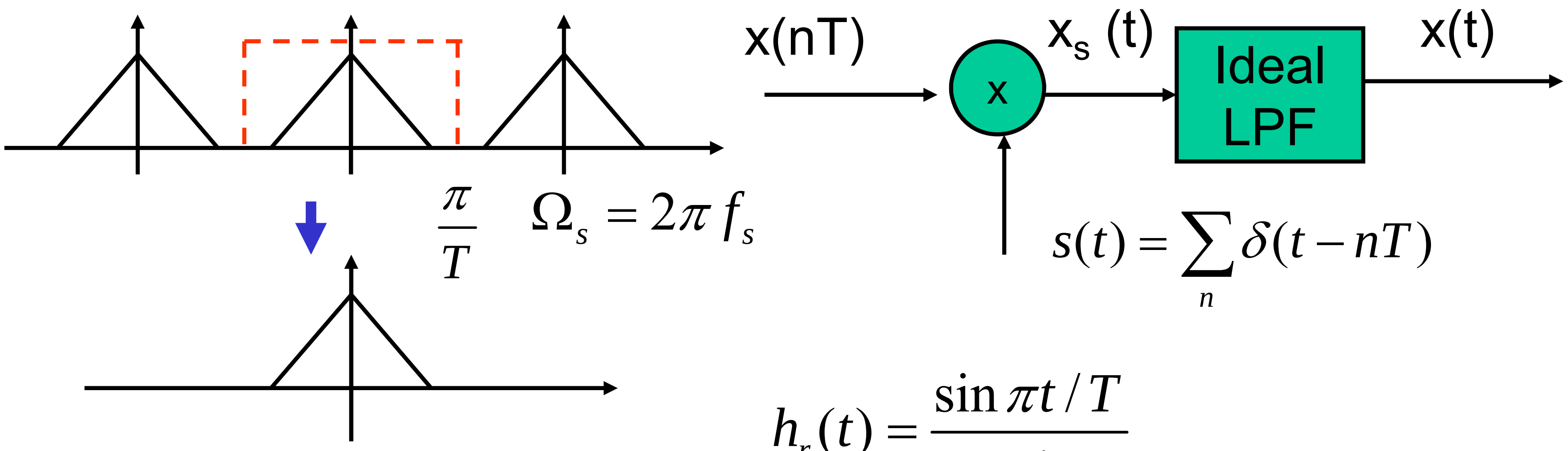
- Then,

$$X_s(j\Omega)=\frac{1}{2\pi}X(j\Omega)*S(j\Omega)\Rightarrow X_s(j\Omega)=\frac{1}{T}\sum_k X(j\Omega-kj\Omega_s)$$

- Spectrum:



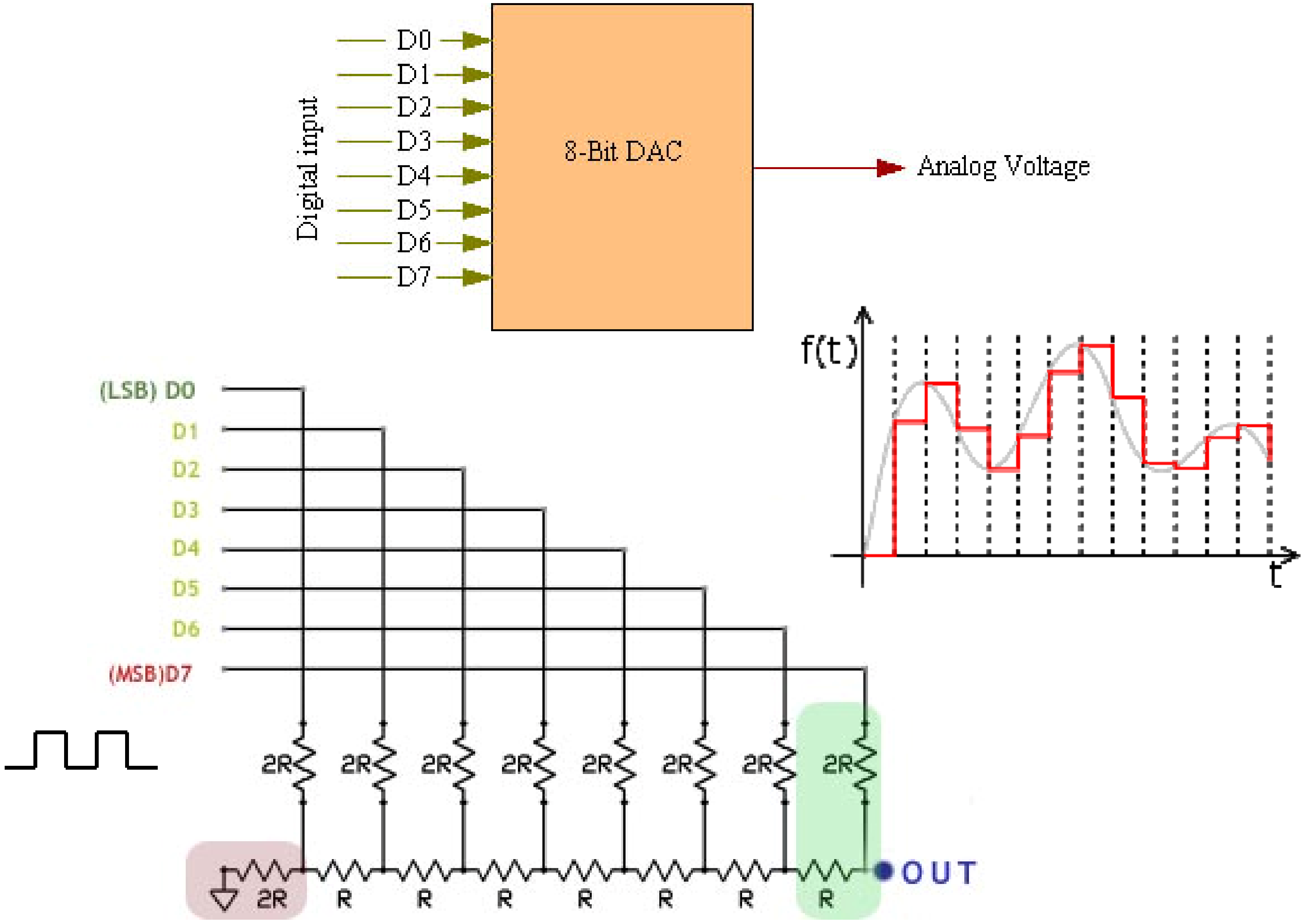
- Reconstruction:



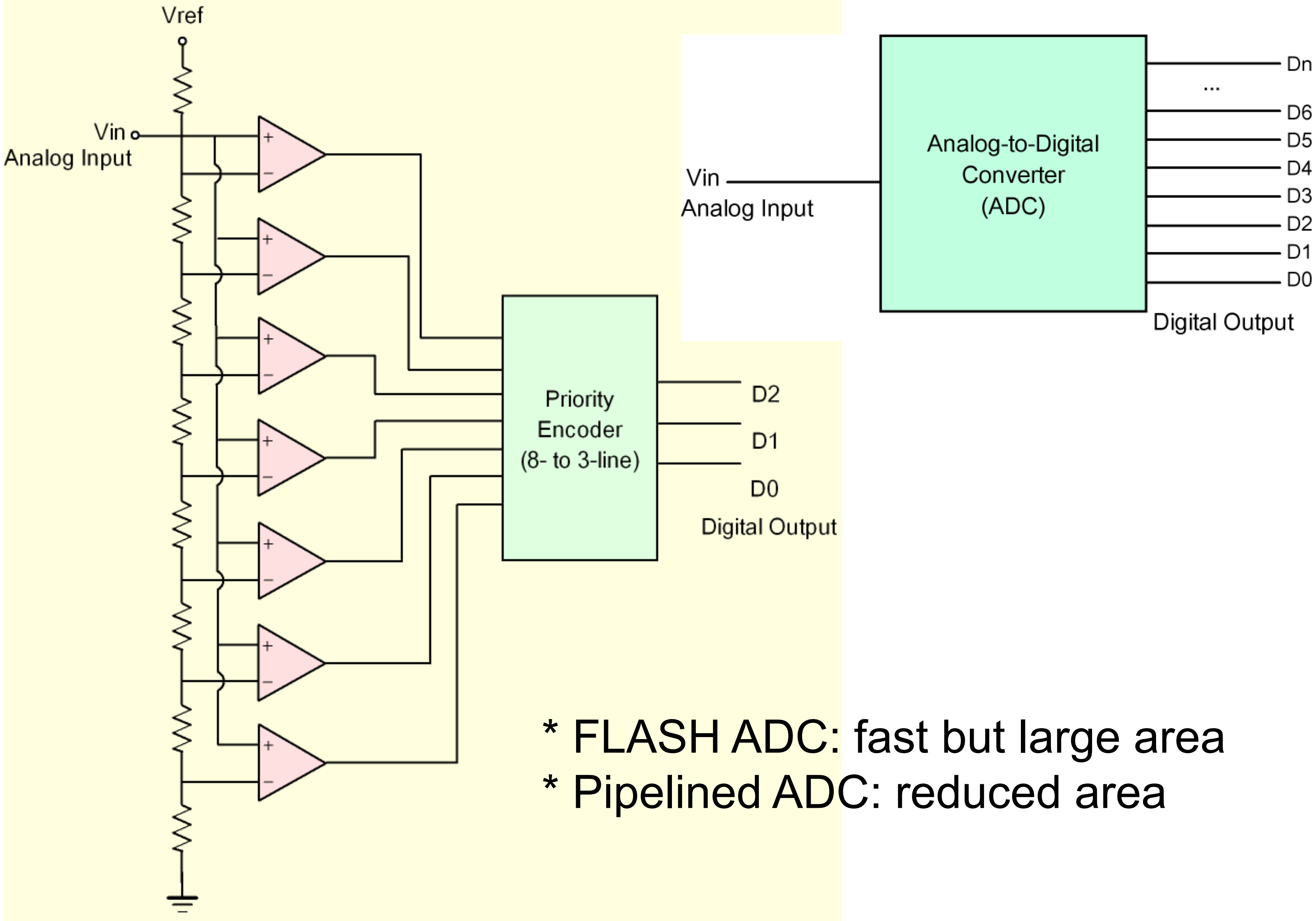
$$s(t) = \sum_n \delta(t - nT)$$

$$h_r(t) = \frac{\sin \pi t / T}{\pi t / T}$$

- Practical reconstruction device (DAC):



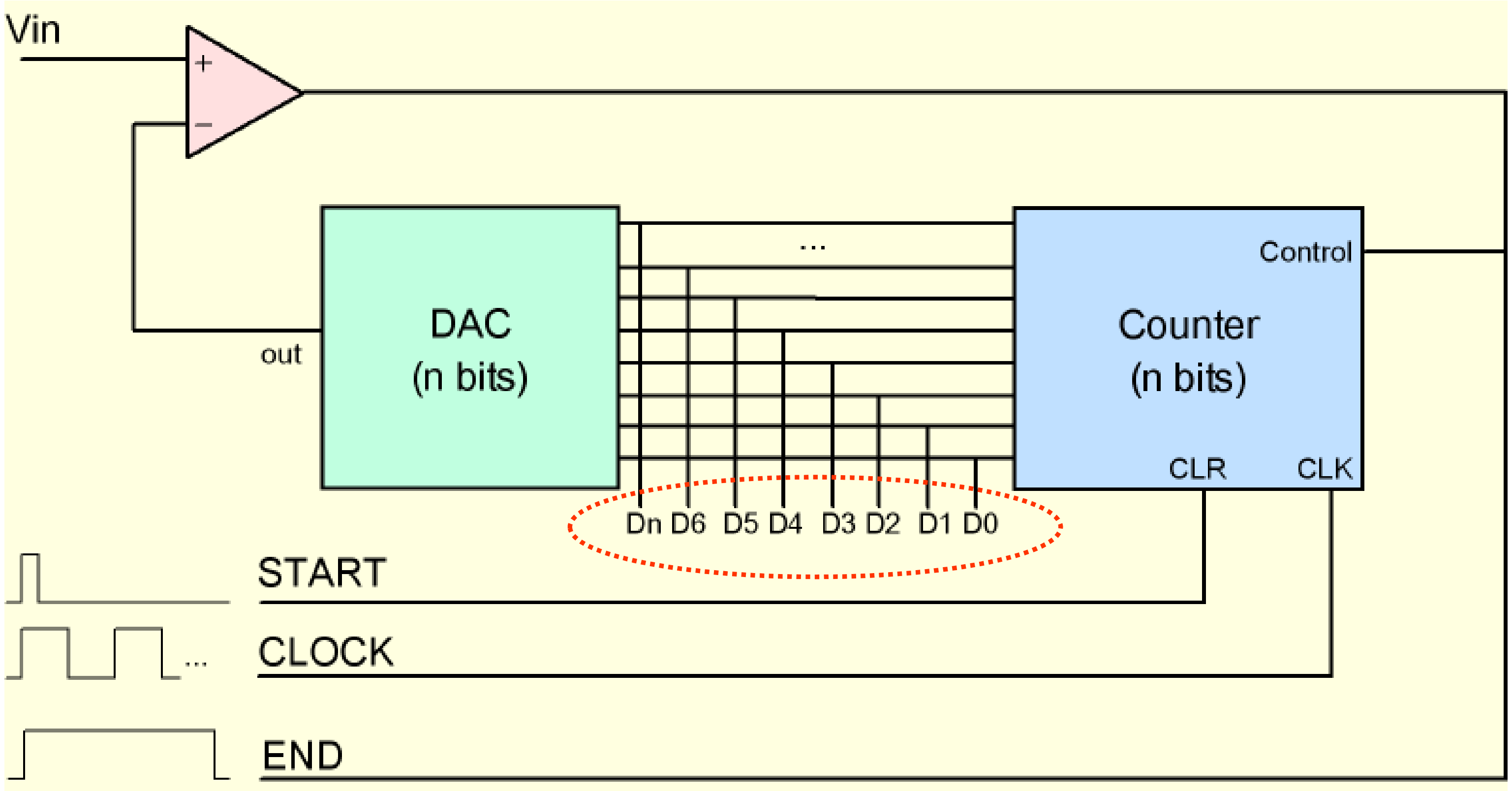
- Practical sampling device (ADC):



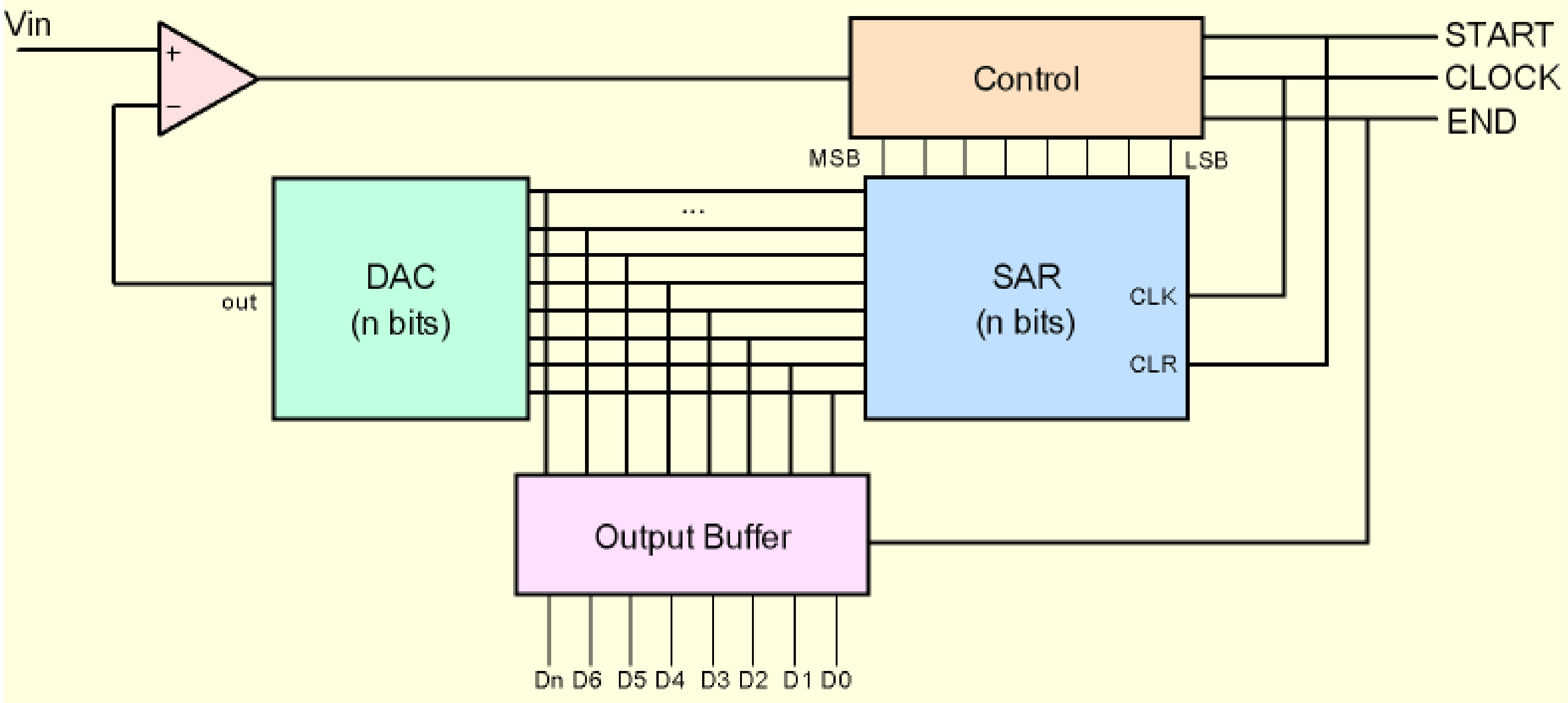
* FLASH ADC: fast but large area

* Pipelined ADC: reduced area

- Ramp counter ADC:



- Successive approximation ADC (SAR):

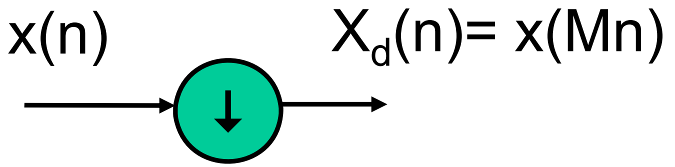


* Tree search

- Downsampling: $* X_s(j\Omega) = \frac{1}{T}\sum_k X(j\Omega - kj\Omega_s)$

$$X(e^{j\omega}) = \frac{1}{T}\sum_k X\left(j\frac{\omega}{T} - j\frac{2\pi k}{T}\right) \Rightarrow$$

$$X_d(e^{j\omega}) = \frac{1}{MT}\sum_m X\left(j\frac{\omega}{MT} - j\frac{2\pi m}{MT}\right)$$



x(n)

$X_d(n)= x(Mn)$

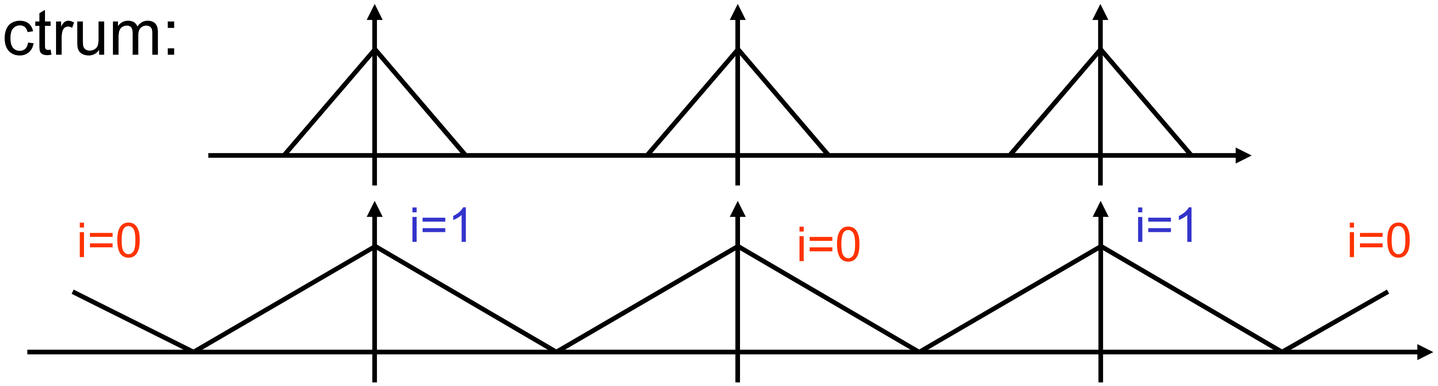
$$* \; \omega = \Omega T = \frac{\Omega}{f_s}$$

- Let m=i+kM and we have

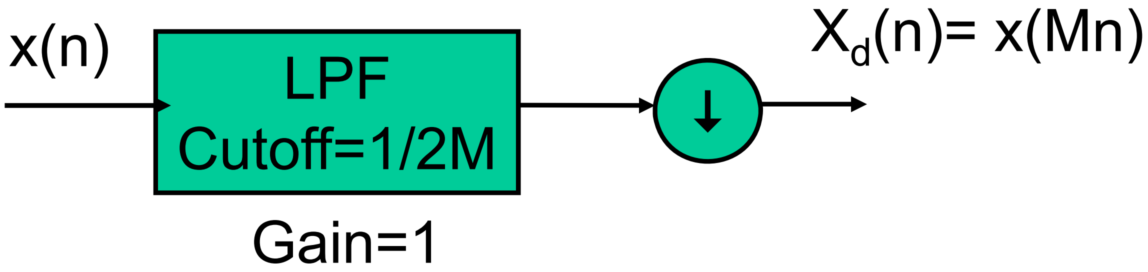
$$X_d(e^{j\omega}) = \frac{1}{M}\sum_{i=0}^{M-1}\left[\frac{1}{T}\sum_k \underbrace{X\left(j\frac{\omega}{MT} - j\frac{2\pi k}{T} - j\frac{2\pi i}{MT}\right)}_{X(e^{j(\omega-2\pi i)/M})}\right] \Rightarrow$$

$$X_d(e^{j\omega}) = \frac{1}{M}\sum_{i=0}^{M-1} X\left(e^{j(\omega/M - 2\pi i/M)}\right)$$

- Spectrum:

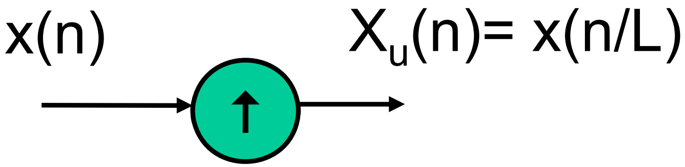


- To avoid aliasing, a filter is generally applied before the downsampling operation.



- Upsampling:

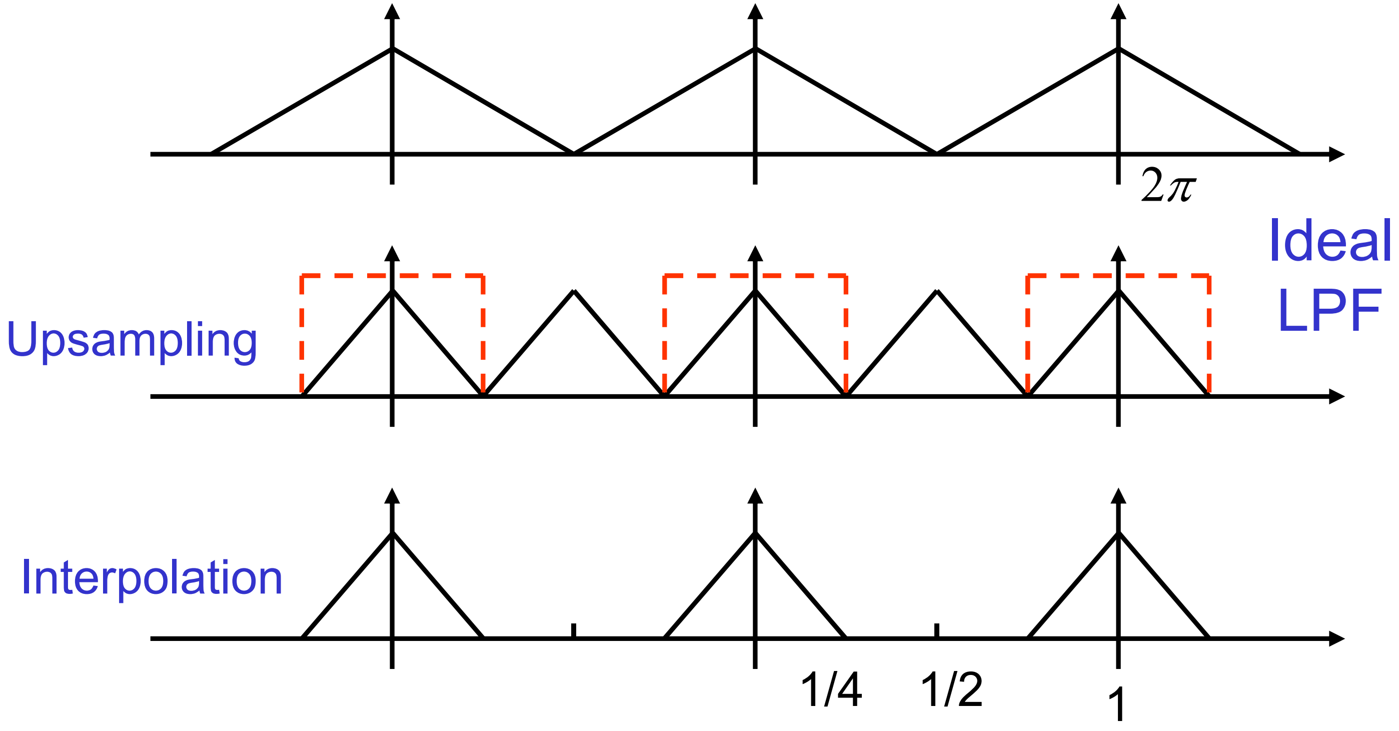
$$x_u(n) = \begin{cases} x(n/L), & n = 0, \pm L, \pm 2L, \cdots \\ 0, & \text{otherwise} \end{cases}$$



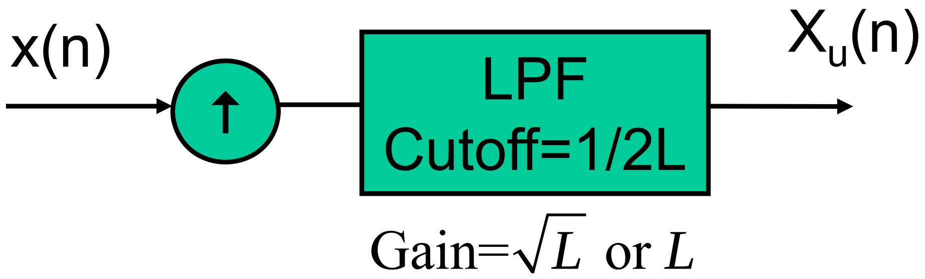
- The spectrum:

$$x_u(n) = \sum_k x(k)\delta(n-kL)$$

$$X_u(e^{j\omega}) = \sum_n \left(\sum_k x(k)\delta(n-kL) \right) e^{-j\omega n} = \sum_k x(k) e^{-j\omega kL} = X(e^{j\omega L})$$



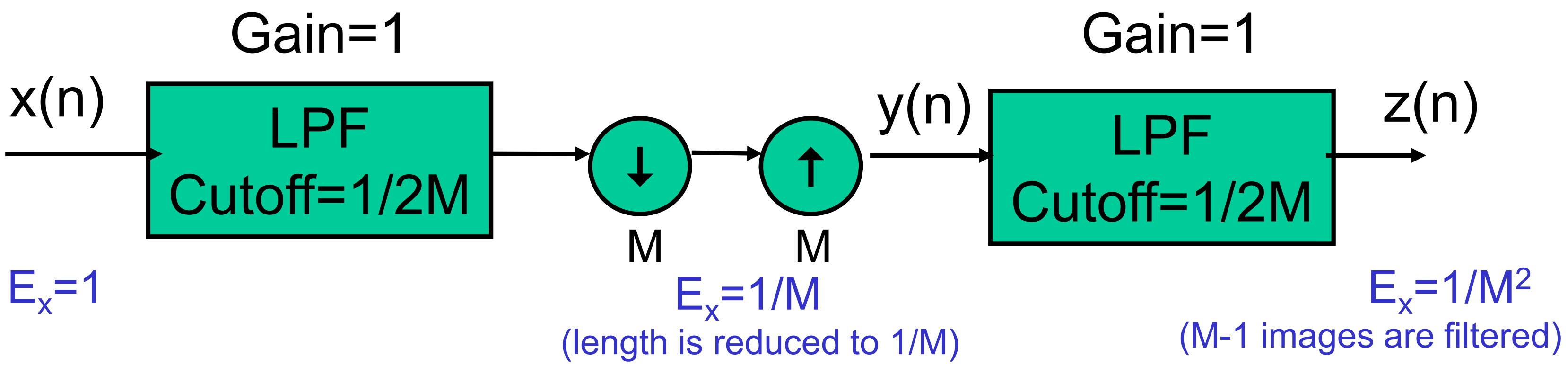
- The upsampling process is then equivalent to increase the sampling rate by a factor of L.



- The filtering operation is also known as interpolation.

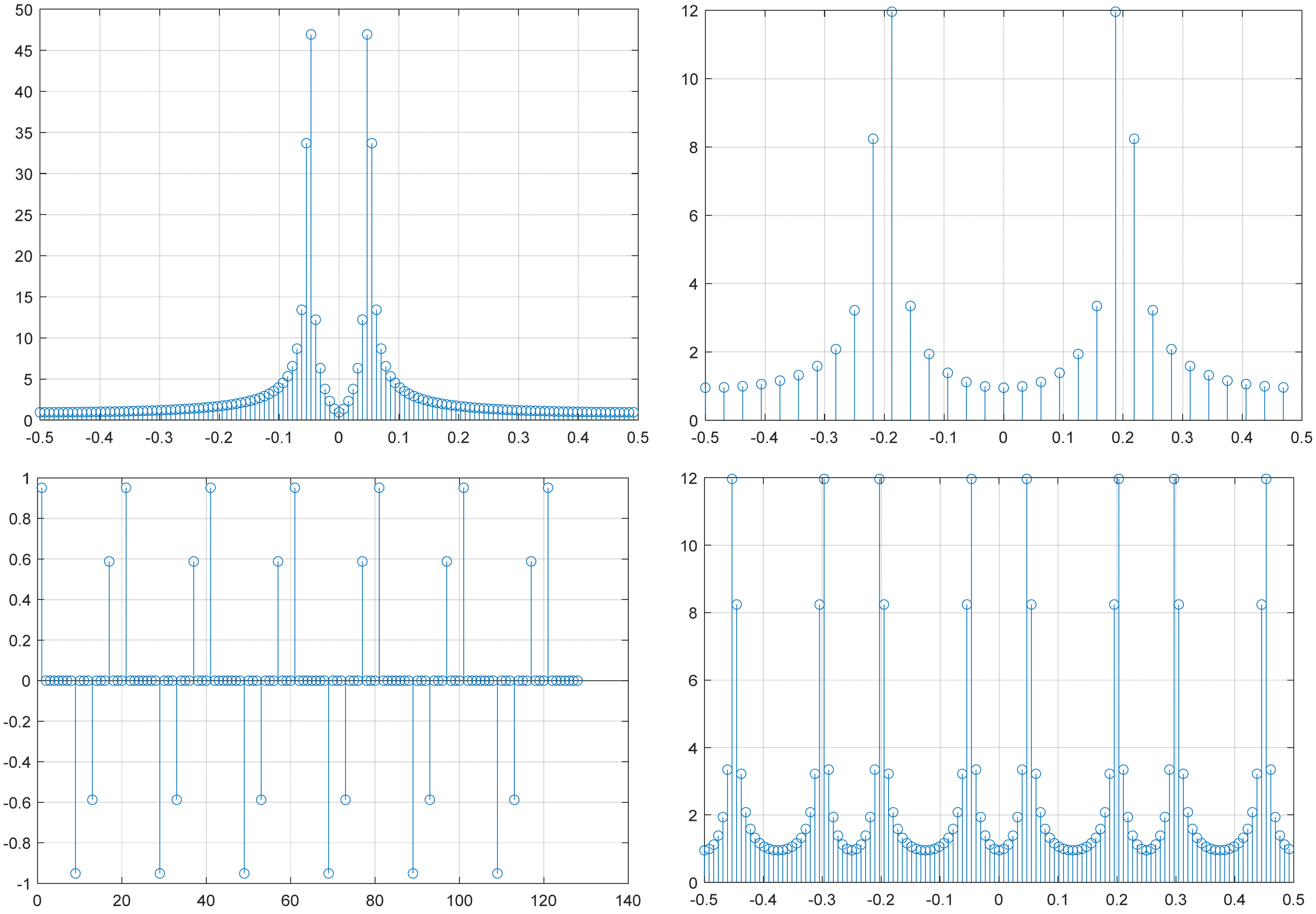
- Practice 1:
  - Generate a sinusoidal signal with frequency 1/20 (length 128) and observe its spectrum.
  - Downsample the signal with a factor of 4, and observe its spectrum.
  - Then, upsample the downsampled signal, and observe its time-domain signal and spectrum.

* Assume that no signal is filtered

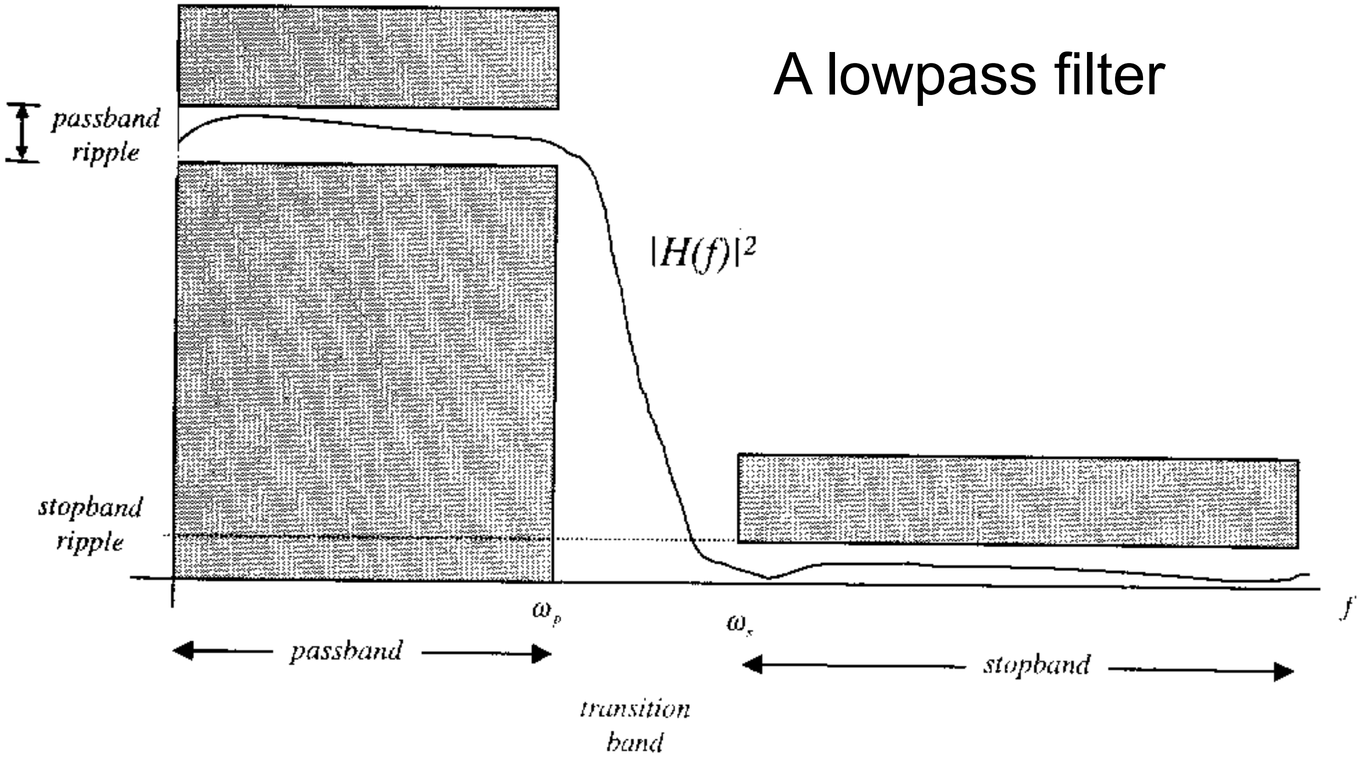


As a result, $\hat{x}(n) = M \times z(n)$

- Results:

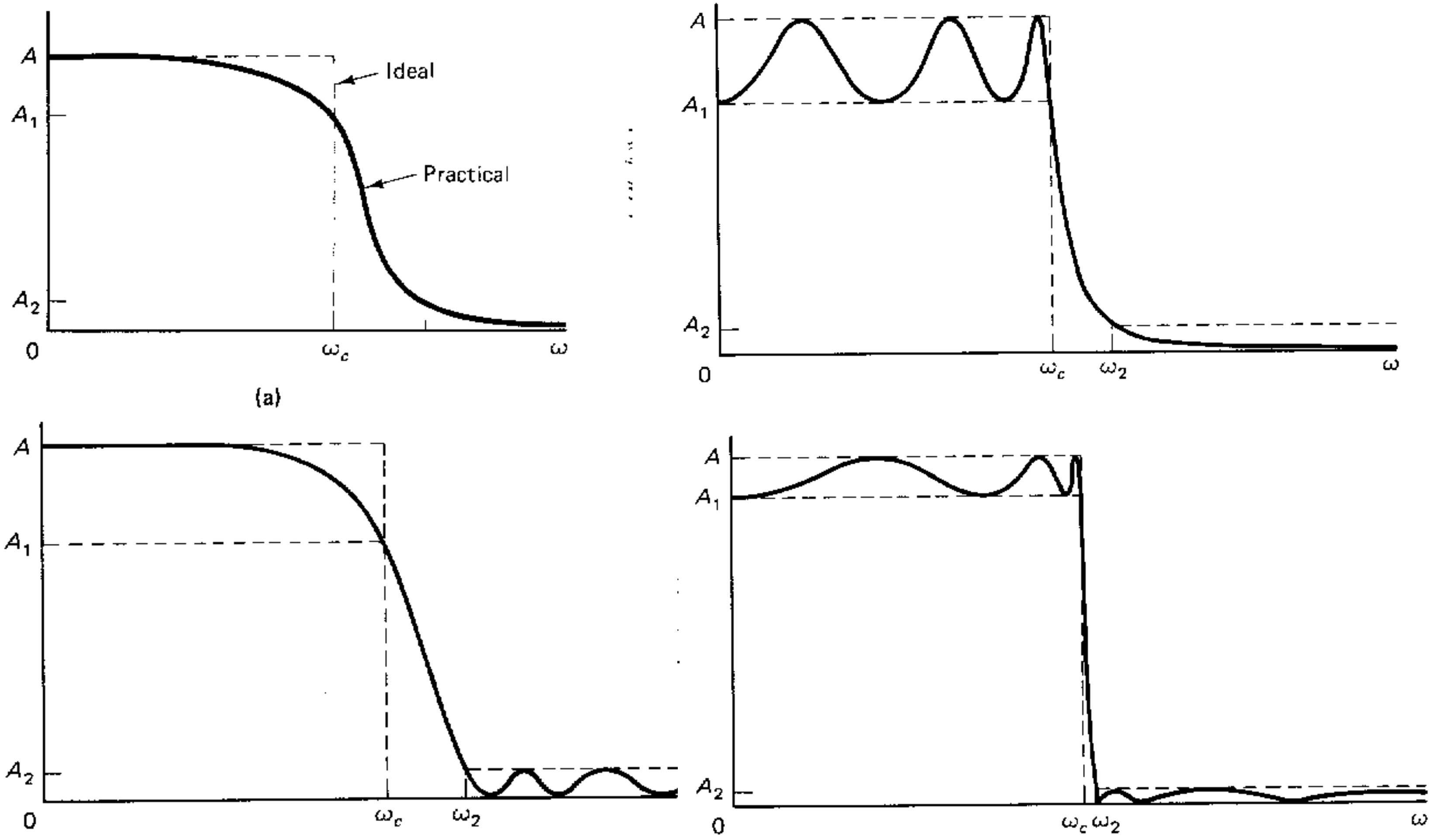

50
45
40
35
30
25
20
15
10
5
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
12
10
8
6
4
2
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120
140
12
10
8
6
4
2
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5

- General filter design:
  - Pass band
  - Stop band
  - Transition band
  - Passband ripple/stopband ripple

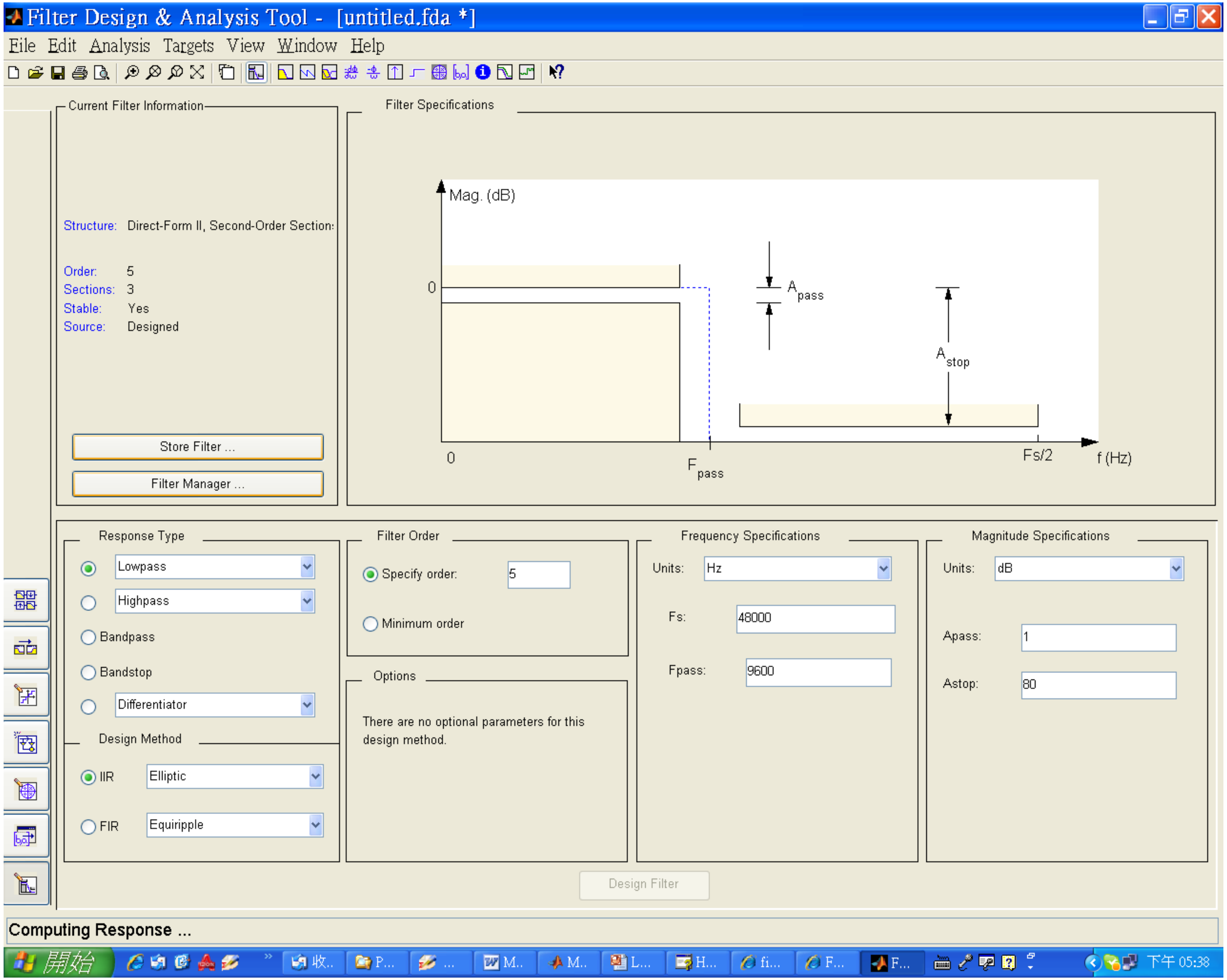


A lowpass filter

- The analog filter design (IIR):
  - 1. Butterworth, 2. Chebychev I, 3. Chebychev II, 4. Ellipic



- filterDesinger in Matlab:



Filter Design & Analysis Tool - [untitled.fda *]
File Edit Analysis Targets View Window Help
Mag. (dB)
0
A pass
A stop
0
F pass
Fs/2
f (Hz)
Store Filter ...
Filter Manager ...
Design Filter

Current Filter Information

Structure: Direct-Form II, Second-Order Section:

Order: 5
Sections: 3
Stable: Yes
Source: Designed

Filter Specifications

Response Type

● Lowpass
○ Highpass
○ Bandpass
○ Bandstop
○ Differentiator

Design Method

● IIR Elliptic
○ FIR Equiripple

Filter Order

● Specify order: 5
○ Minimum order

Options

There are no optional parameters for this design method.

Frequency Specifications

Units: Hz
Fs: 48000
Fpass: 9600

Magnitude Specifications

Units: dB
Apass: 1
Astop: 80

Computing Response ...

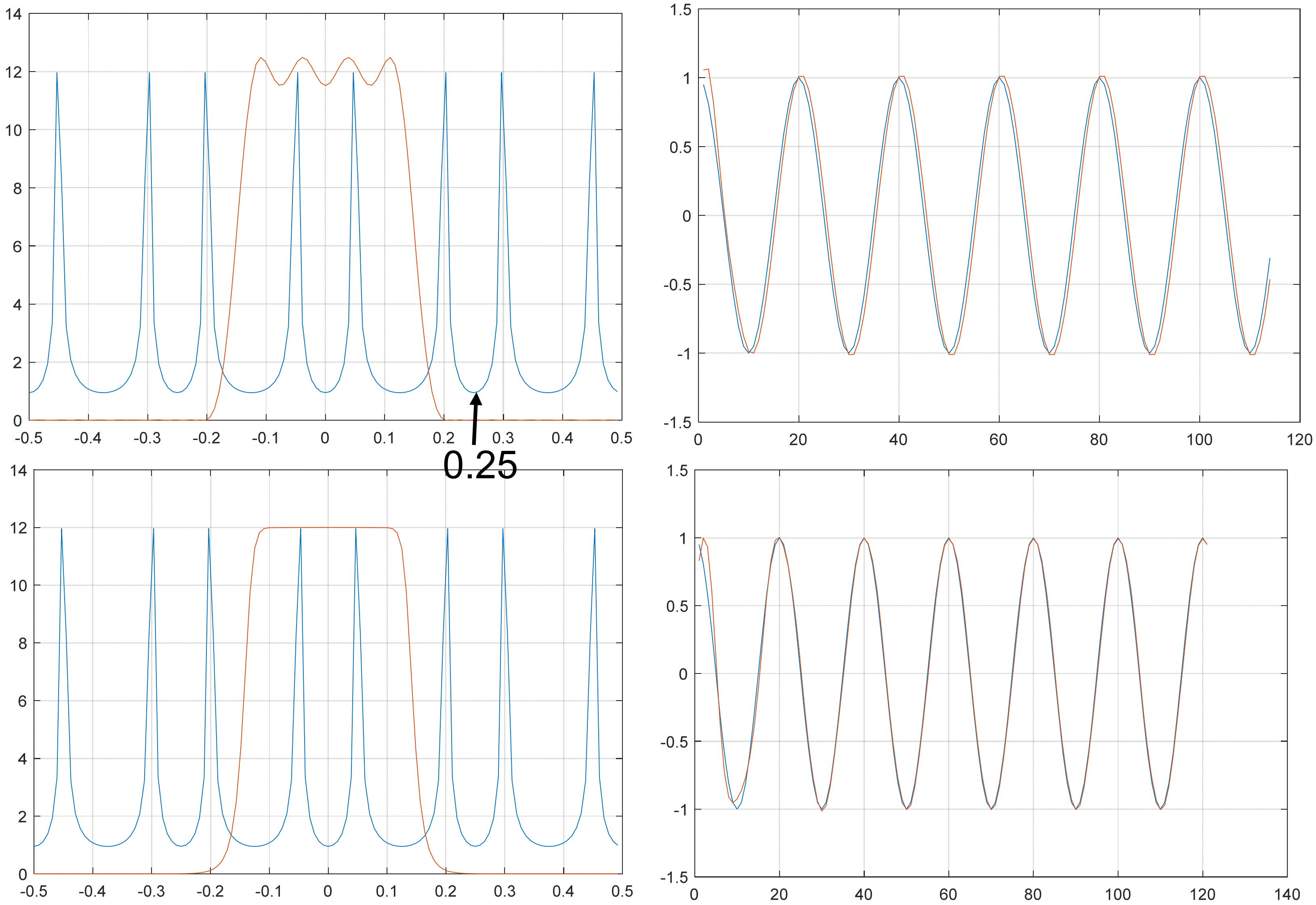
- Practice 2:
  - With the downsampled and then upsampled sinusoidal signal in Practice 1, design an FIR LPF to recover the original signal.
  - Plot the spectrum of upsampled signal and the frequency response of the filter.
  - Design a Butterworth IIR LPF to do the job.
  - Plot the spectrum of upsampled signal and the frequency response of the filter.
  - Calculate the MSE of these two interpolation schemes.

$$s(n)+v(n)\rightarrow y(n)=\overline{s}(n)+\overline{v}(n)$$

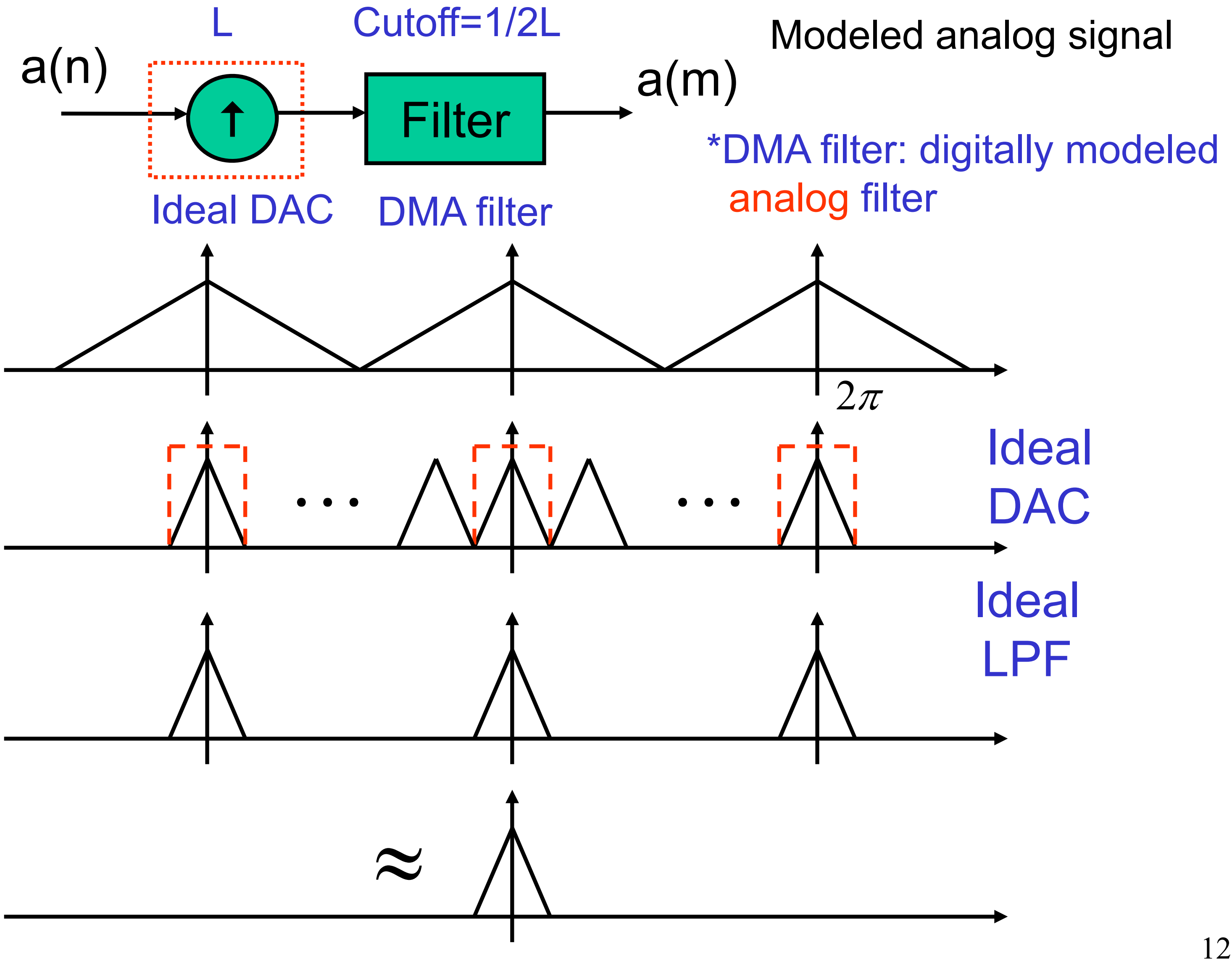
$$e(n)=\overline{s}(n)-s(n)+\overline{v}(n)=y(n)-s(n)$$

$$\mathrm{MSE(dB)}=10\log 10E\left\{\left|e(n)\right|^2\right\}$$

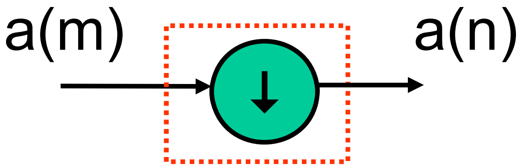
- Results:


14
12
10
8
6
4
2
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
0.25
1.5
1
0.5
0
-0.5
-1
-1.5
0
20
40
60
80
100
120
140

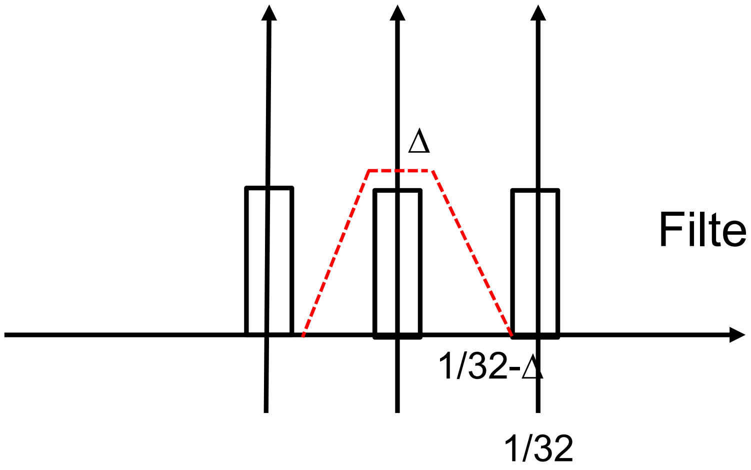
- How to simulate the analog signal obtained with DAC?
  - Discrete signal with high sampling rate.



- The filter, which is digital, used to model the analog filter is called digitally modeled analog (DMA) filter.
- Thus, an ideal DAC can be modeled as an device to increase the sampling rate.
- Note that a DMA filter, modeling an analog filter, always has an IIR responses.
- In our simulations, we then have two types of sampling rate conversion
  - Digital signal to digital signal (for digital processing)
  - Digital signal to analog signal (for DAC)
- An ideal ADC can be modeled by a device downsampling the modeled analog signal.

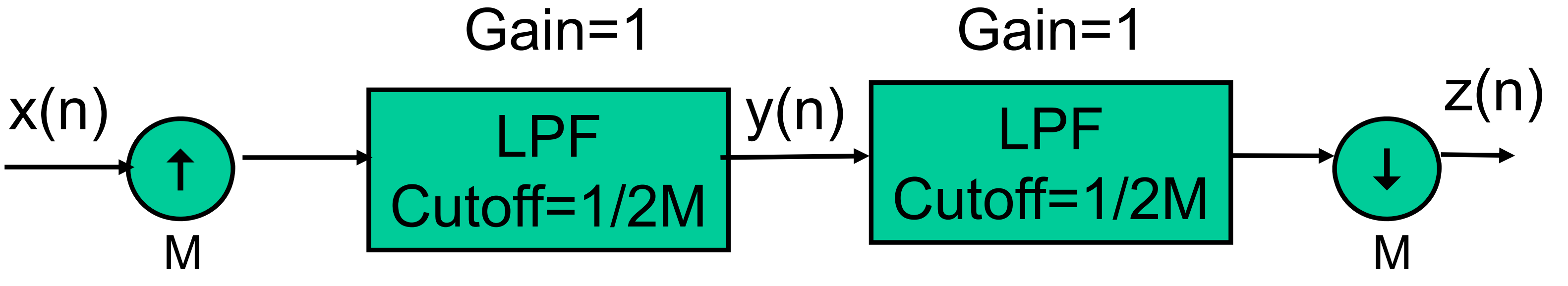


- Practice 3:
  - Given a digital signal (S6P3.mat), design a DMA filter, and let the signal pass through an ideal DAC with the up-sampling rate of 32.
  - Plot the original and DMA filtered signals.
  - Let the modeled analog signal pass with an ADC with the downsampling rate of 32.
  - Compare the ADC-sampled signal with the original signal.



Filter specification

- Signal energy variation:



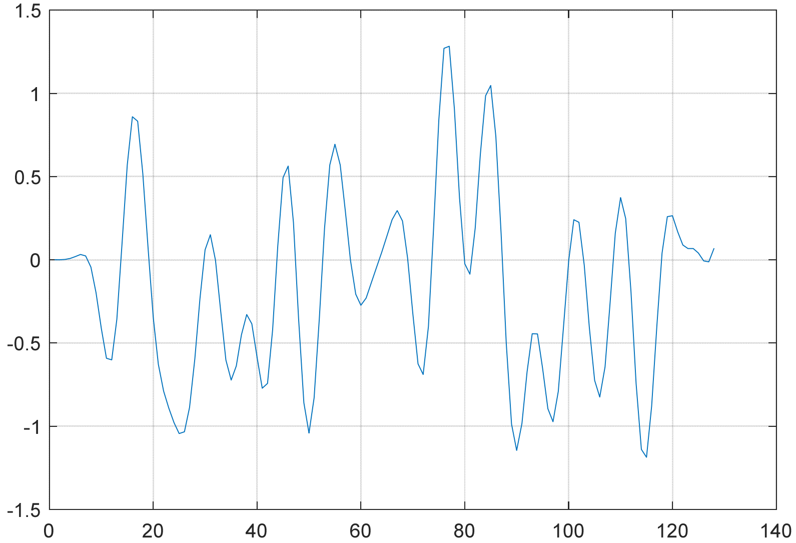
E_x=1

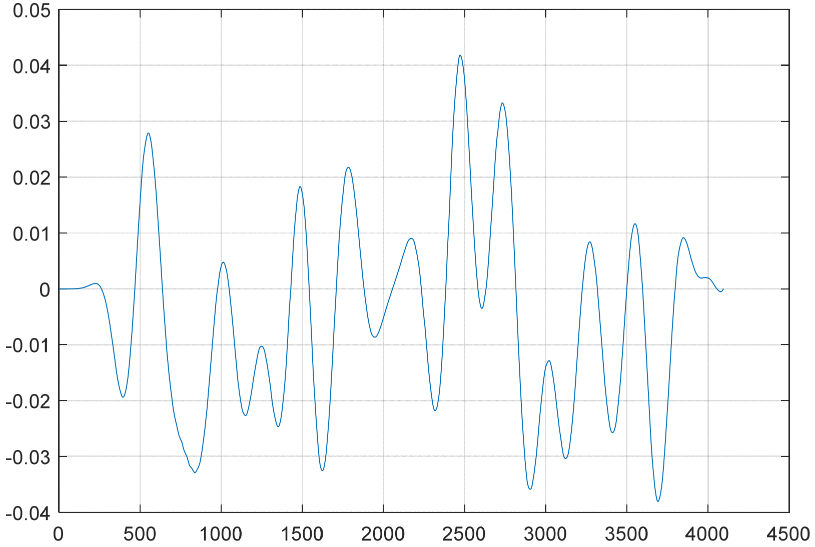
E_x=1/M
(M-1 images are filtered)

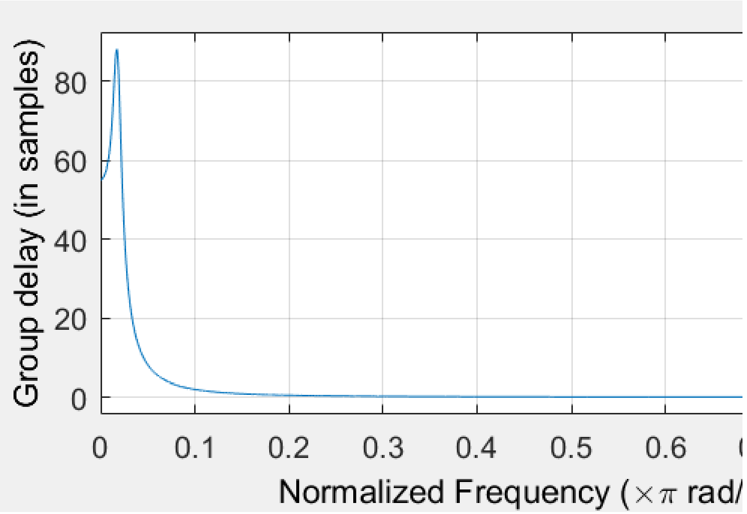
E_x=1/M^2
(length is reduced to 1/M)

As a result, $\hat{x}(n) = M \times z(n)$

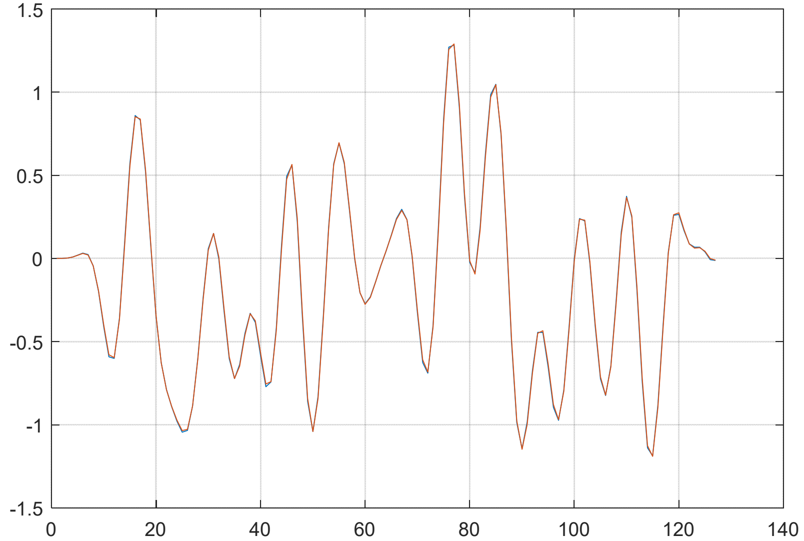
- Results:







Delay=64/32=2

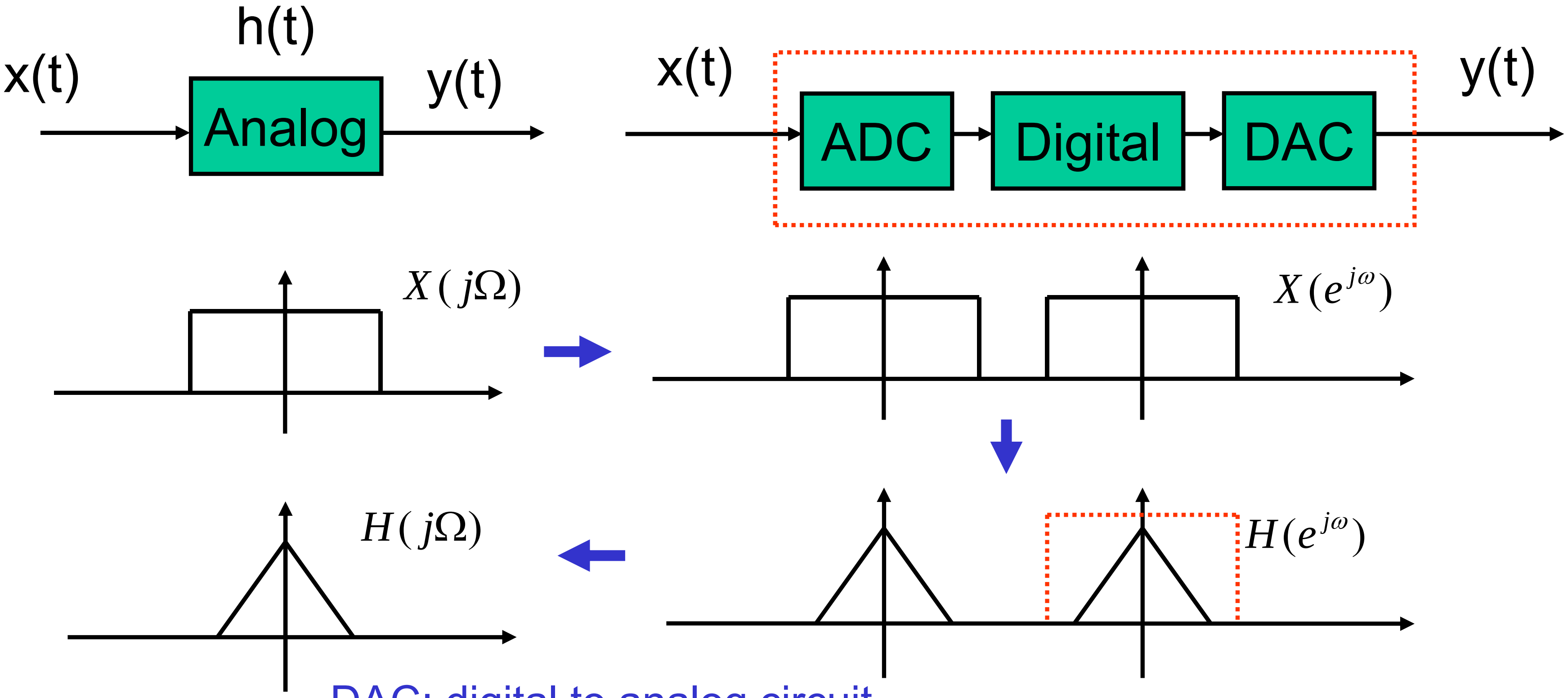


- Homework:
  - For a given signal (S6HW.mat), try to conduct down - samping with a largest factor without causing aliasing.
  - Design an IIR filter to conduct upsampling and interpolation recovering the original signal.
  - Design an FIR filter to do the same job.

- Reading assignment:
  - Pulse shaping
  - RC, SRRC

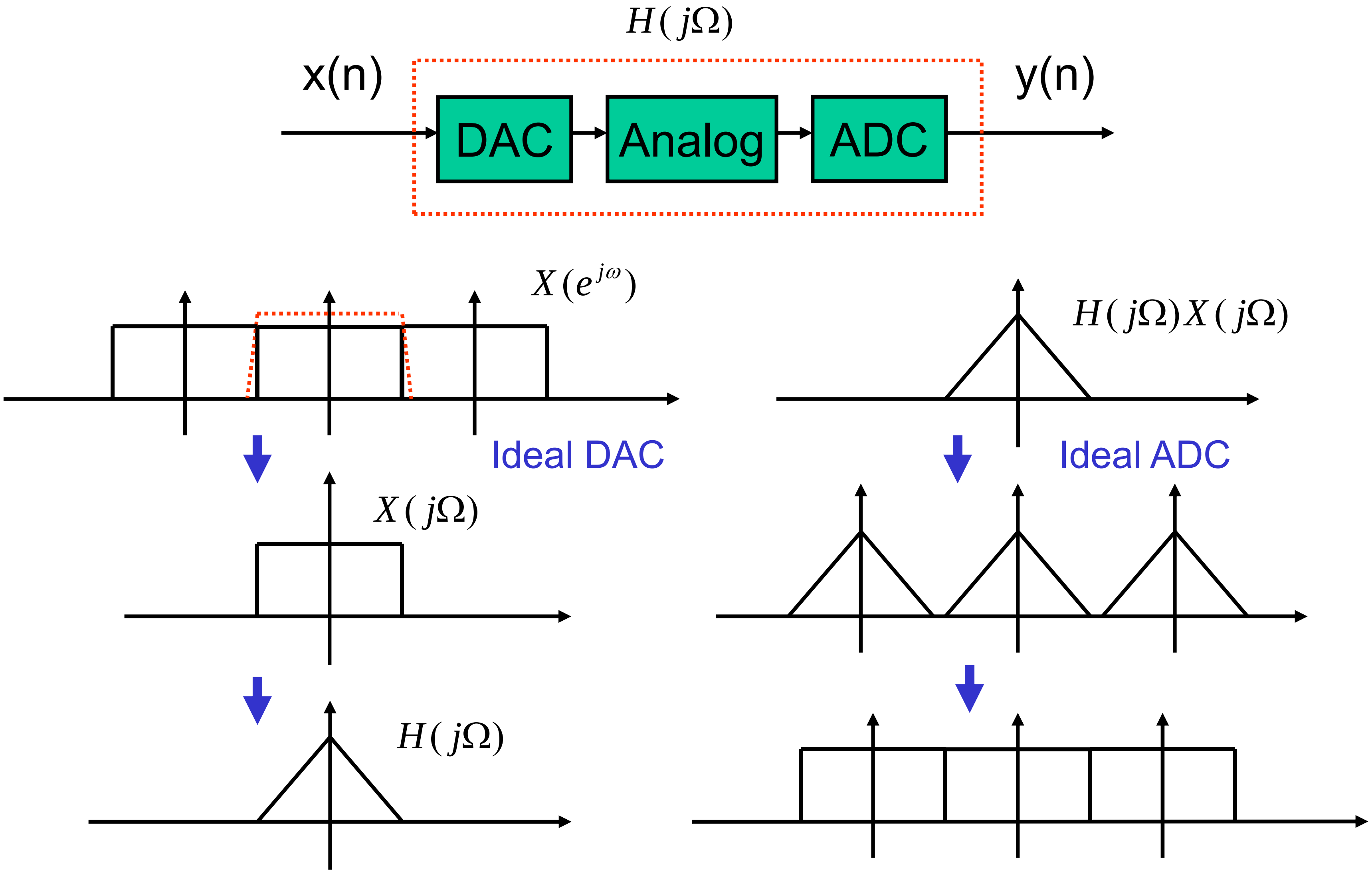
Lab. 7 Transmit Filtering/Up conversion I

- Digital processing of analog systems:

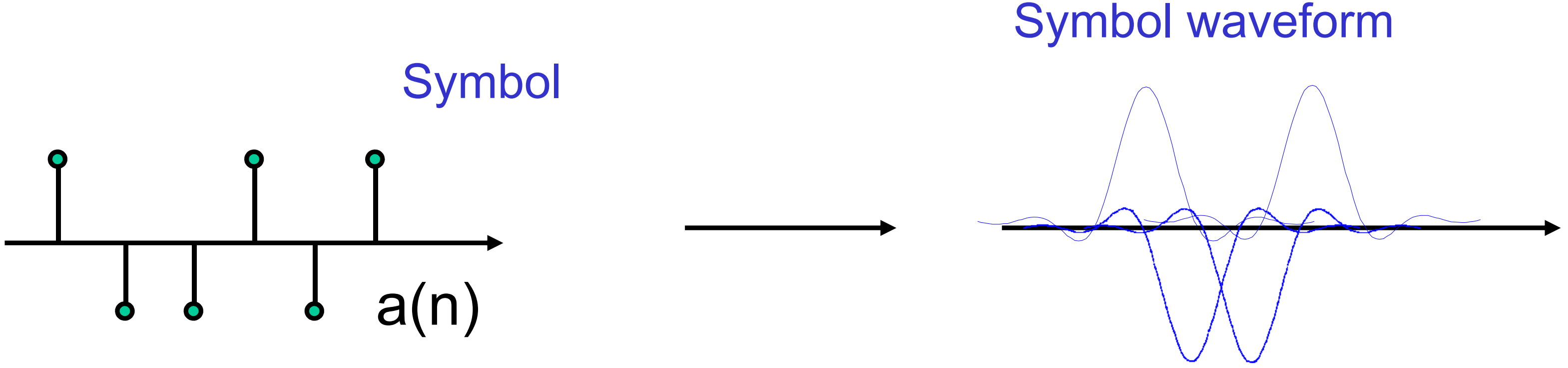


DAC: digital to analog circuit
ADC: analog to digital circuit

- Digital communication system:

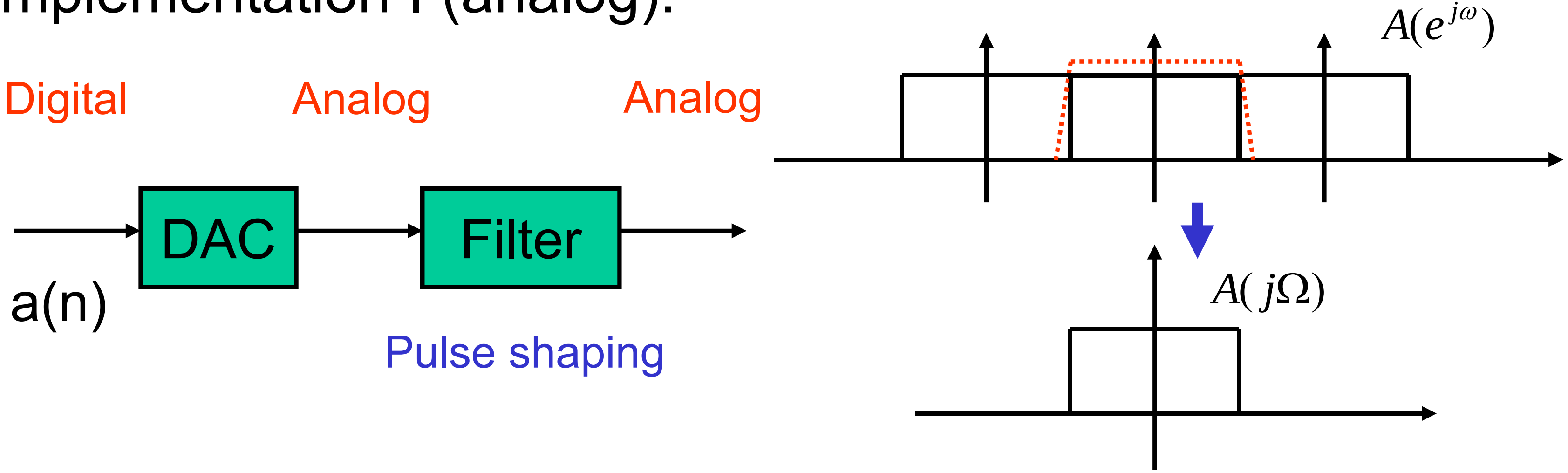


- Transmit filtering (pulse shaping):
  - Generate analog symbol

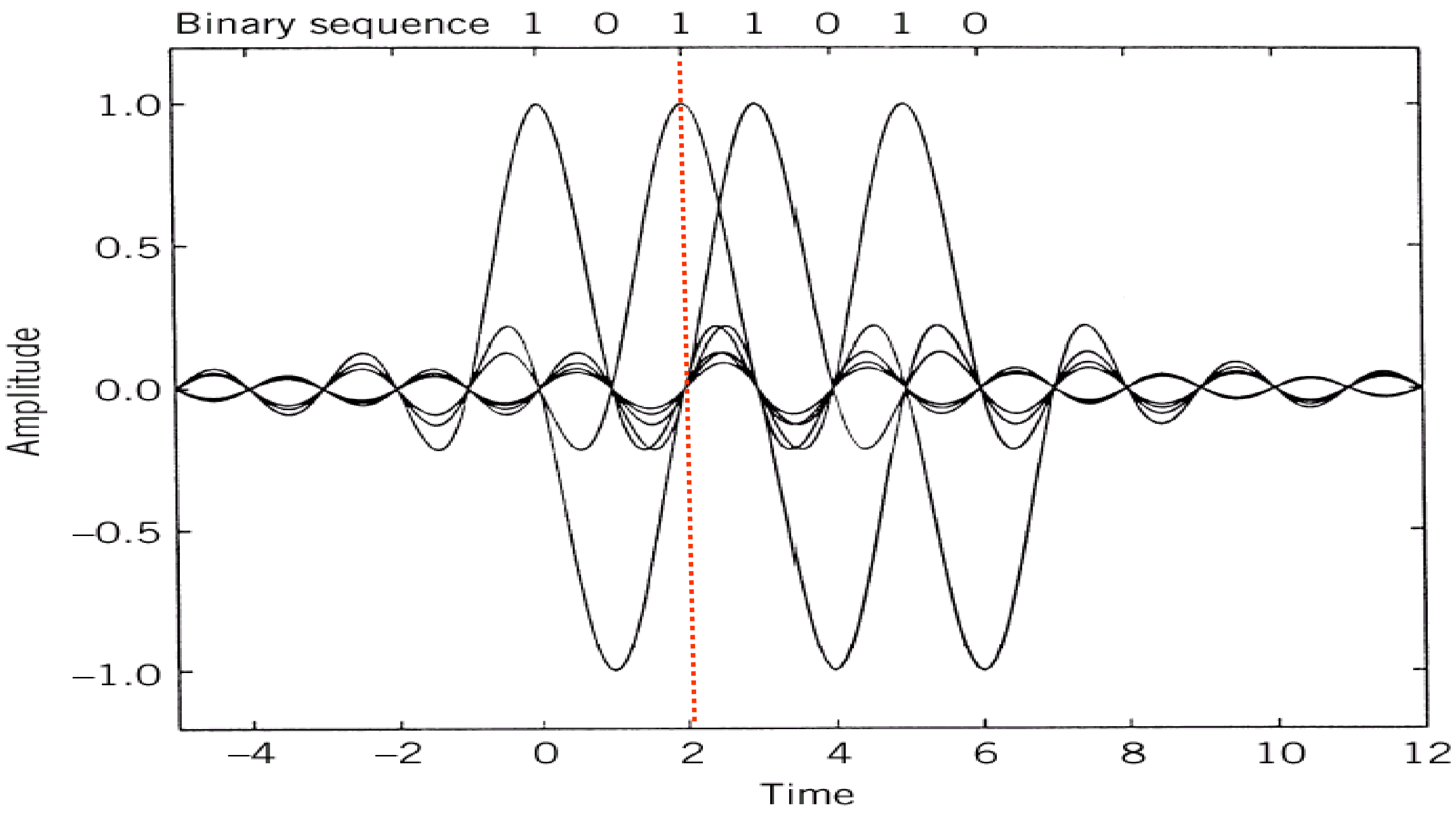


- Implementation I (analog):

Analog LPF

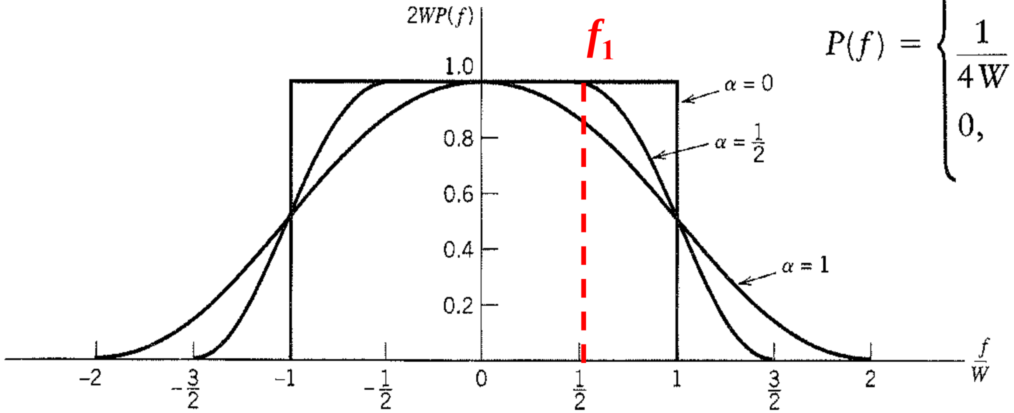


- Nyquist pulse shaping:



Inter symbol interference (ISI) free pulses

- Raised cosine (RC) pulse:

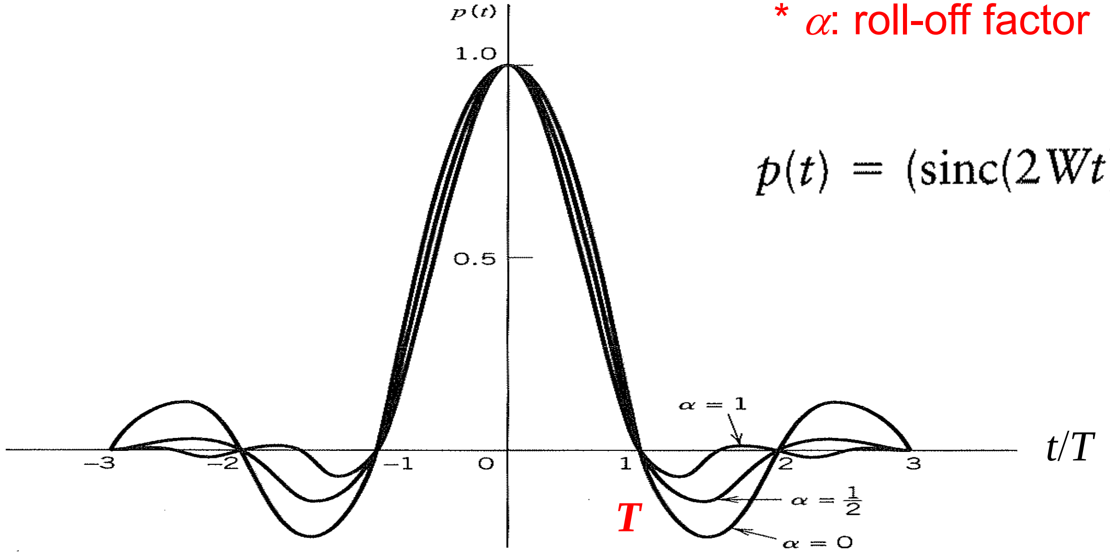


$$P(f) = \begin{cases} \frac{1}{2W}, & 0 \le |f| < f_1 \\ \frac{1}{4W}\left\{1 - \sin\left[\frac{\pi(|f| - W)}{2W - 2f_1}\right]\right\}, & f_1 \le |f| < 2W - f_1 \\ 0, & |f| \ge 2W - f_1 \end{cases}$$

$$\boxed{f_1 = W(1-\alpha)}$$

$$* \ \mathrm{sinc}(x) = \frac{\sin(\pi x)}{x}$$

* α: roll-off factor



$$p(t) = (\mathrm{sinc}(2Wt))\left(\frac{\cos(2\pi\alpha Wt)}{1 - 16\alpha^2W^2t^2}\right)$$

$$\boxed{T = \frac{1}{2W}}$$

T: symbol period,
Symbol rate = bandwidth

- Squared root raised cosine pulse (SRRC):

* RC: $P(f)$

SRRC: $\sqrt{P(f)}$

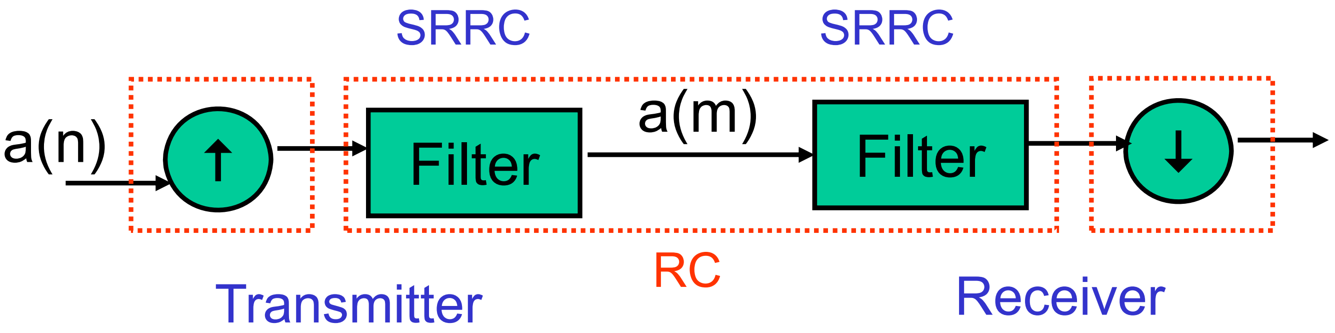
$$h(t)=\frac{4\alpha}{\pi}\frac{\cos((1+\alpha)\pi t/T)+T\sin((1-\alpha)\pi t/T)/(4\alpha t)}{1-(4\alpha t/T)^2}$$

- To discretize the pulse, we can let the signal in a T (symbol) interval be sampled with M points, i.e., t=n(T/M). Then,

$$h(n)=\frac{4\alpha}{\pi}\frac{\cos((1+\alpha)\pi n/M)+M\sin((1-\alpha)\pi n/M)/(4\alpha n)}{1-(4\alpha n/M)^2}$$

* One T has M samples

- Why SRRC:

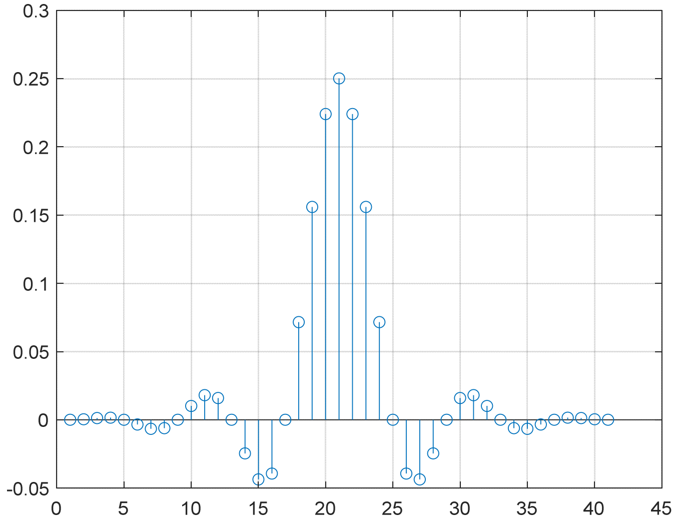


- Note that the pulse shaping filter here is an analog filter and it should have an IIR response.
- In reality, the RC and SRRC pulses cannot be generated with analog filters.
- For the time being, we just assume that there exist analog filters that can generate RC and SRRC pulse.

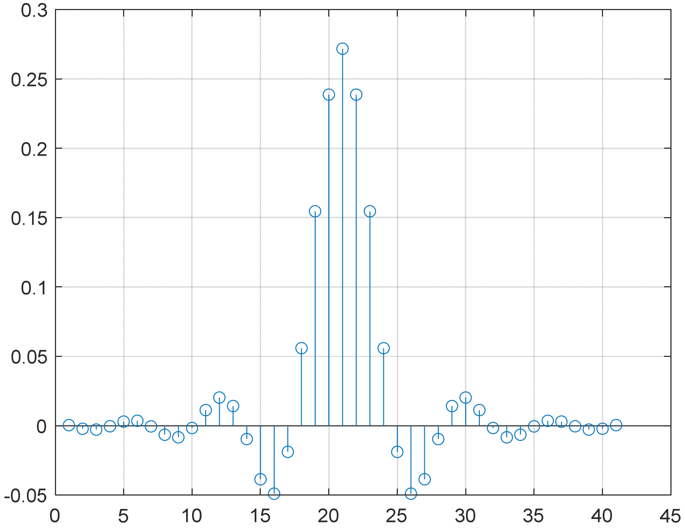
- Practice 1:
  - Use the method discretizing SRRC filter to discretize RC filter. Plot a RC pulse (three parameters: roll-off factor α, oversampling factor M, pulse span, S).
    - α=0.3, M=4, S=5 (you will have MS+1 sampling points)
- Practice 2:
  - Plot a SRRC pulse (α=0.3, M=4, S=5).
  - Check if the convolution of a SRRC and another SRRC pulses will give you a RC pulse.
  - Compare RC and SRRC pulses.

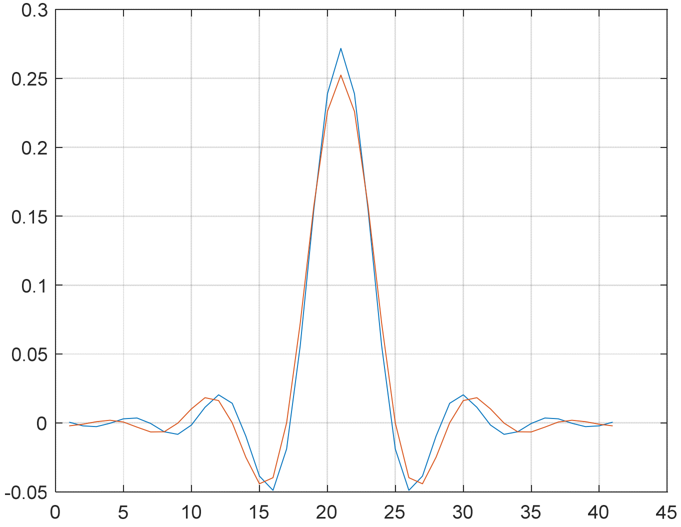
- Results:

RC



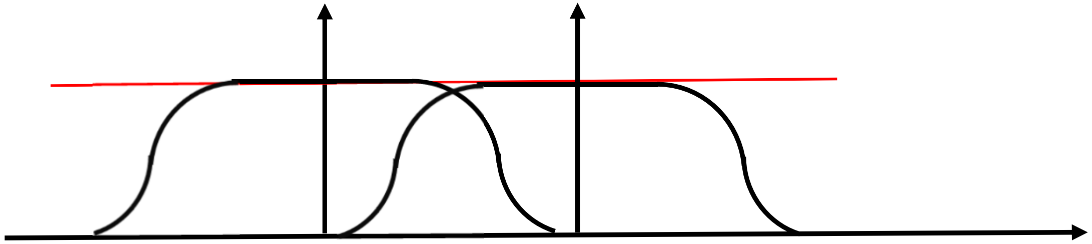
SRRC



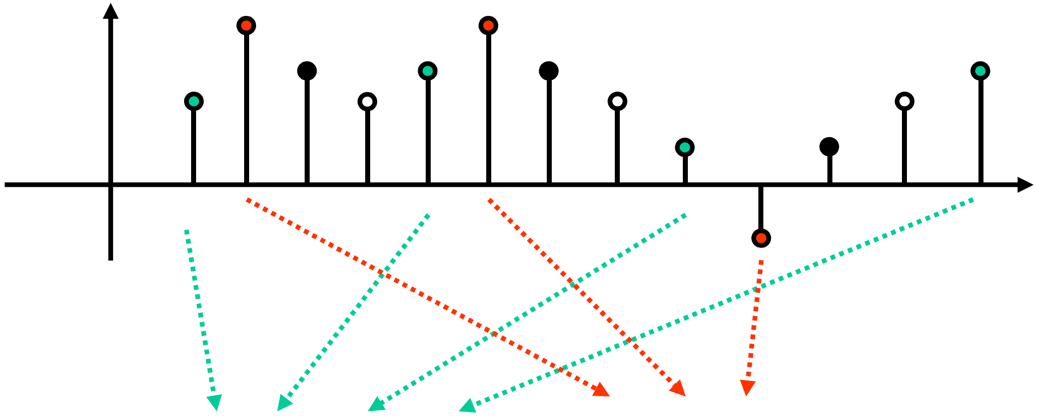


* DC gain is set to be one

- The sampled RC pulse is designed to have aliasing with symbol rate sampling (aliased spectrum is flat).



- Any pulse has this property is called Nyquist pulse (Nquist pulse does not satisfy Nyquist sampling theory).
- For a dowsampling with factor M, there are M possible results (M phases).



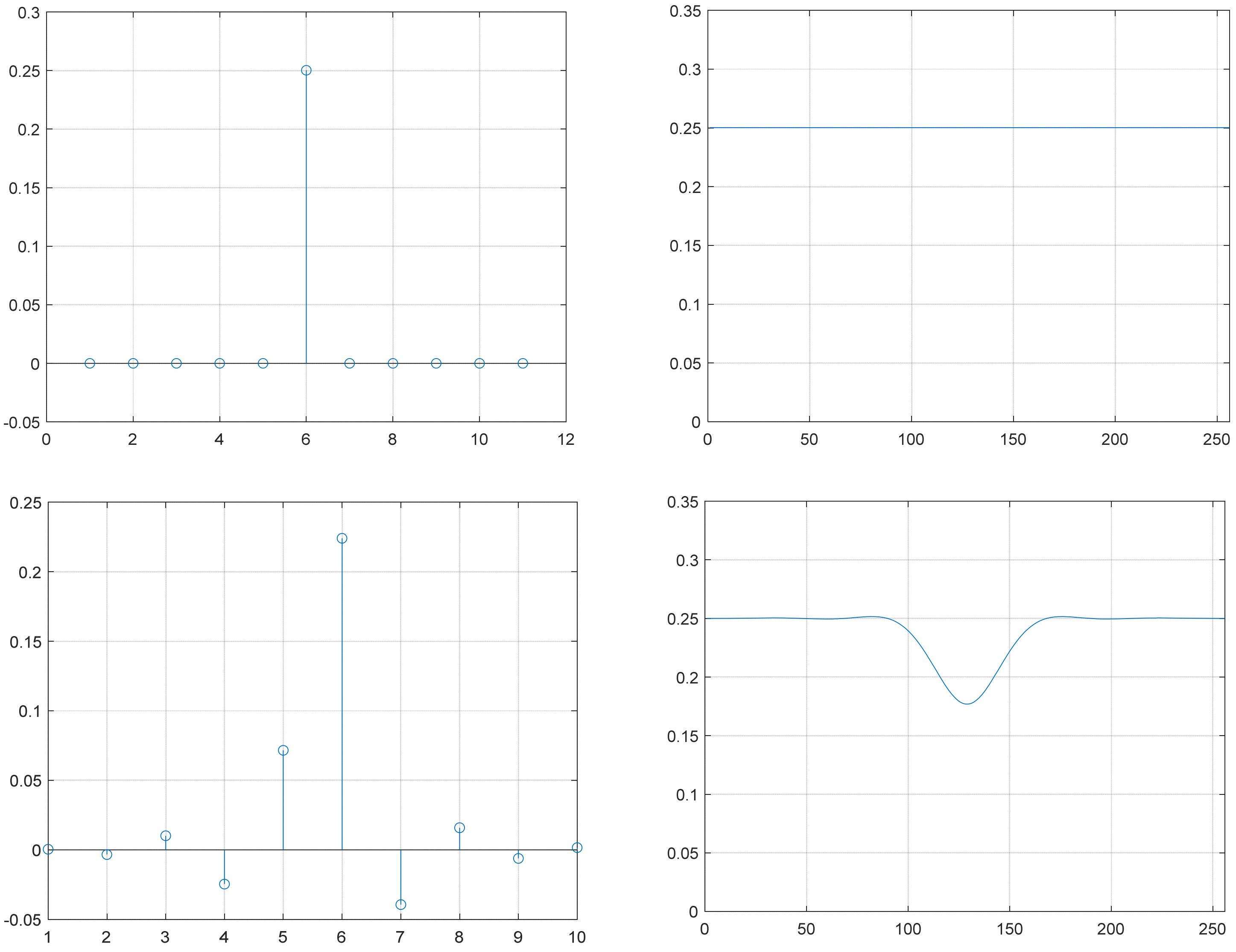
First phase (delay 0)

Second phase (delay 1)

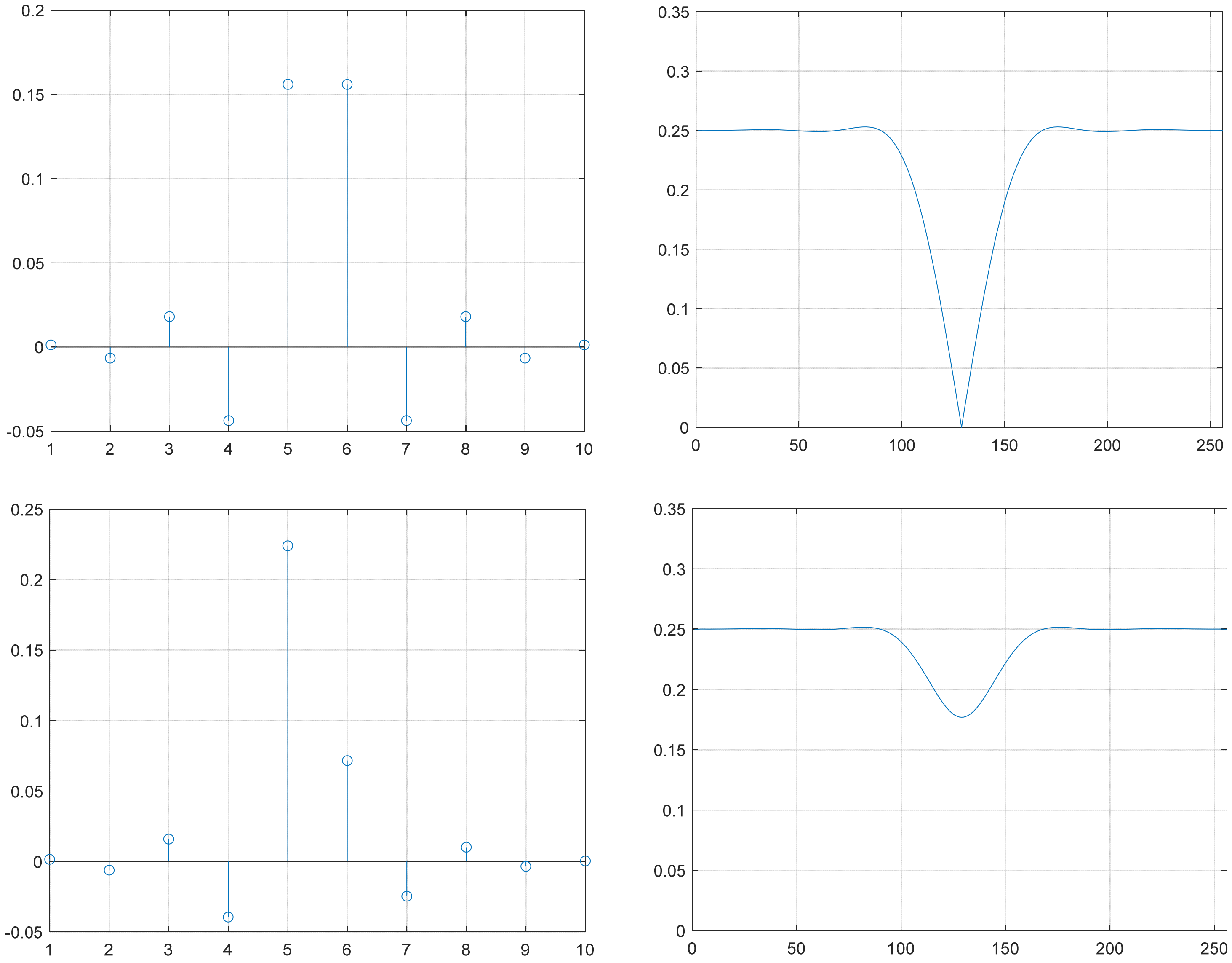
- There exists an sampling phase that will give the sampled RC pulse the flat spectrum and the corresponding time-domain signal is an impulse.
- Other sampling phases will not give an impulse, and inter -symbol interference (ISI) will occur.
- How can we know which phase is right for sampling?
  - Synchronization

- Practice 3:
  - Downsample (symbol rate) the RC pulse obtained in Practice with four different phases.
  - Observe the results and also see the corresponding spectrums (zero padding to have a size 256).

- Results:

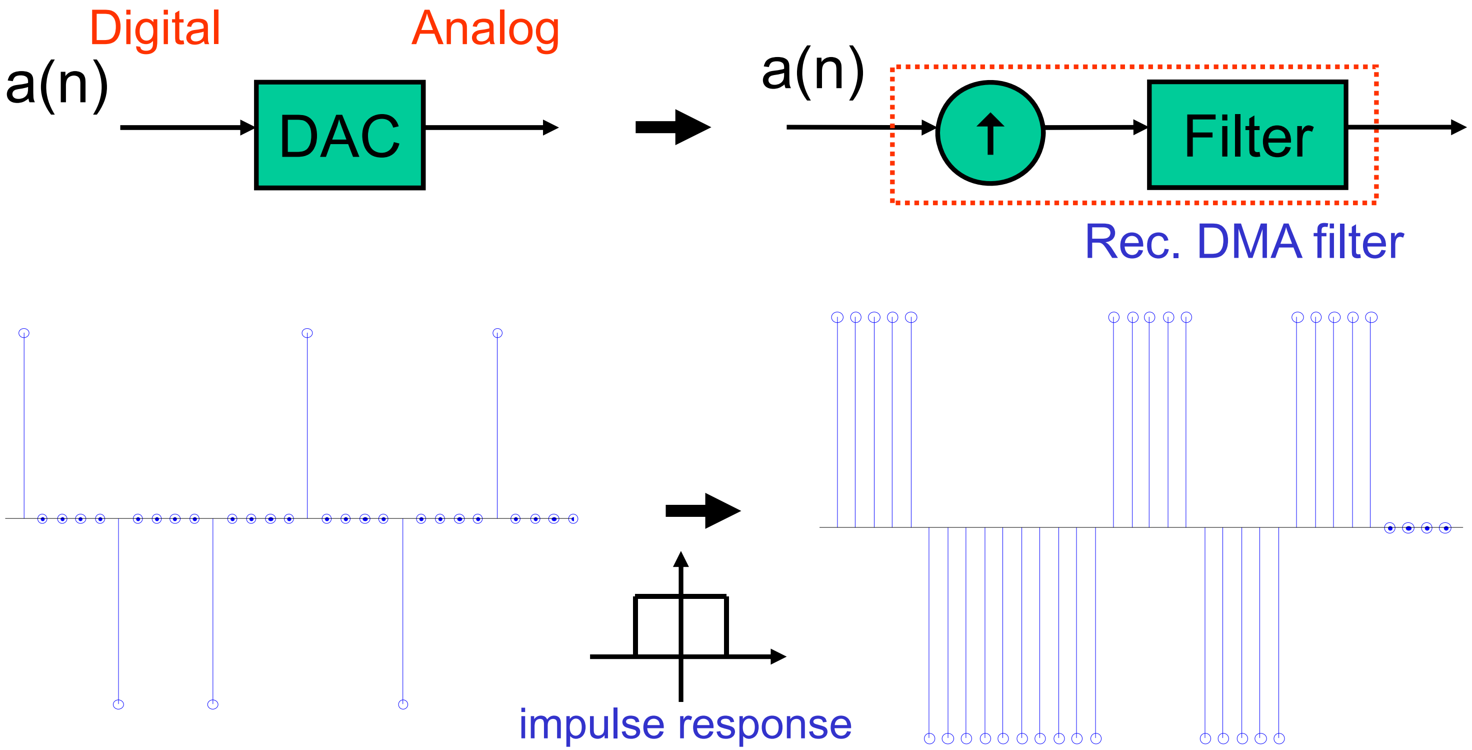

0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
0
2
4
6
8
10
12
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
50
100
150
200
250
0.25
0.2
0.15
0.1
0.05
0
-0.05
1
2
3
4
5
6
7
8
9
10
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
50
100
150
200
250

▪ Results:

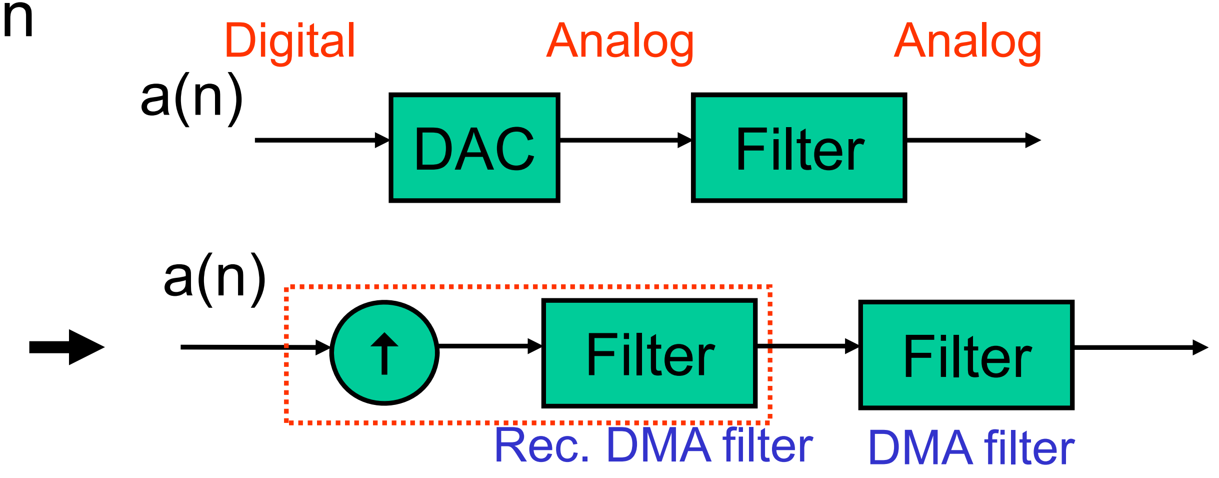


0.2
0.15
0.1
0.05
0
-0.05
1
2
3
4
5
6
7
8
9
10
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
50
100
150
200
250
0.25
0.2
0.15
0.1
0.05
0
-0.05
1
2
3
4
5
6
7
8
9
10
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
50
100
150
200
250

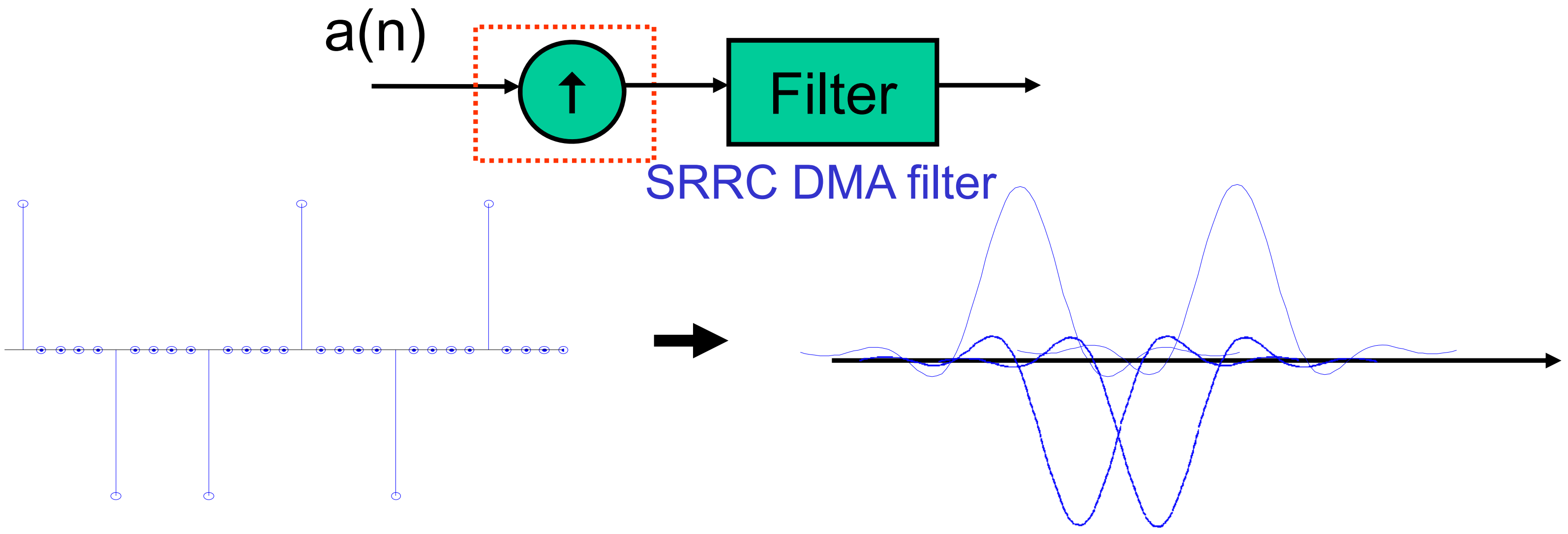
- Note that we use a digital pulse with a high sampling rate to model the analog pulse. As mentioned, we call this a digital modeled analog filter (DMA filter).
- The operation of a practical DAC can be modeled with a up-sampling operation followed by a rectangular filtering .



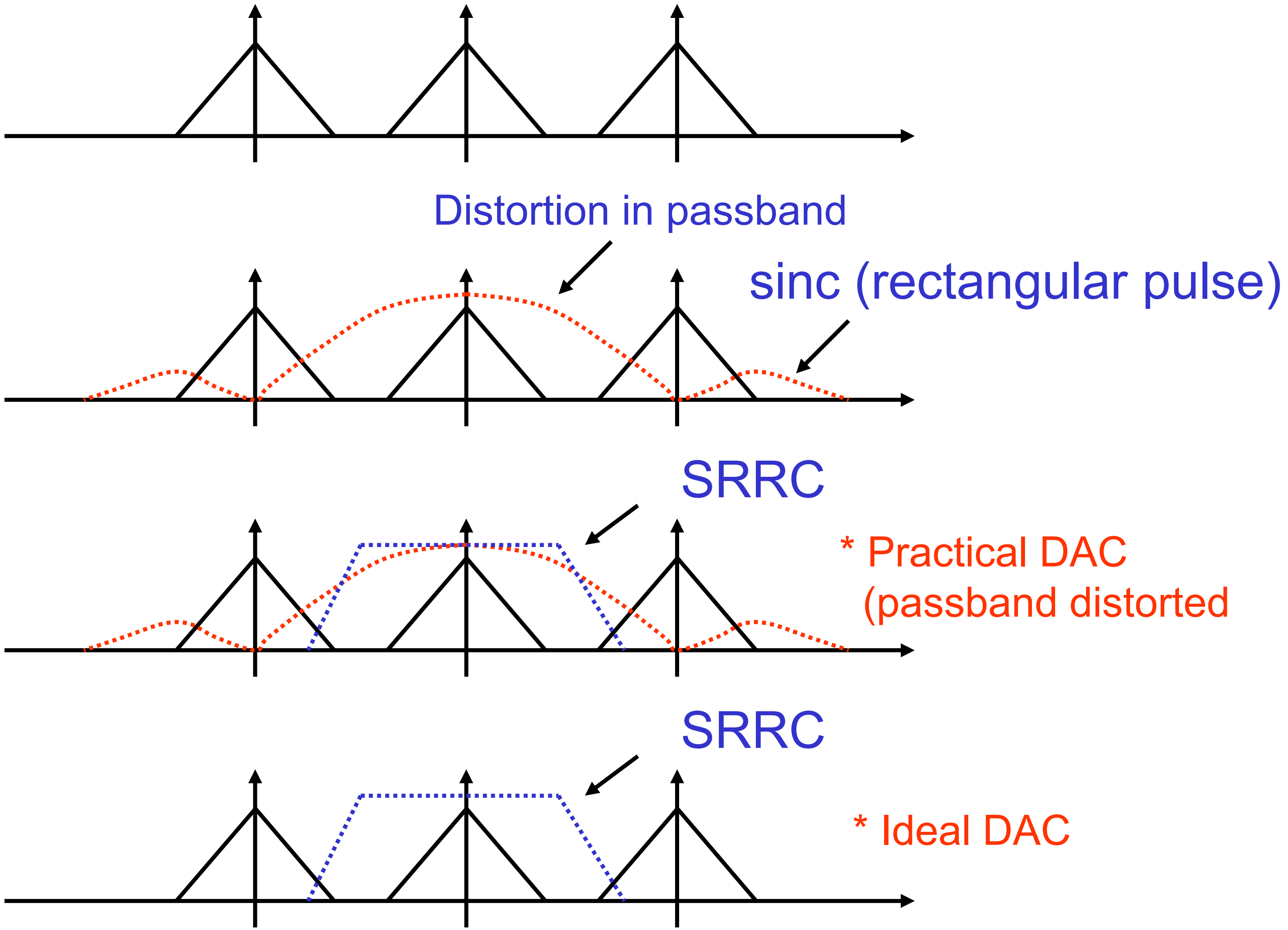
- Then



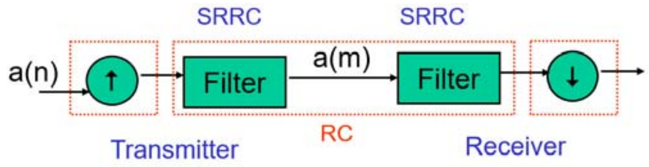
- The ideal DAC ignores the rectangular filtering such that there is no distortion in the passband.



- Spectrum of practical DAC:

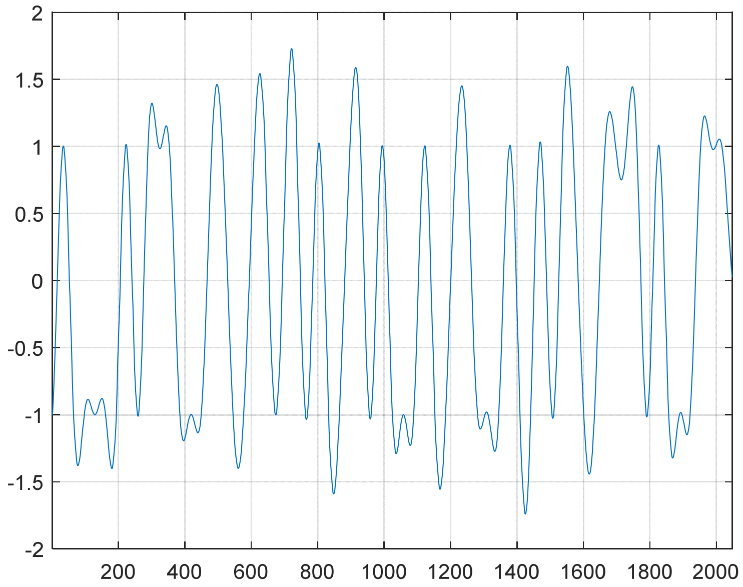


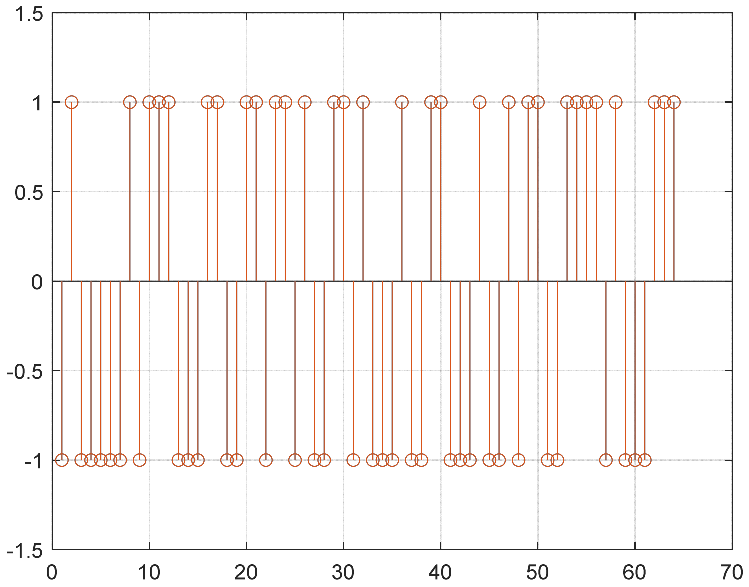
- Practice 4:
  - Make RC and SRRC pulse generation programs in Practice 2 as functions (e.g., rcfun, srrcfun).
  - Given a BPSK sequence (S7P4.mat), conduct the RC pulse shaping operation with an ideal DAC (upsampled by a factor of 32).
    - RC pulse: α=0.3, M=32, S=5



  - Conduct a dowsampling operation (ADC) to see if you can obtain the original signal (by a factor of 32).
    - Which phase is the right one and why?
- Practice 5:
  - Replace the ideal DAC with an practical DAC in Practice 3 and redo the simulations.
  - Compare the response of SRRC and SRRC convolved with rectangular pulses.

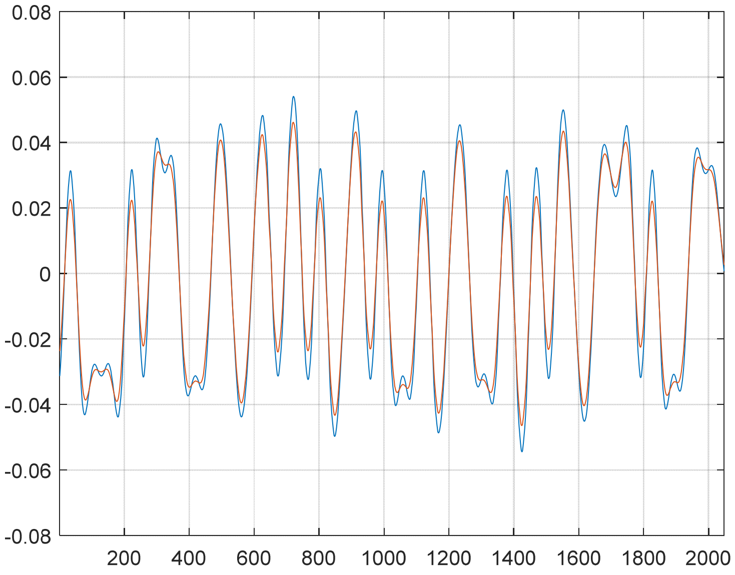
- Results (Practice 4):

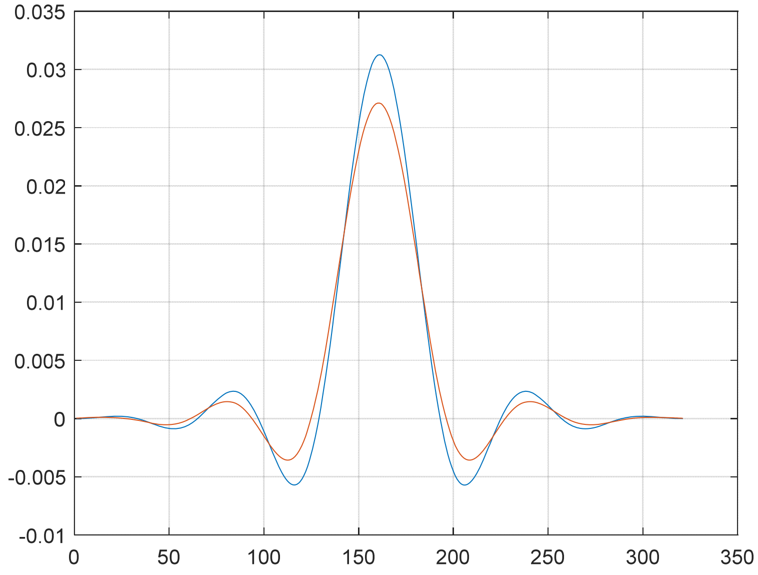


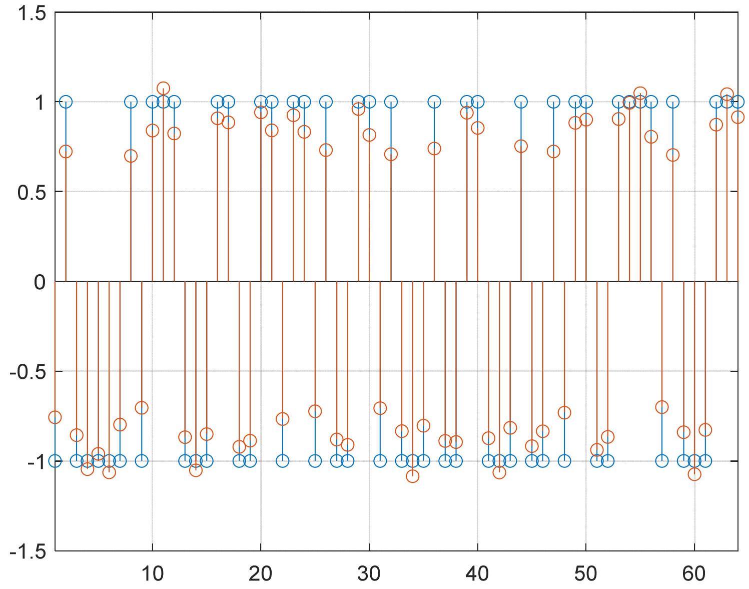


  - The filter delay has to be considered first.
  - With the span we define, the first phase turns out to be the right one

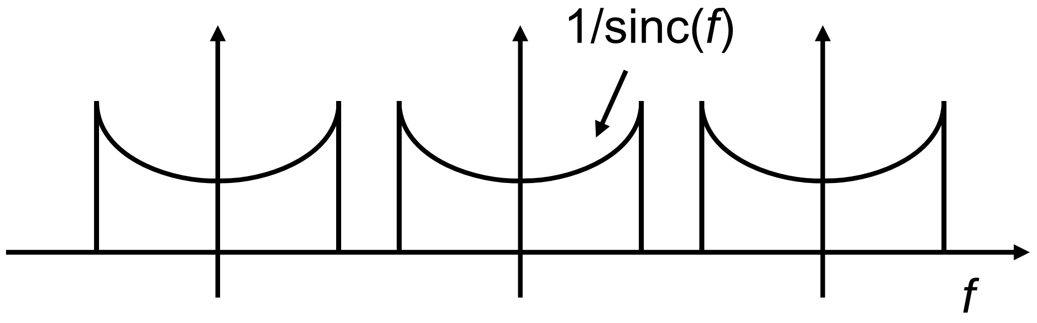
- Results (Practice 5):





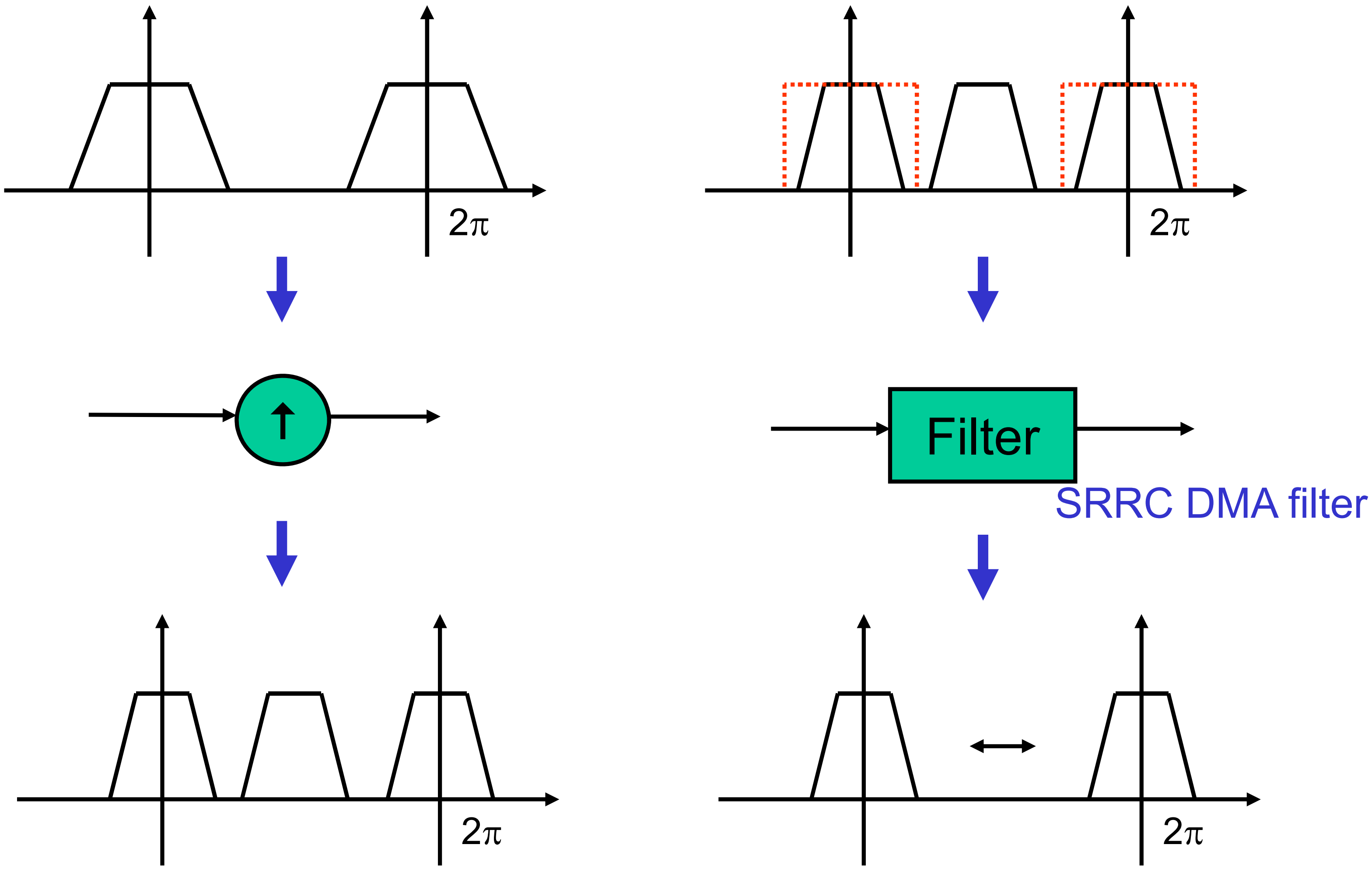


- Since the practical DAC distorts the signal in passband, a compensation filter (pre-equalizer) may be added at transmitter to correct this distortion.



- The design of the filter may need further investigation. For simplicity, we mainly use ideal DAC in our simulations.

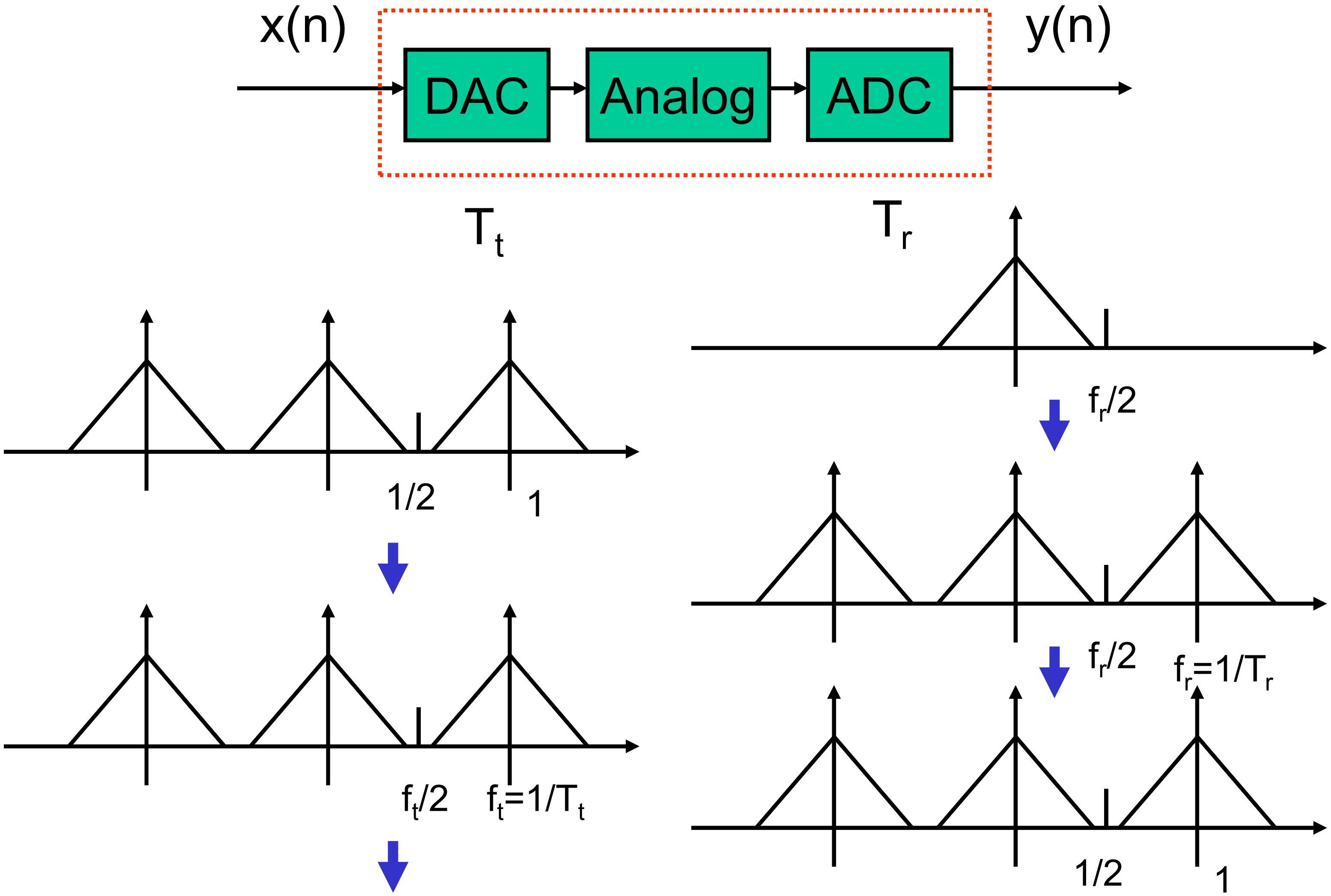
- The operation of DAC in the frequency domain:



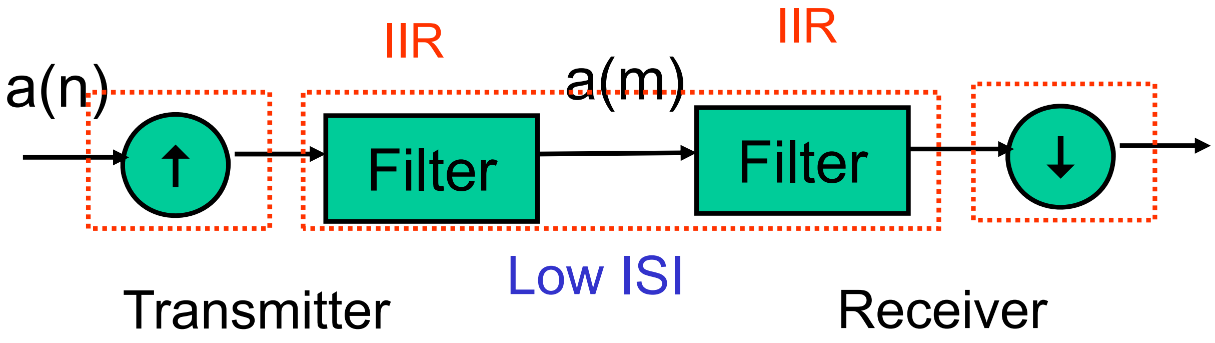
SRRC DMA filter

Approximate analog spectrum!

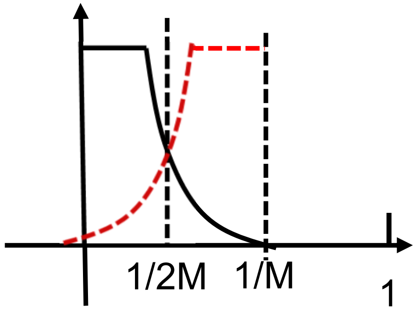
- Relationship of analog and digital frequency.



- The response of the analog LPF for the DAC can be used as the pulse shaping (generating analog symbol) also.
- Unlike raised cosine pulses, the Nyquist criterion may not be perfectly met for IIR pulse shaping. As a result, ISI may occur. With careful design, low-ISI IIR pulse can be obtained.



- Design guideline:

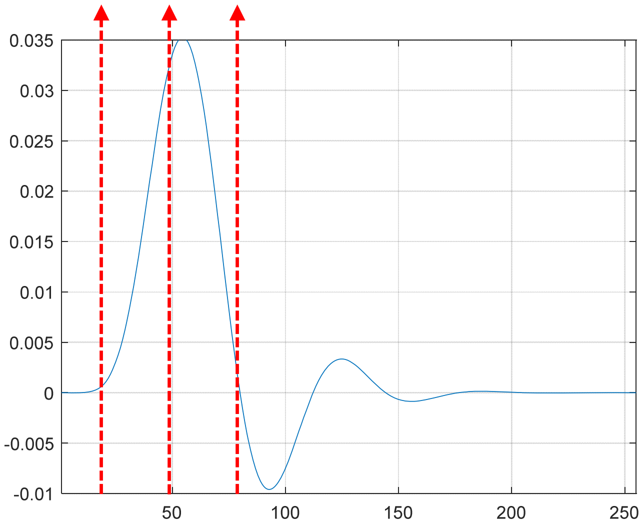


- Use Butterworth
- Mainlobe width should be around 2M wide (smaller)
- Passband edge ~ 1/(2M)
- Stopband edge ~ 1/M
- Stopband ripple ~ -10 ~ -20dB
- Passband ripple ~ -1 ~ -5dB

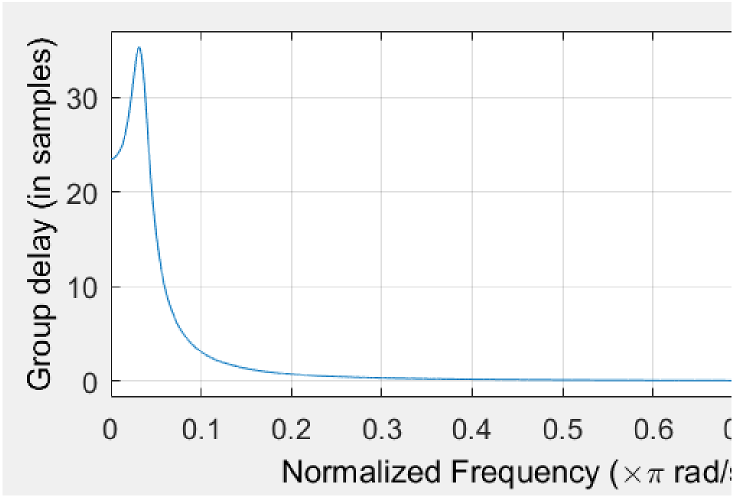
- Practice 6:
  - Design an IIR DMA filter for a upsampling process with L=32.
  - Given a BPSK sequence (S7P5.mat), use the designed filter as the pulse shaping filter.
  - Let the same filter be used at the receiver. Conduct the detection at the receiver.
  - Adjust the cutoff to obtain low-ISI pulses.

* Note that the delay of the IIR filter can be obtained from its group delay

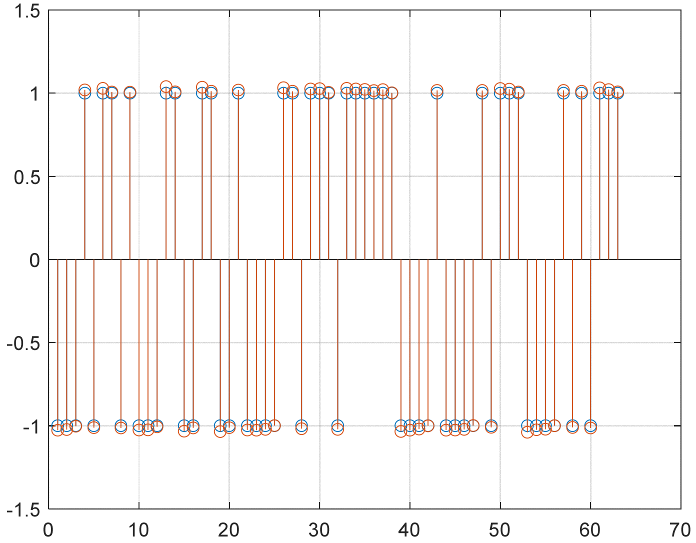
- Results:



48-32 48 48+32

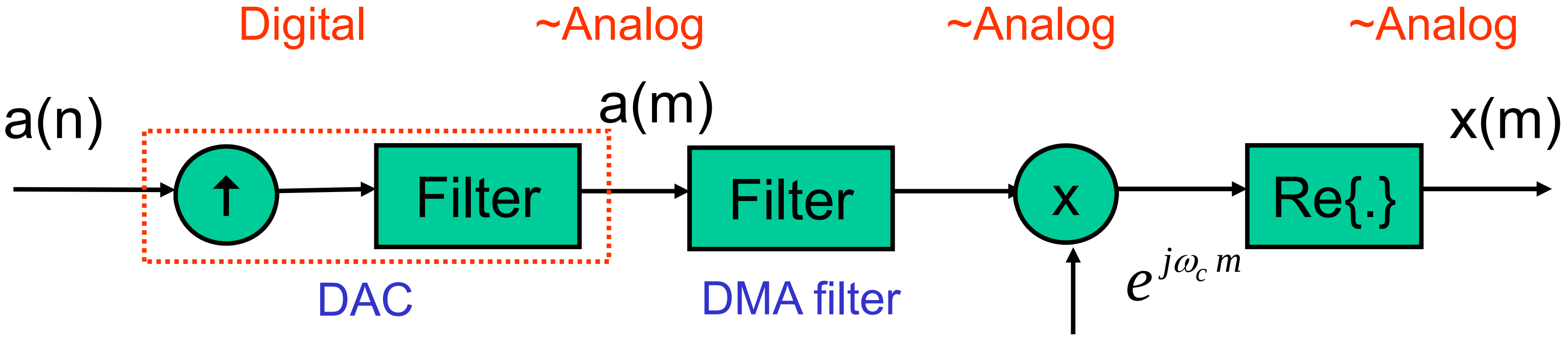


Group delay: IIR filter



Sampling phase (delay): 24x2=48

- Up-conversion:

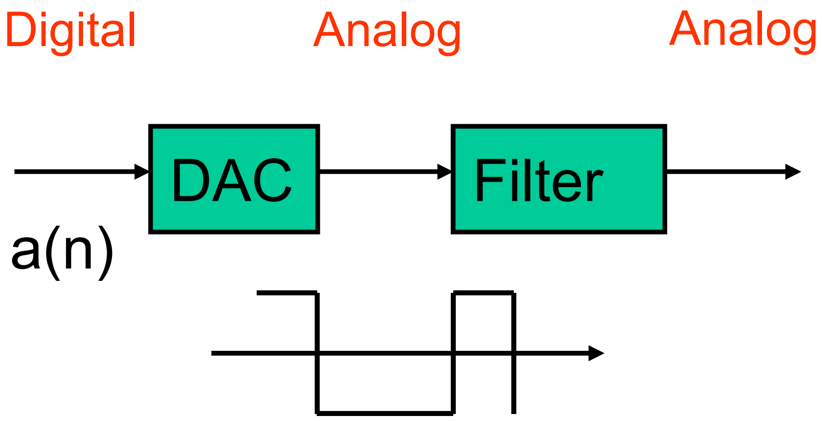


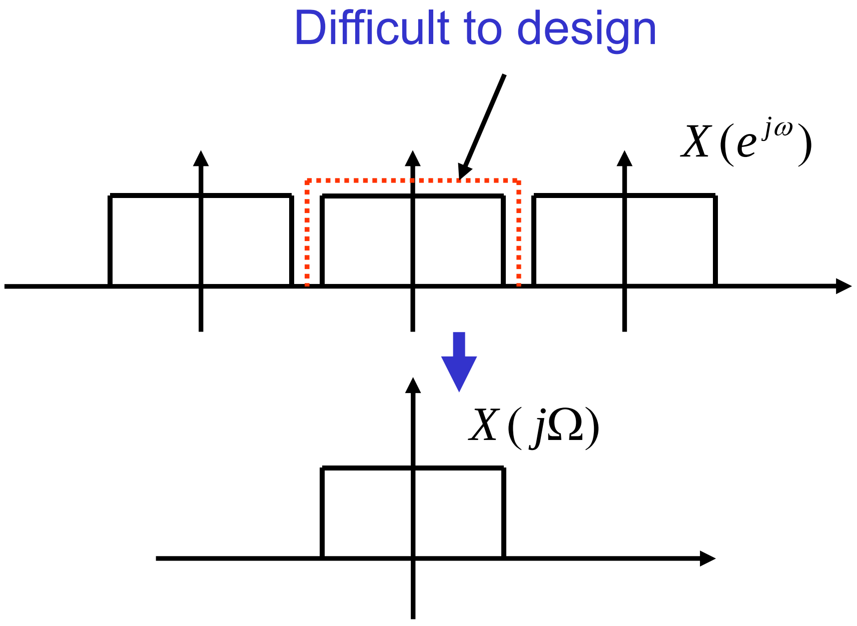
- Calculation of the carrier frequency:
  - The sampling rate is MR (R: symbol rate, MR→1).
  - If the carrier frequency is f_c (digital frequency), then the corresponding analog carrier frequency is MRf_c.
- Example:
  - Let the symbol rate be 1MHz, the upsampling factor be 100, and the carrier frequency be 1/4.
  - The analogy carrier frequency is then 25MHz.

- Homework:
  - Conduct SRRC pulse shaping for a given sequence (S7HW.mat).
  - Use the practical DAC (the upsampling factor is 64) to do the job.
  - Let the symbol rate be 1MHz, the carrier frequency be 8MHz. Conduct the up-conversion operation in the equivalent digital domain.
  - Observe the up-converted spectrum to see if your design is correct.

Lab. 8 Transmit Filtering/Up conversion II

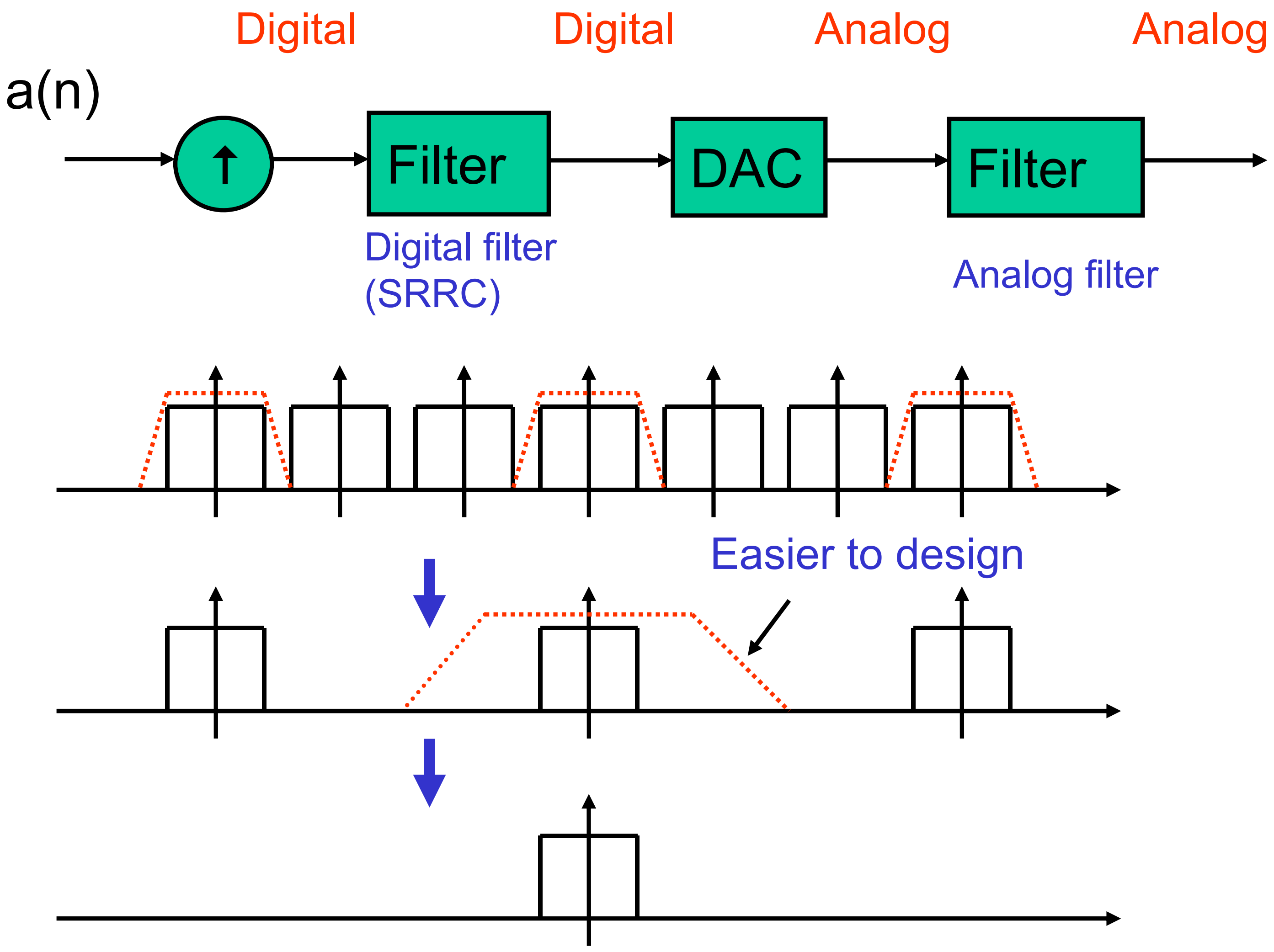
- Transmit pulse shaping:



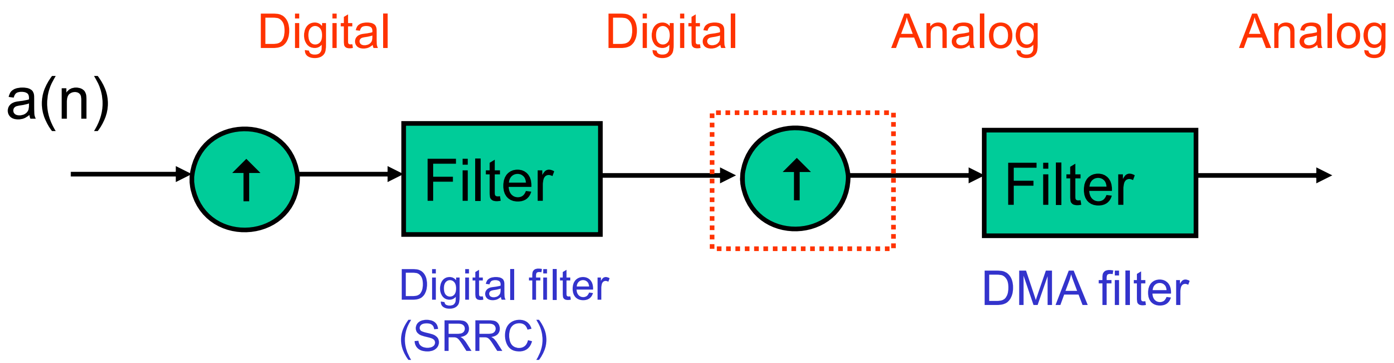


- The analog filter may be difficult to implement due to its stringent requirements.
- One way to solve the problem is to use a digital filter sharing the analog filtering operation (hybrid filtering).

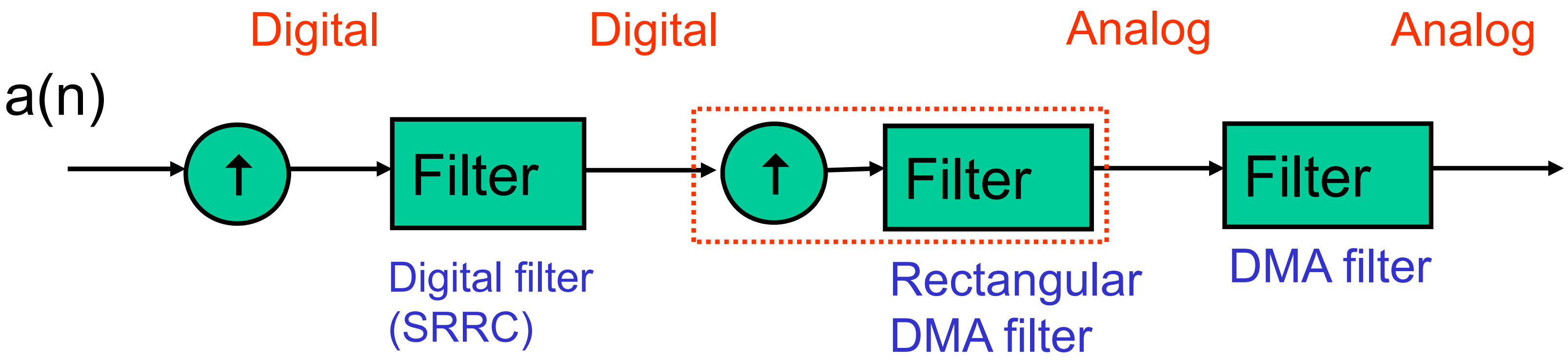
- Pulse shaping implementation II (hybrid):



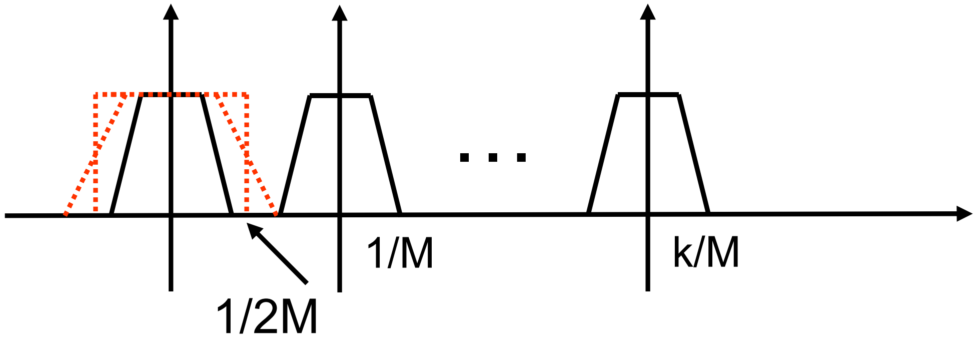
- For an ideal DAC, we then have



- For an practical DAC, we then have

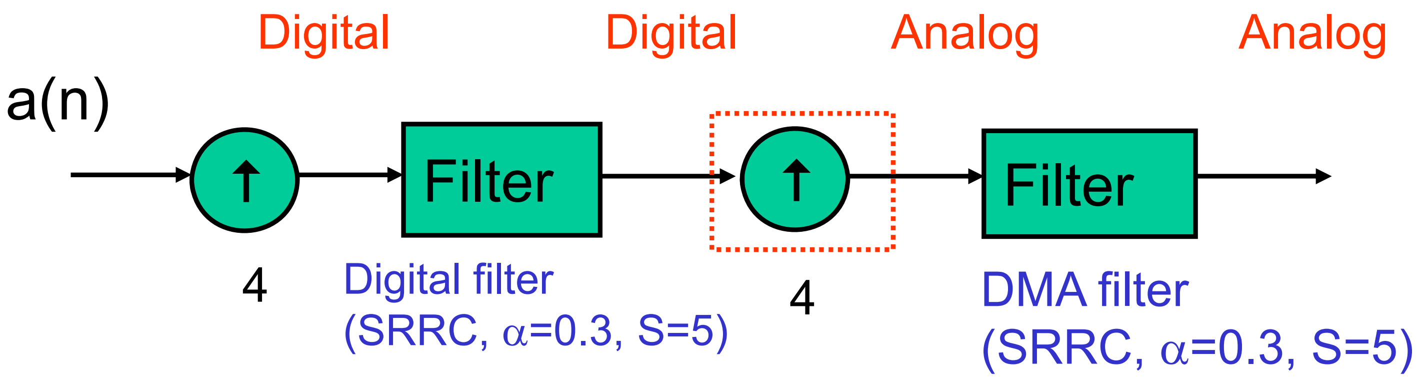


- Thus, we have three frequencies in the system.
  - The symbol rate
  - The up-sampled rate I (digital)
  - The up-sampled rate II (analog)
- For the ease of simulation, the SRRC filter can first be used as the lowpass DMA filter (assumed to be analog).
  - Let M be the upsampling factor.
  - The mid-point of passband and stopband edges is 1/2M.



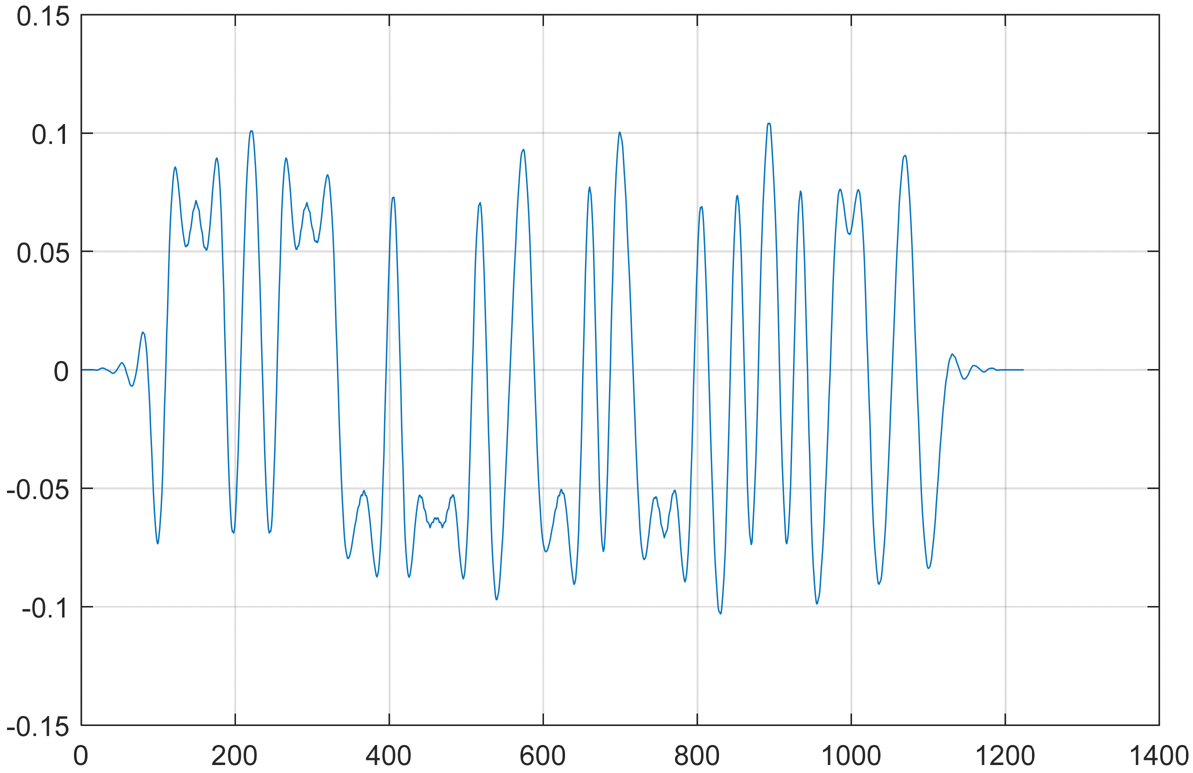
- Note that the analog SRRC filter is not realizable.

- Practice 1
  - Conduct SRRC pulse shaping (in the digital domain) for a given BPSK sequence (S8P1.mat). The SRRC filter is also used as the DMA filter. The up-sampling factor for digital filtering is 4, that for the DMA filter is 16, and the DAC is ideal



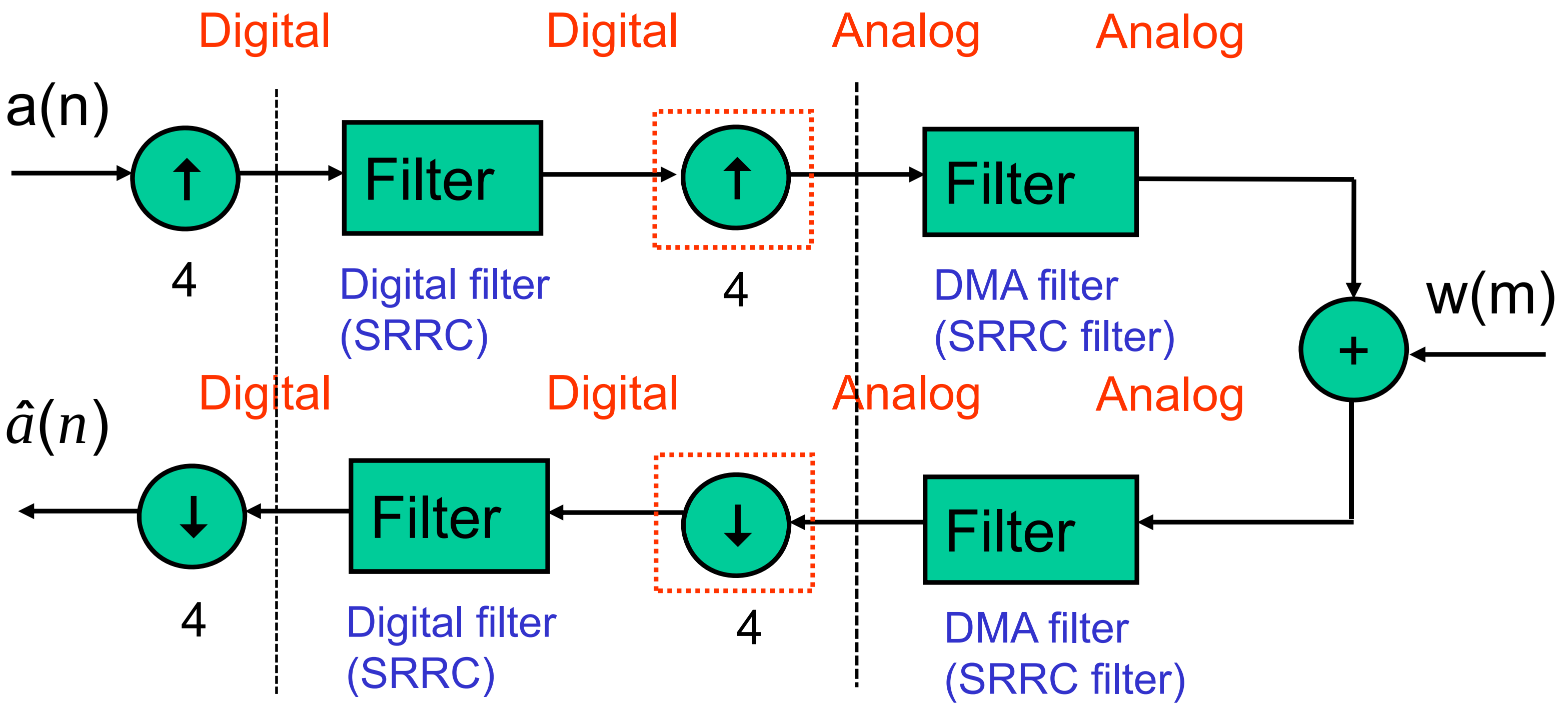
Note: digital SRRC is upsampled by a factor of 4 and DMA SRRC is also upsampled by a factor of 4.

- Result:



- A filter may have the non-causal property making a delay required in the output.
  - For linear-phase FIR filters, the delay is (L-1)/2. If multiple FIR filters are cascaded, delays can added up directly
  - For IIR filters, the delay is found by the derivative of the phase response (group delay).
- As mentioned, for a dowsampling with factor M, there are M possible results (M phases).
- There will be one phase giving the best result. Without synchronization, the trial-and-error approach may be used to find the phase (based on group delay).
- Note that If downsampled signal is not aliased, then the sampling phase is not that critical.

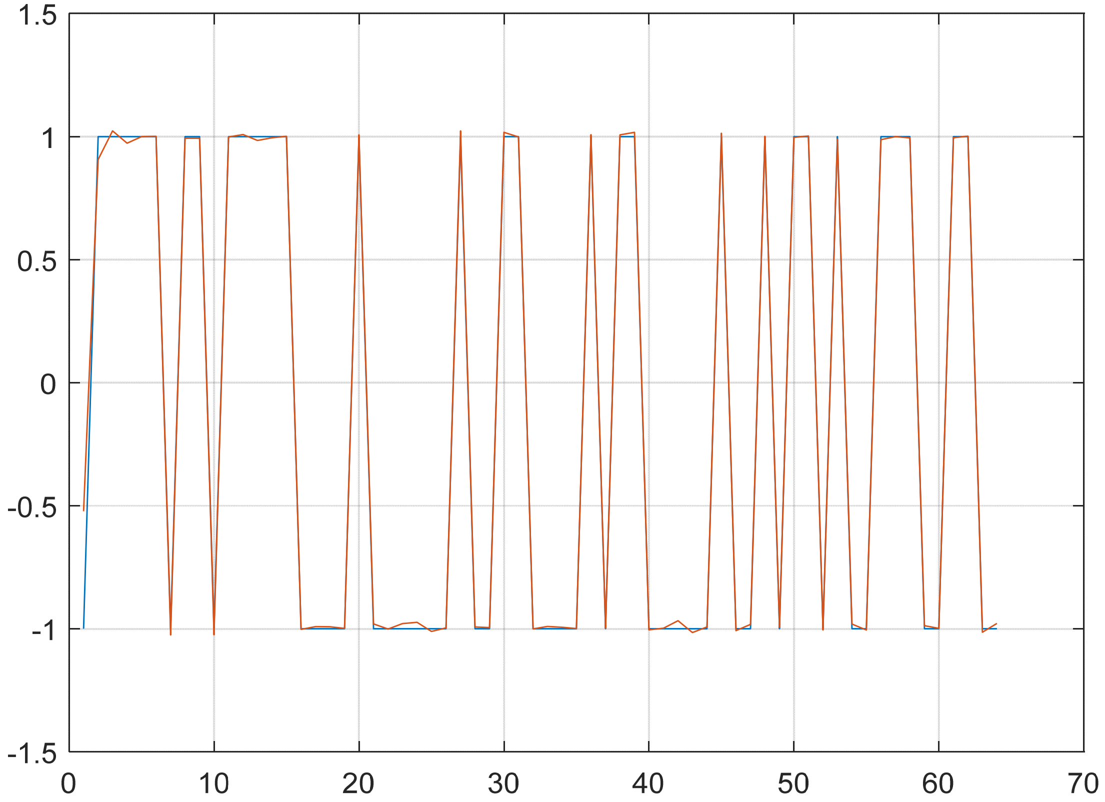
- Practice 2:
  - Using the setting in Practice 1, conduct the signal recovery at the receiver side.



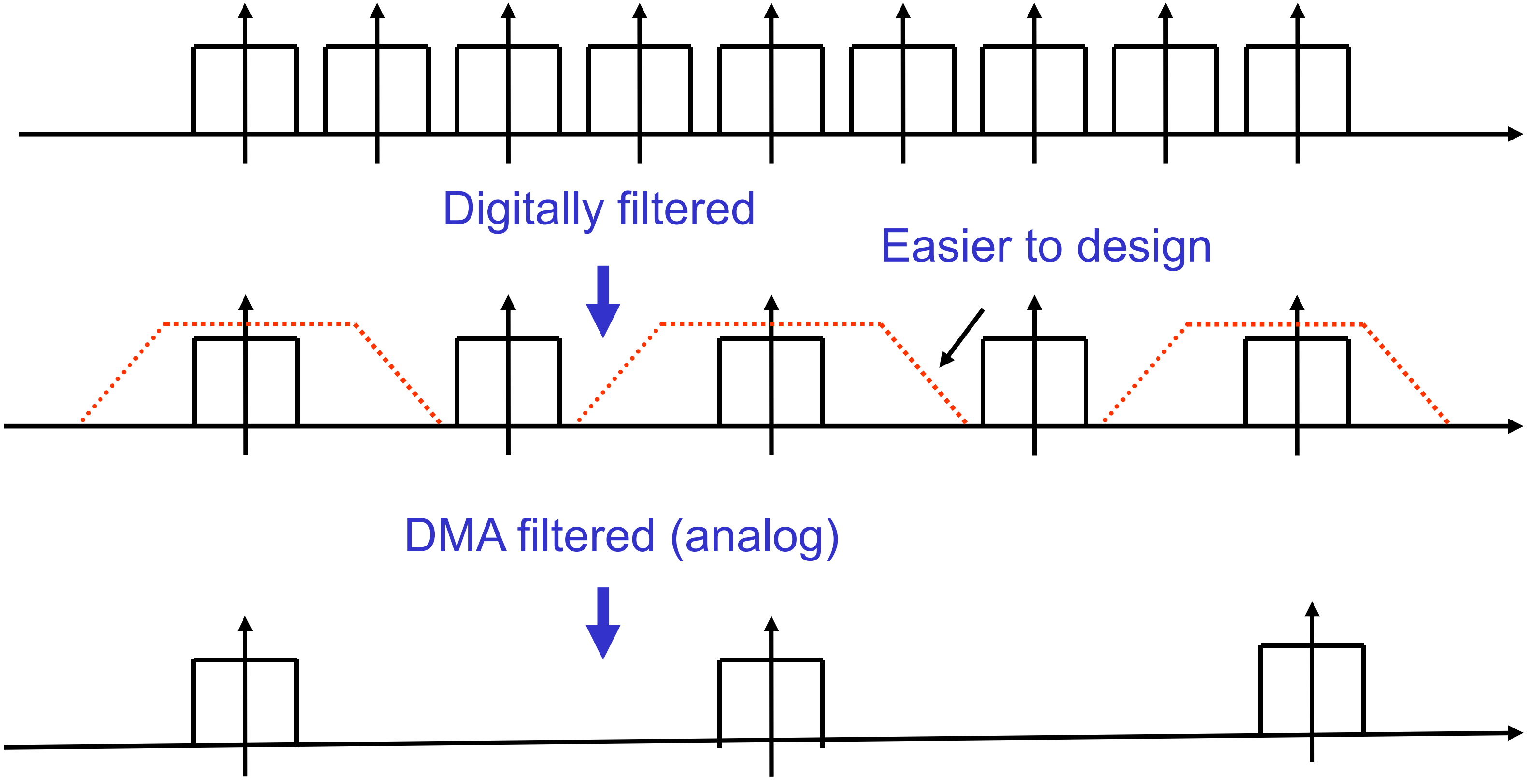
Delay (sampling phase):
(L-1)/2x2=L-1

Delay (sampling phase):
(L-1)/2x2=L-1

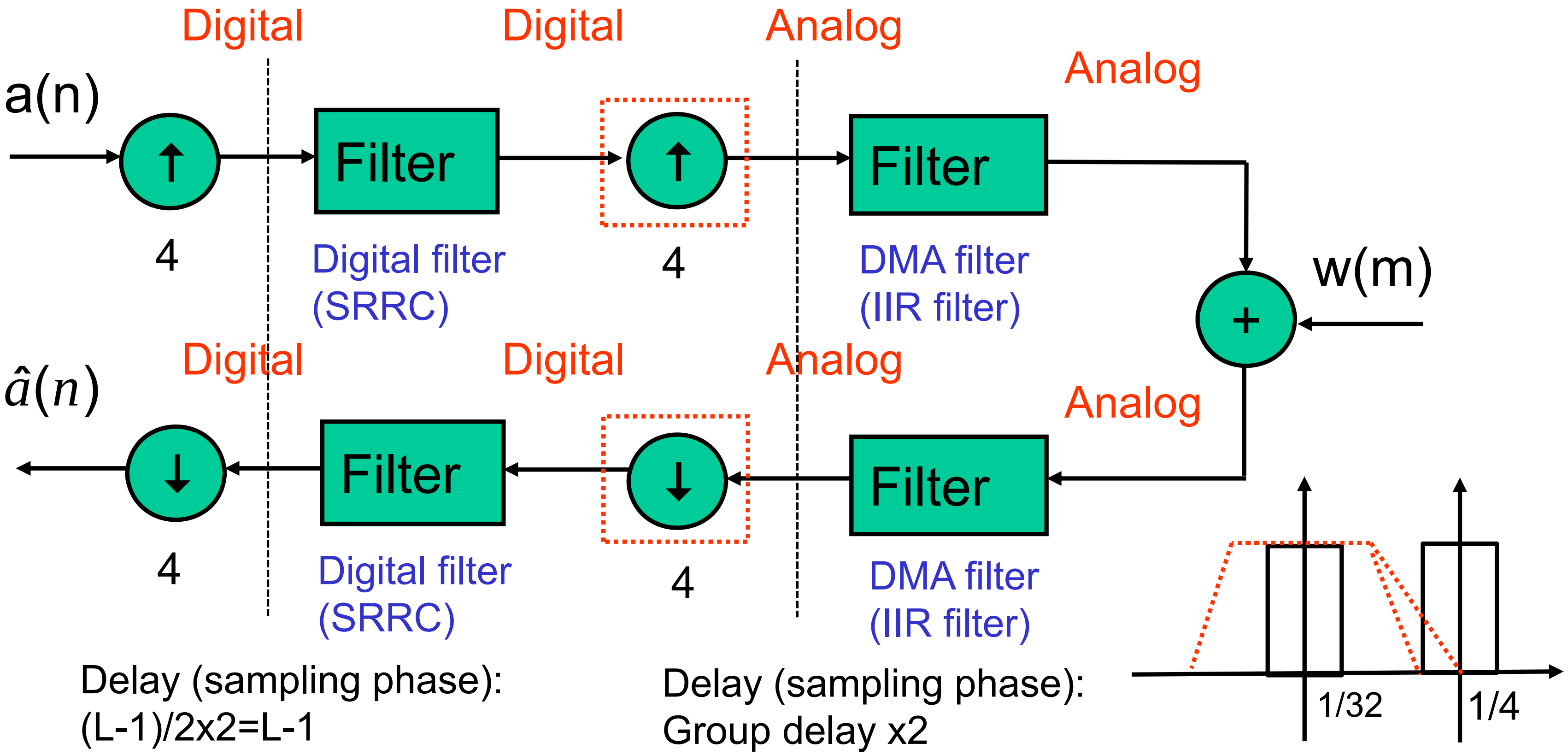
- Result:



- Note that the DMA filter we design is to approximate the analog lowpass filter.
- The only requirement of the DMA filter is to remove the filter images.



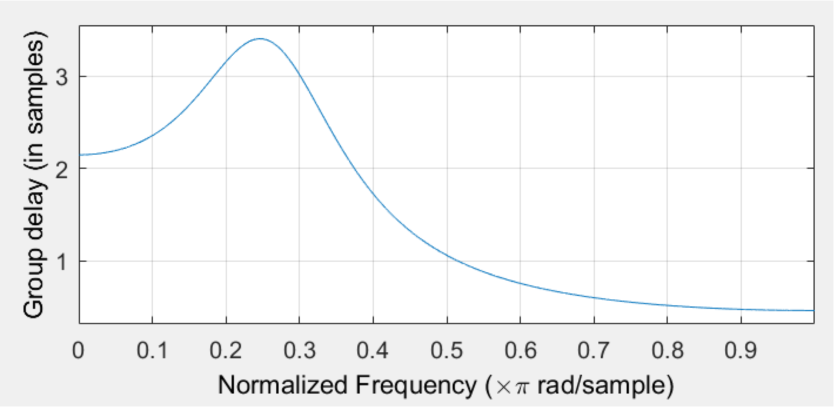
- Practice 3:
  - Design a lowpass IIR DMA filter (replace the SRRC DMA filter) for the system in Practice 2.
  - Conduct the simulation again.



Delay (sampling phase): (L-1)/2x2=L-1

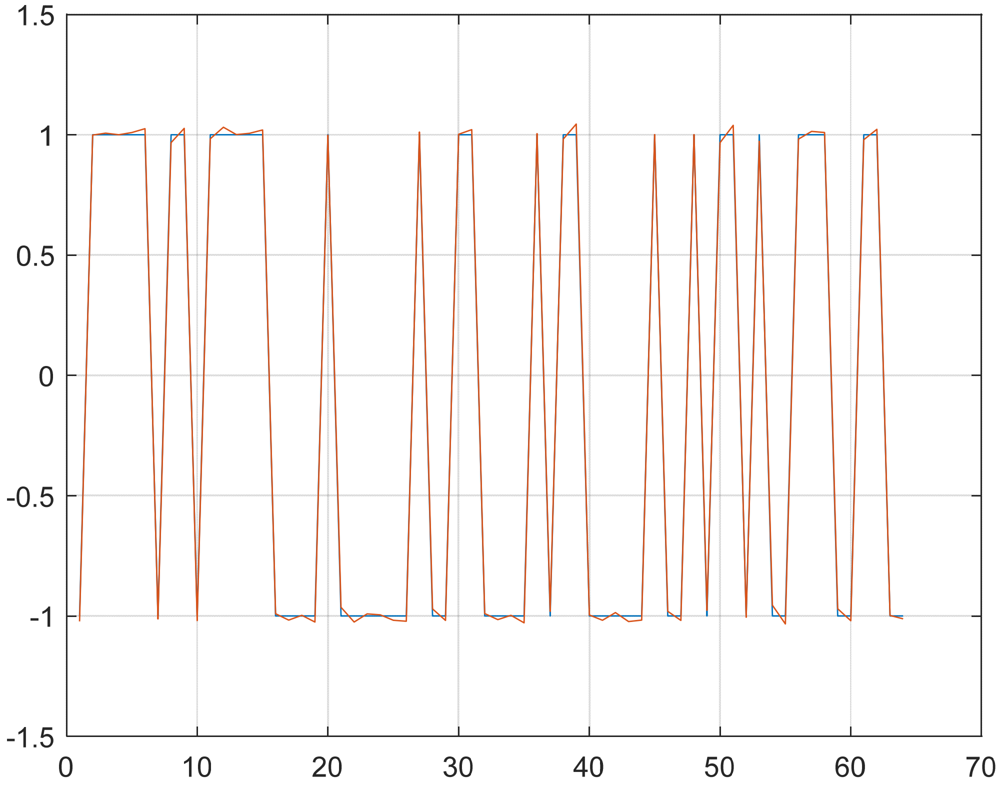
Delay (sampling phase): Group delay x2

- Result:

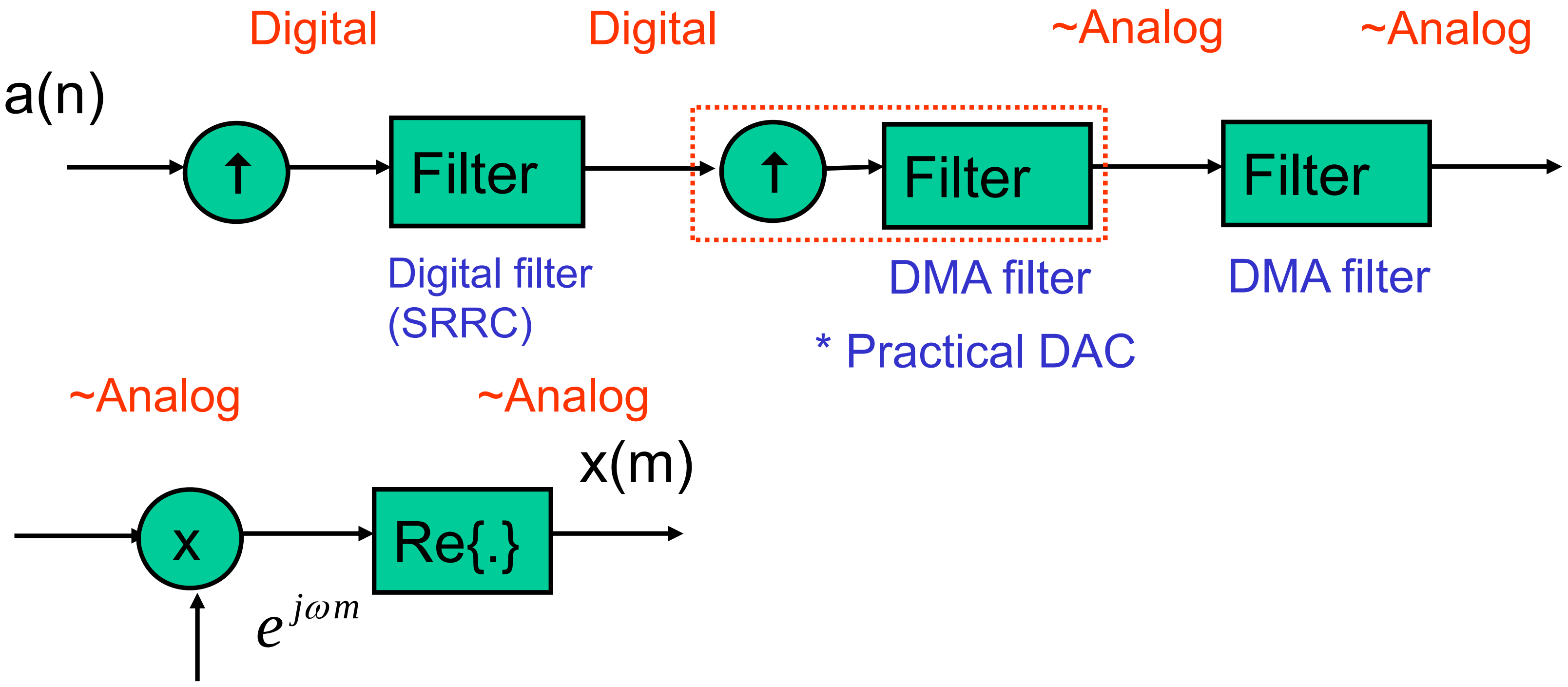


Group delay:
IIR filter

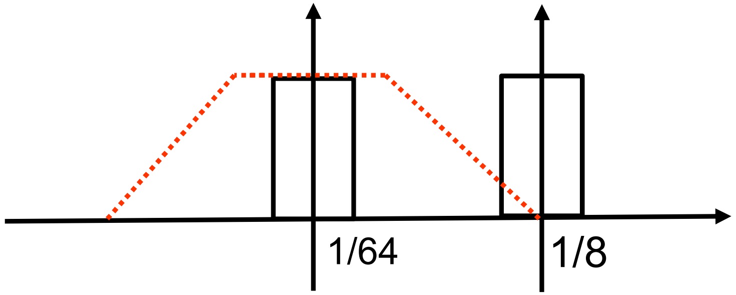
Delay:2x2=4



- Up-conversion:

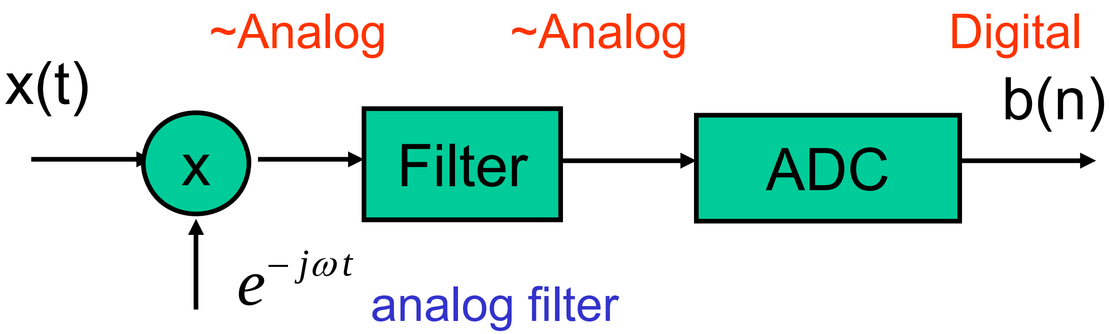


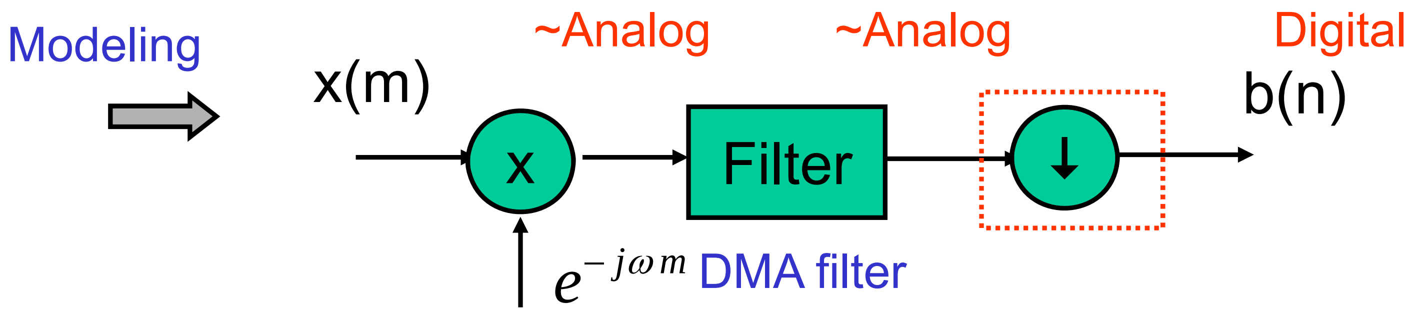
- Homework:
  - Let the symbol rate for a system be 1MHz, the sampling rate of the DAC be 4MHz, and sampling rate for DMA filter be 32MHz.
  - Let the modulation be BPSK, and the digital pulse shaping is SRRC.
  - Design an IIR DMA filter with 5 coefficients (second order and one section) that maximizes stopband attenuation.
  - Conduct the transmit and receive operation for a sequence (without carrier).



Lab. 9 Receive Filtering/Down conversion

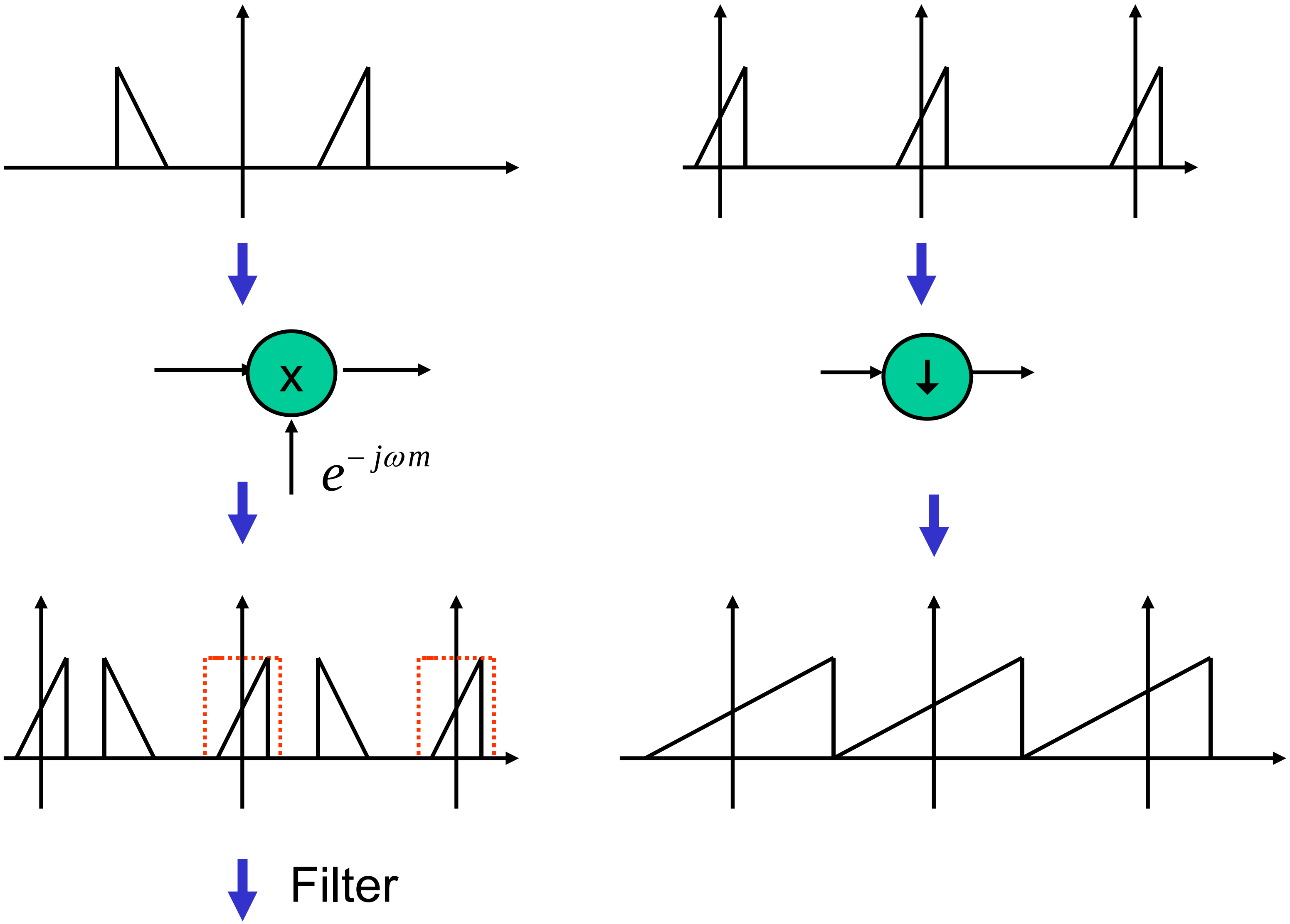
- Down-conversion (direct conversion):





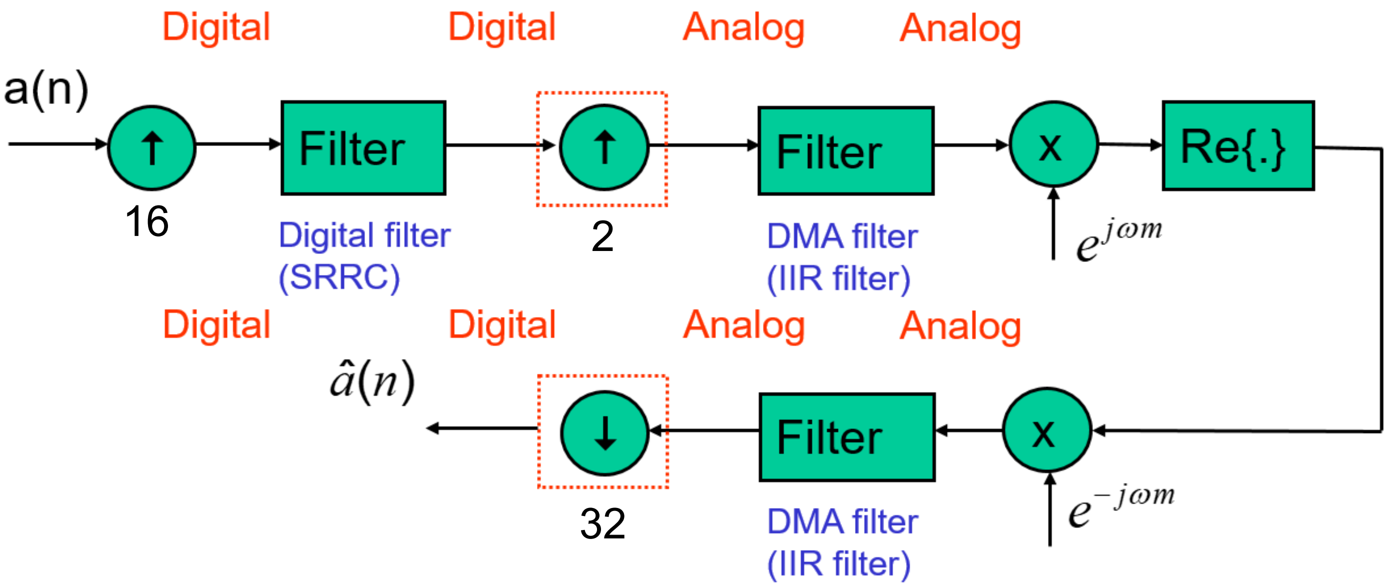
- The purpose of the DMA filter here is for noise filtering.

- Spectrum relationship (digitally modeled):

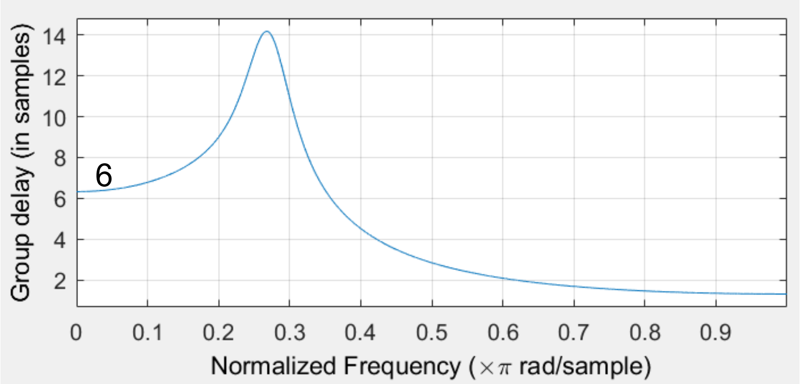


Filter

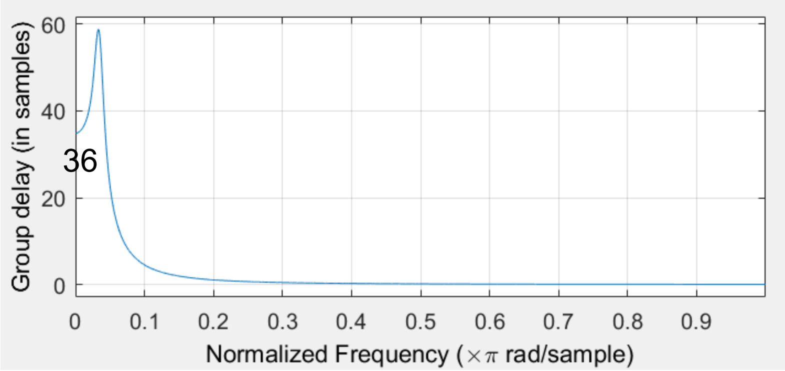
- Practice 1:
  - Let the symbol rate of a BPSK system (S9P1.mat) be 1MHz, the sampling rate of the DAC be 16MHz, the sampling rate for DMA filter be 32MHz, and the carrier frequency be 8MHz.
  - Design a SRRC filter and an IIR lowpass DMA filter to implement the transmitter.
  - Let the sampling frequency of the ADC be 1MHz. Design a DMA filter, use direct conversion to downconvert the receive signal, and recover the transmit symbols.



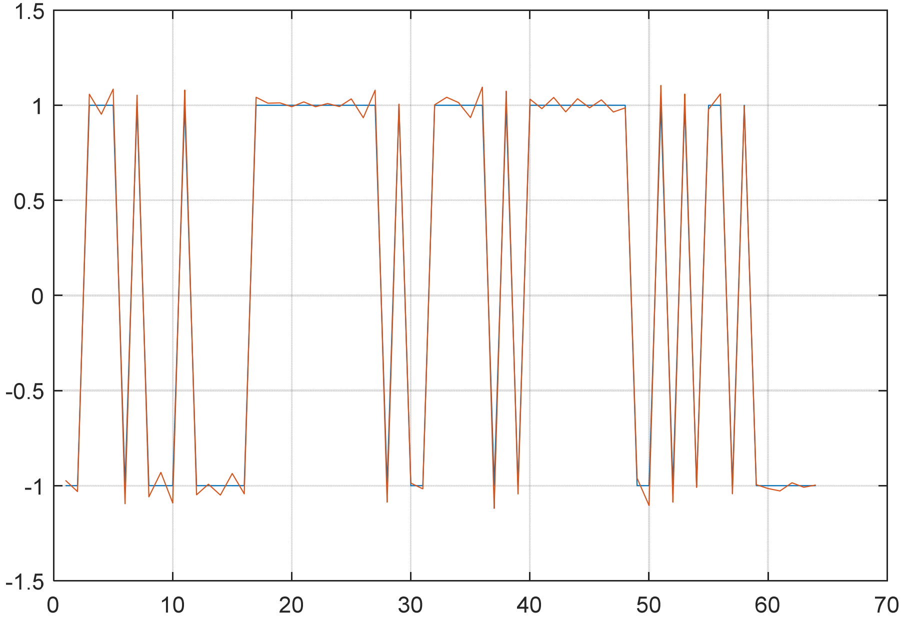
- Result:

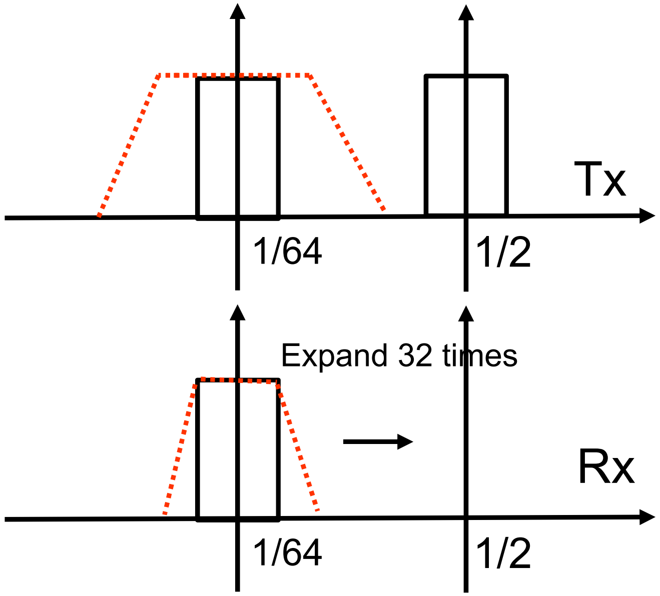


Group delay: transmit filter



Group delay: receive filter





Tx

1/64

1/2

Expand 32 times

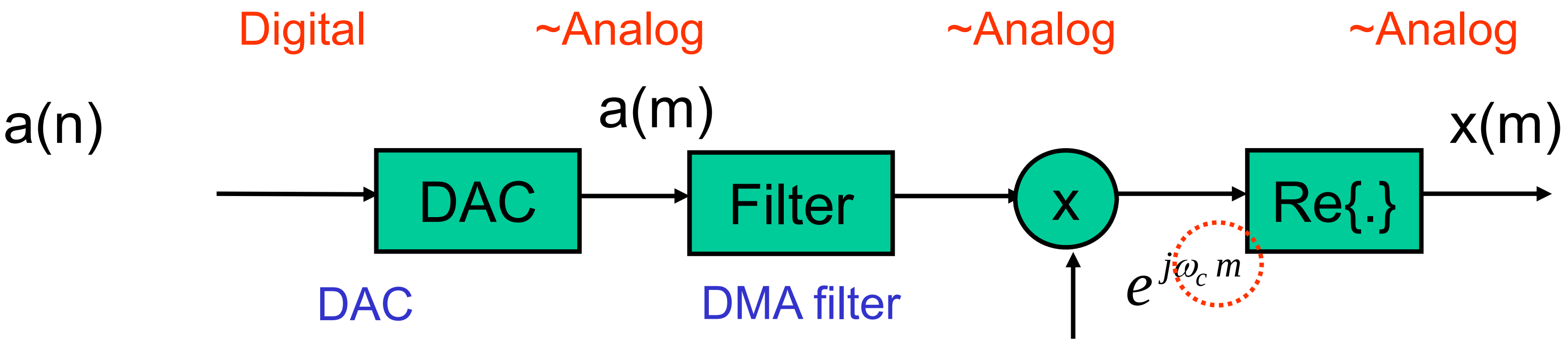
Rx

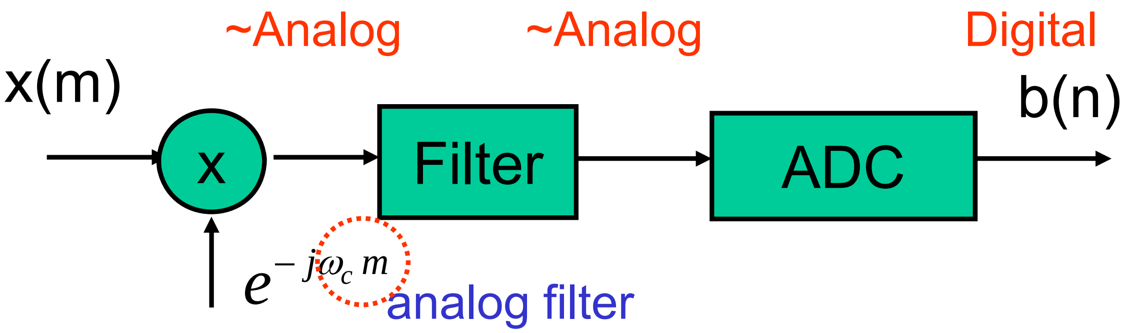
1/64

1/2

* Signal will not aliased. The Only concern is noise

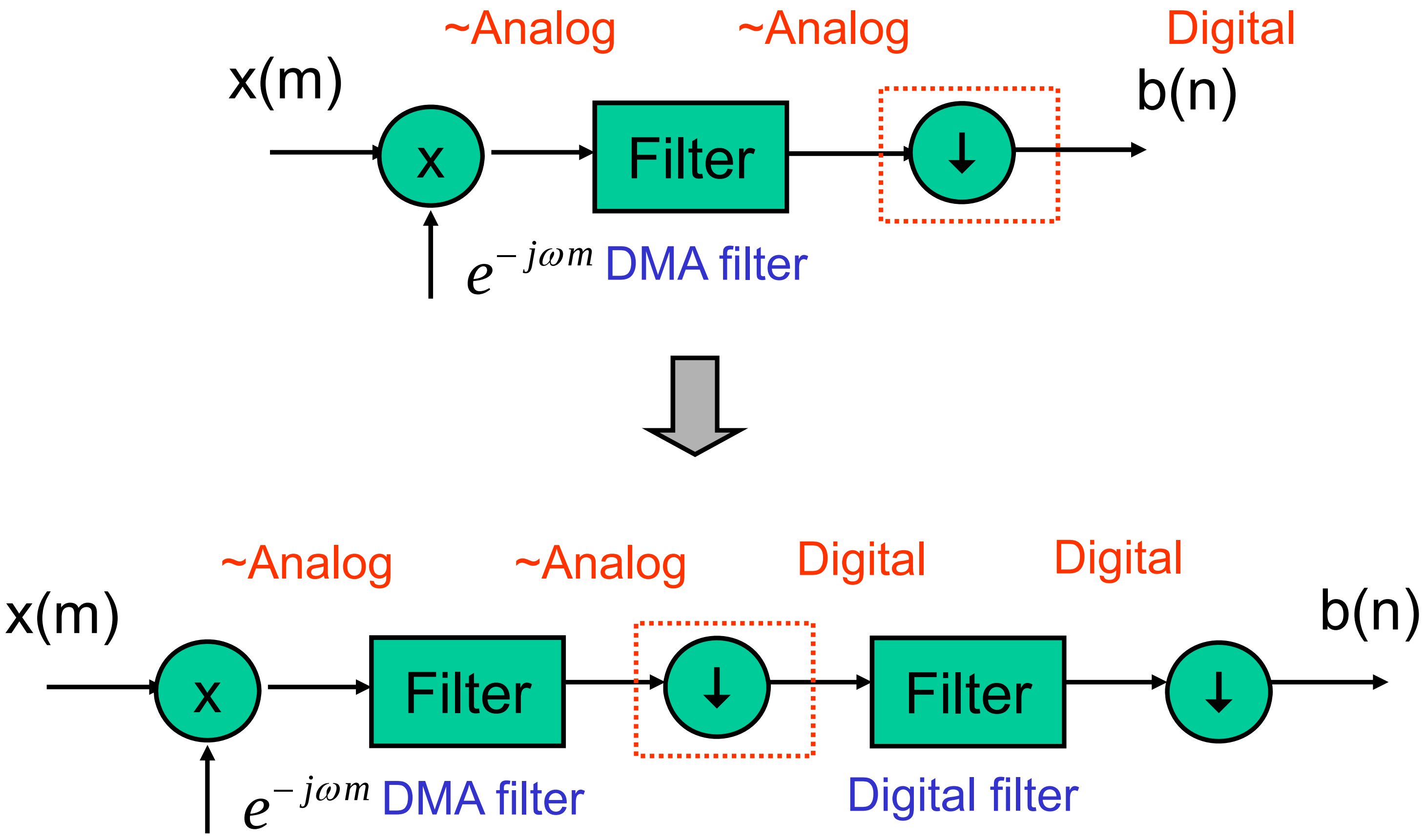
- The modulation system we consider is a coherent system meaning that the phase of the demodulating carrier must be the same as that of transmitter.



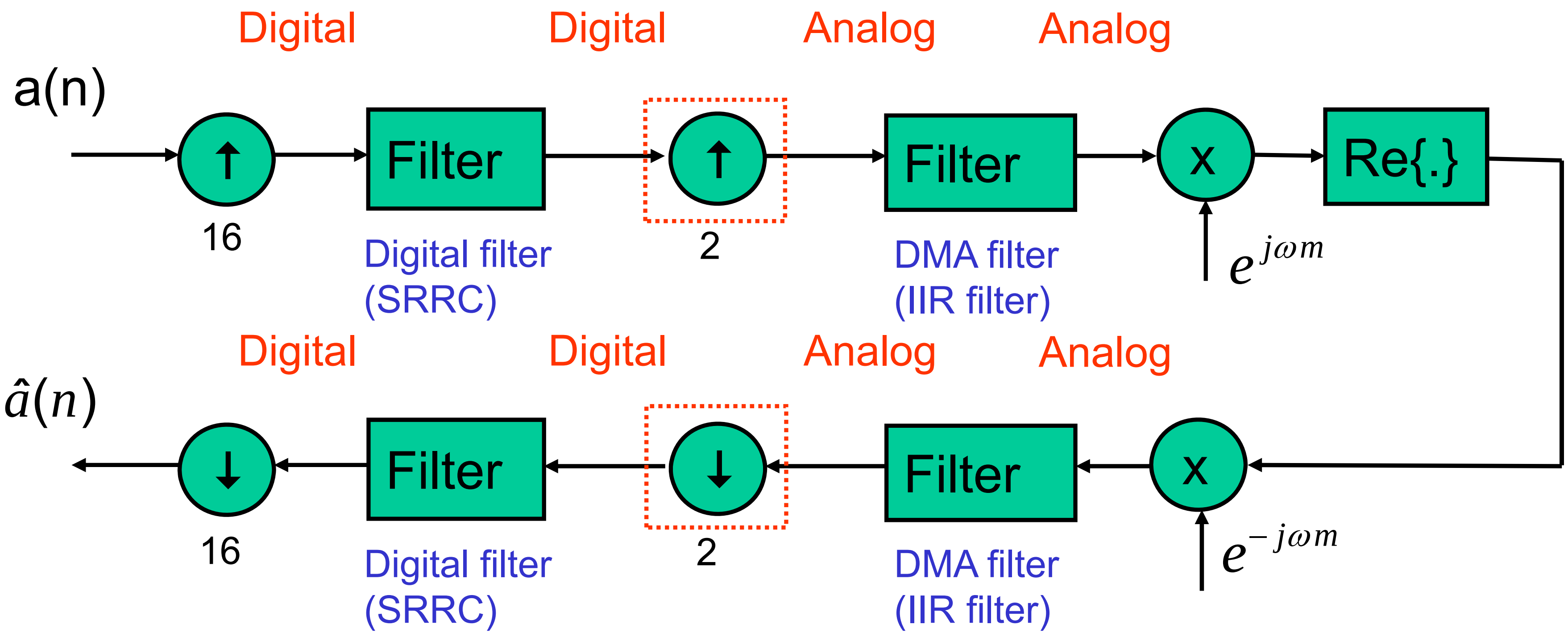


* What will happen if $e^{-j(\omega_c m+\theta)}$ is used for demodulation ?

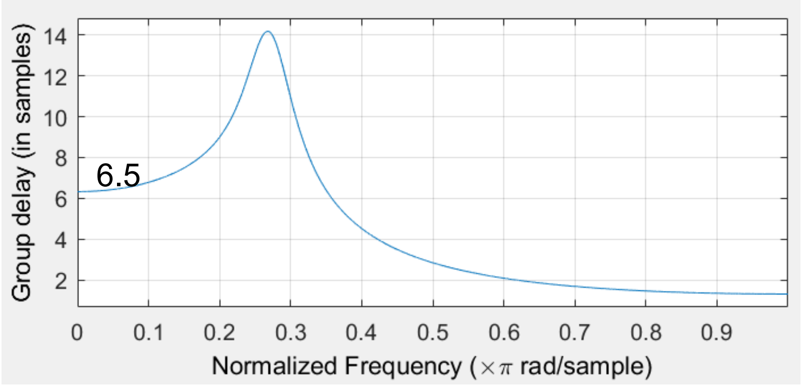
- The LPF at the receiver can also be implemented with a hybrid form.
  - Better control of filter spectrum (good for noise filtering)



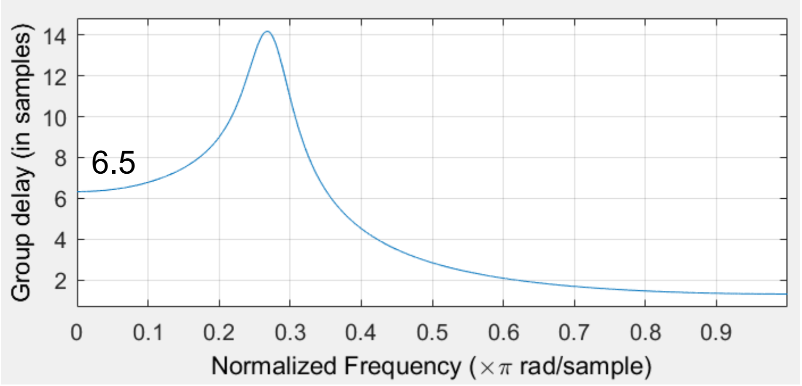
- Practice 2:
  - Use the transmitter in Practice 1.
  - Let the sampling rate of the ADC be 16MHz. Conduct direct conversion (using same filters) to downconvert the receive signal and recover the transmit symbols.



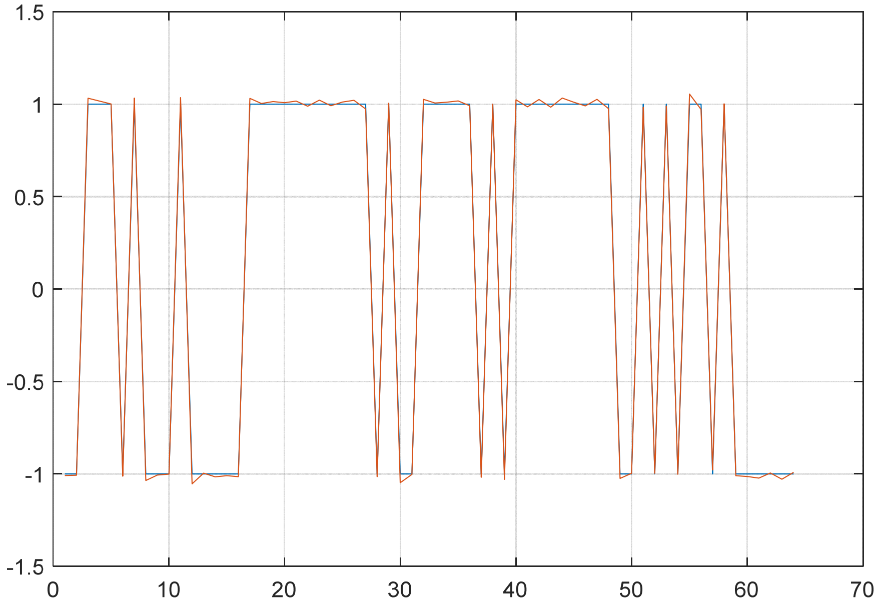
- Result:



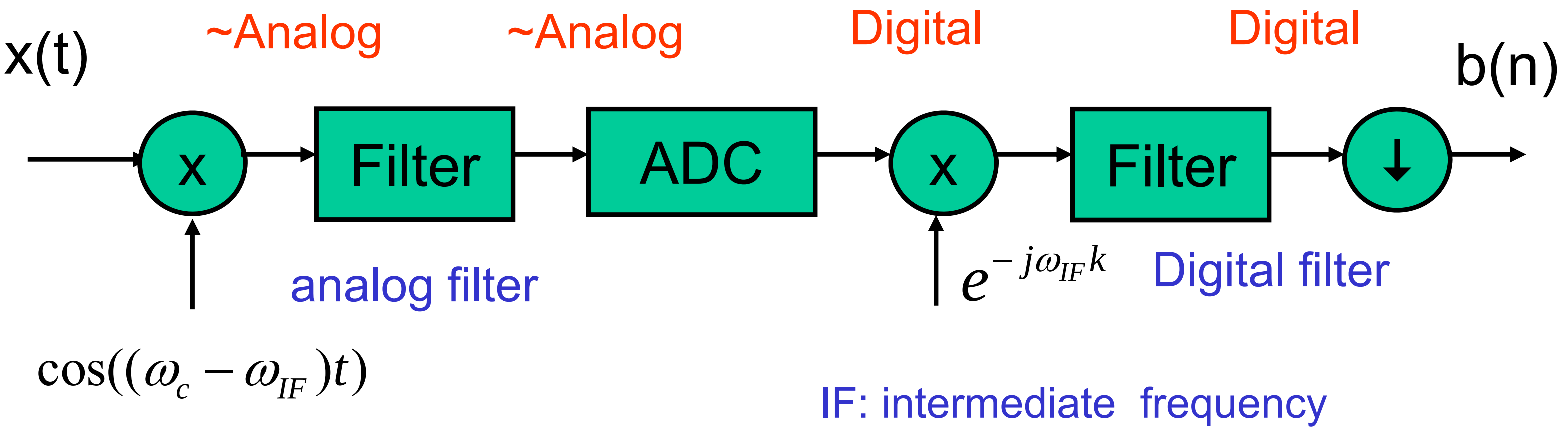
Group delay: transmit filter

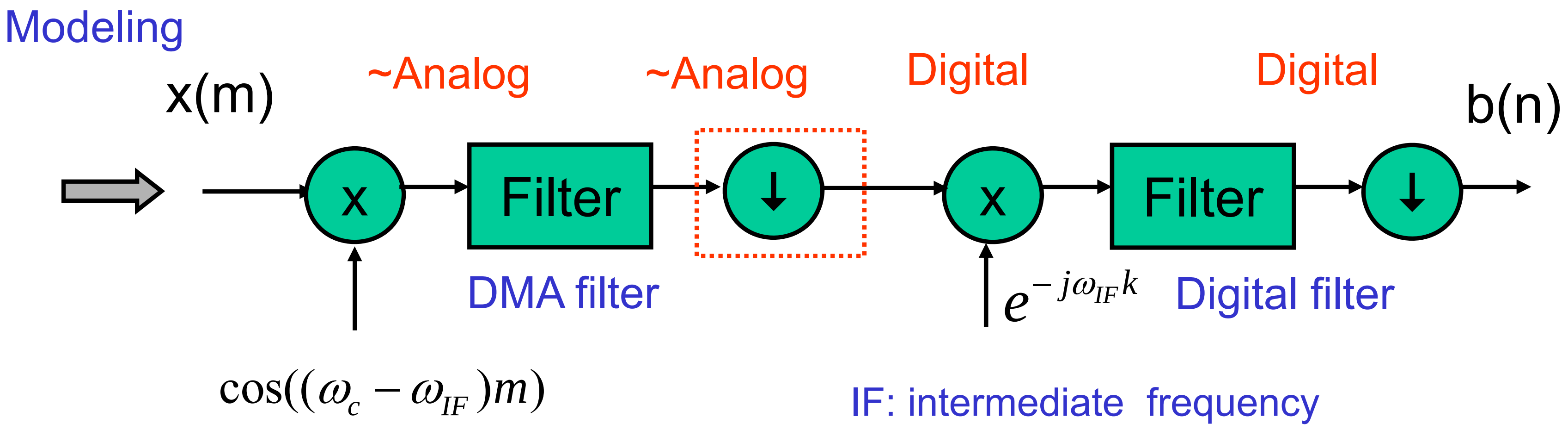


Group delay: receive filter

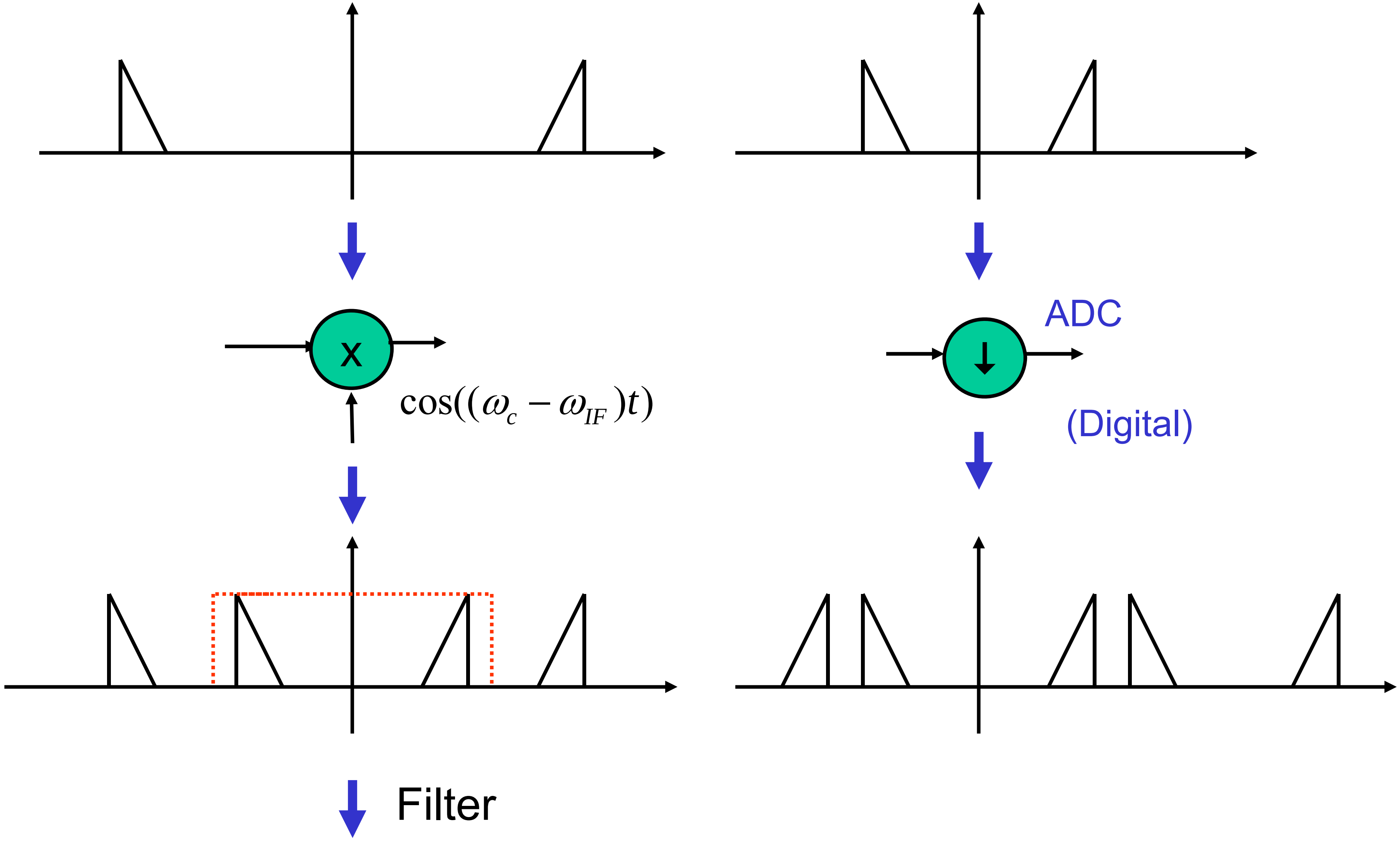


- Intermediate-frequency (IF) demodulation:

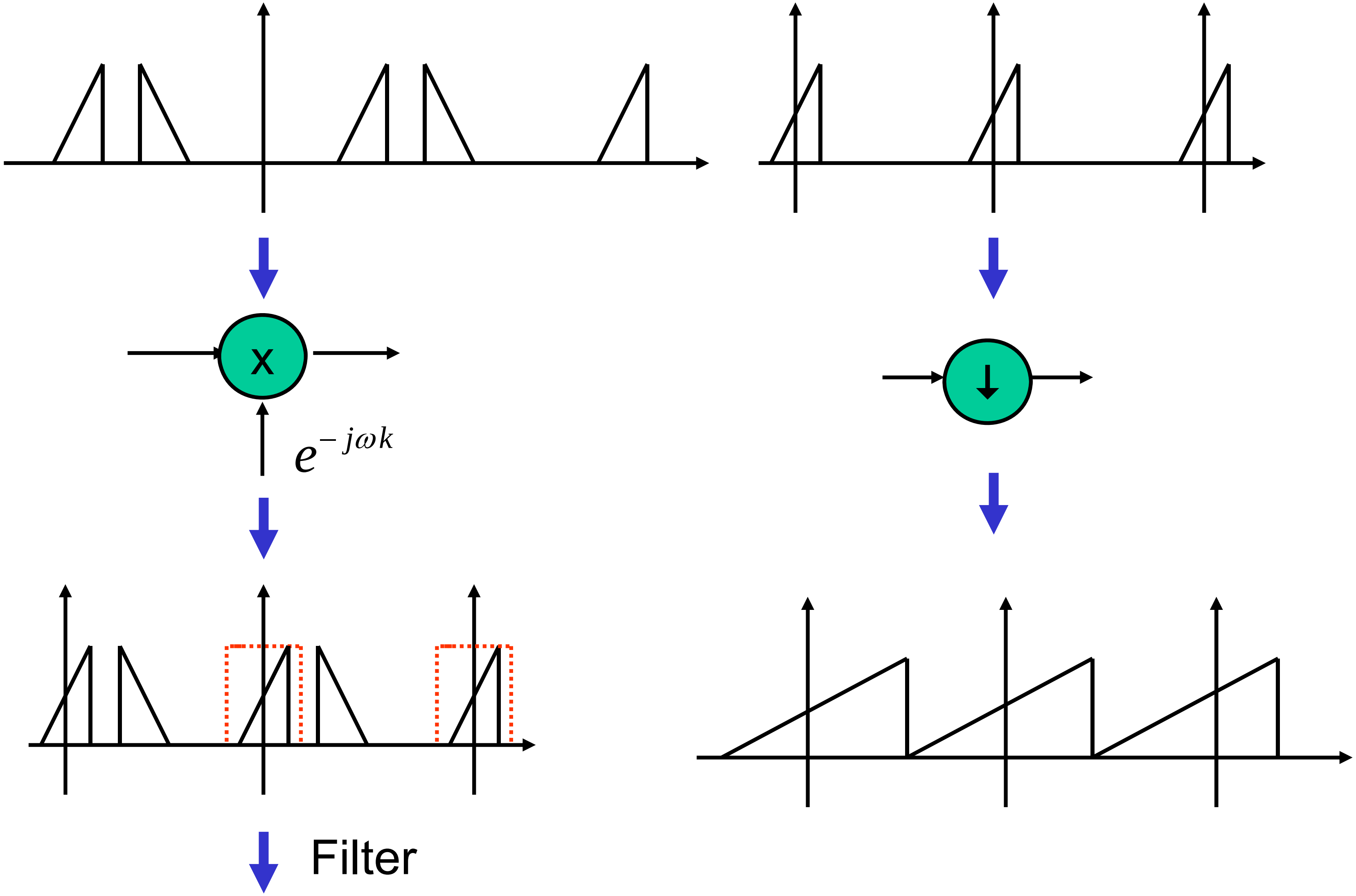




- Spectrum relationship:



- Continue-



x

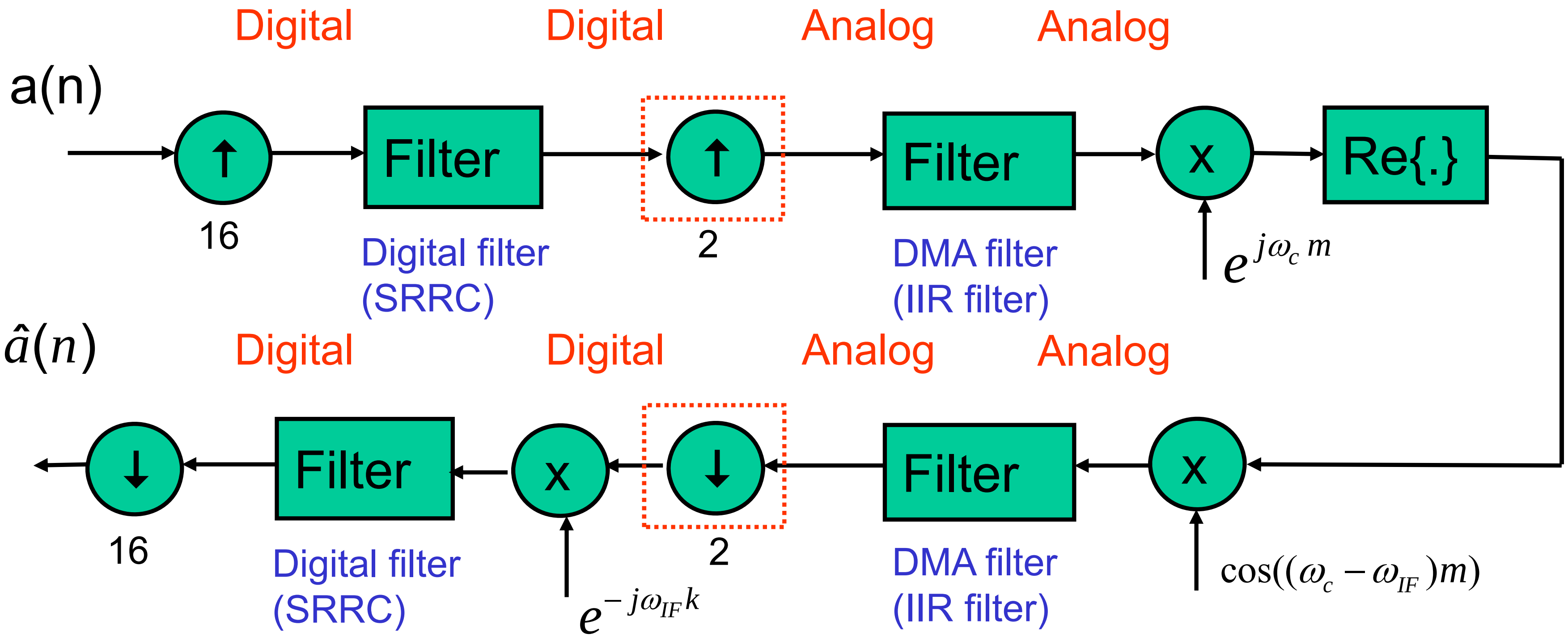
$e^{-j\omega k}$

↓

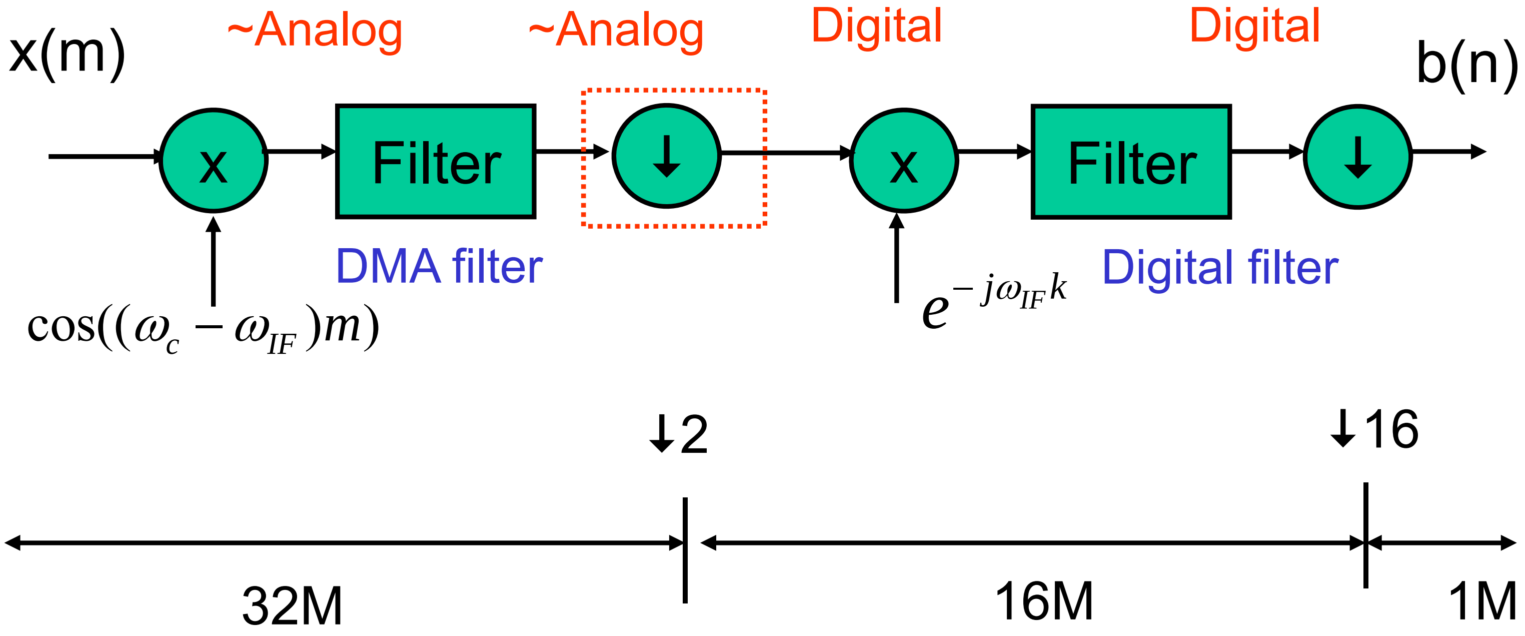
Filter

- What is the advantage of IF demodulation?
  - Only is one ADC required.
  - Avoid IQ imbalance (mismatch) problem.
- Disadvantage:
  - Higher sampling rate
- Note that the IF structure can also be used at the transmitter, i.e. IF modulation.

- Practice 3:
  - Use the transmitter in Practice 1.
  - Let the IF frequency be 4M, the sampling rate of the ADC be 16MHz, that of analog signal be 32 MHz, and the carrier frequency be 8MHz.
  - Design a lowpass DMA filter to conduct the IF demodulation.
  - Check the spectrum of the IF signal.



- Hint:

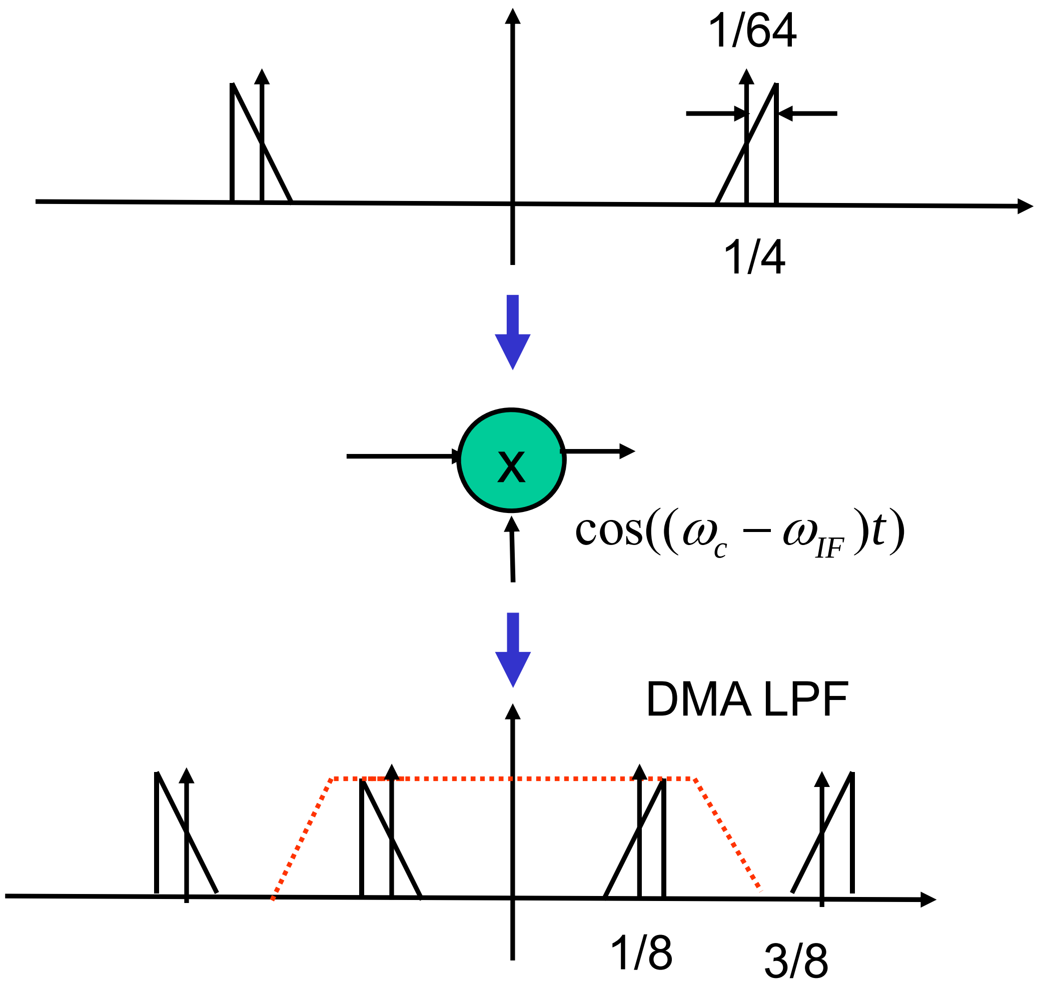


$$\cos(2\pi \frac{\text{8M-4M}}{\text{32M}} m) = \cos(\frac{2\pi m}{8})$$

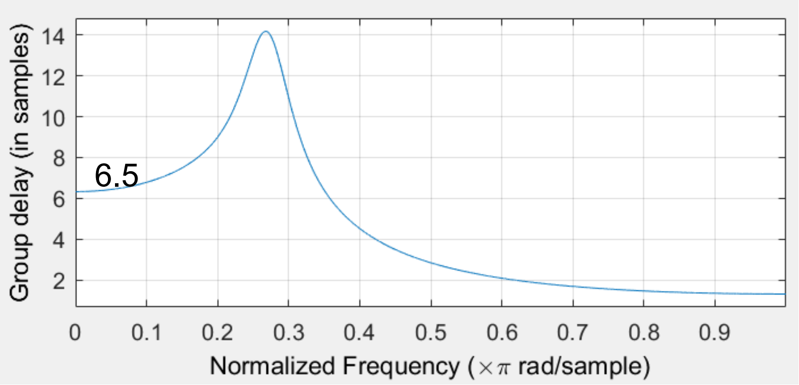
$$e^{-j(2\pi \frac{\text{4M}}{\text{16M}} m)} = e^{-j(\frac{2\pi m}{4})}$$

* Note that there is a delay (phase) problem in exp(-jω_{IF}k)
(due to the receive filter) which will rotate demodulated signal

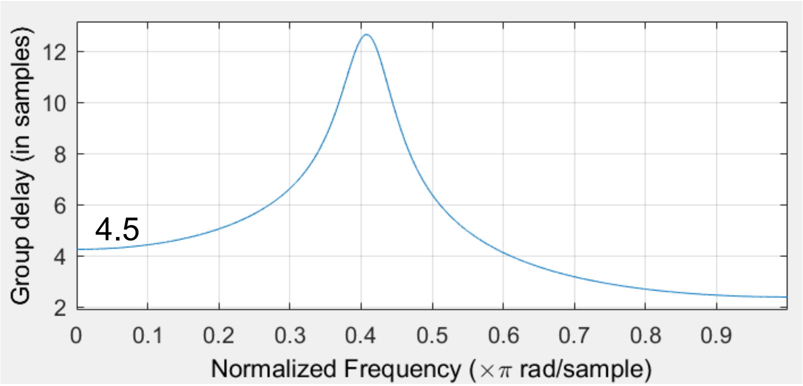
- The spectrum (receive filter):



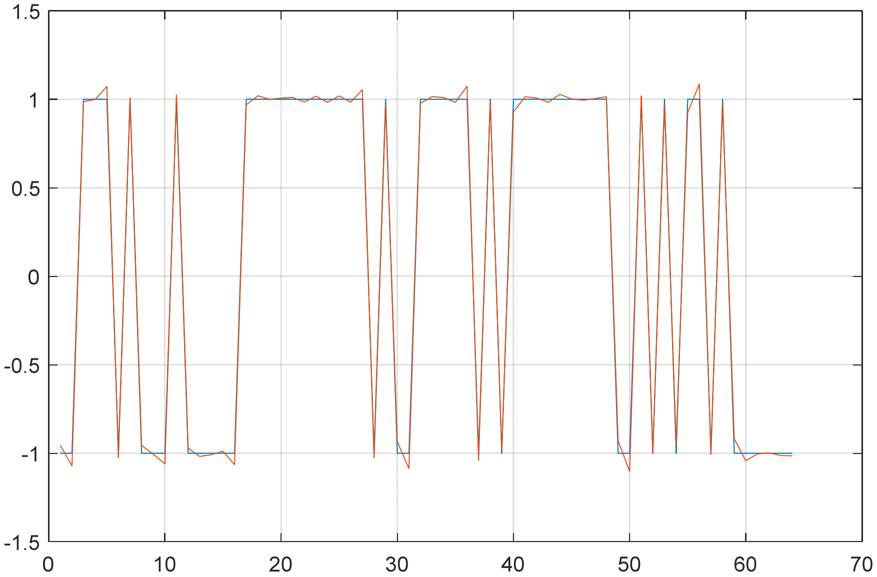
- Result:

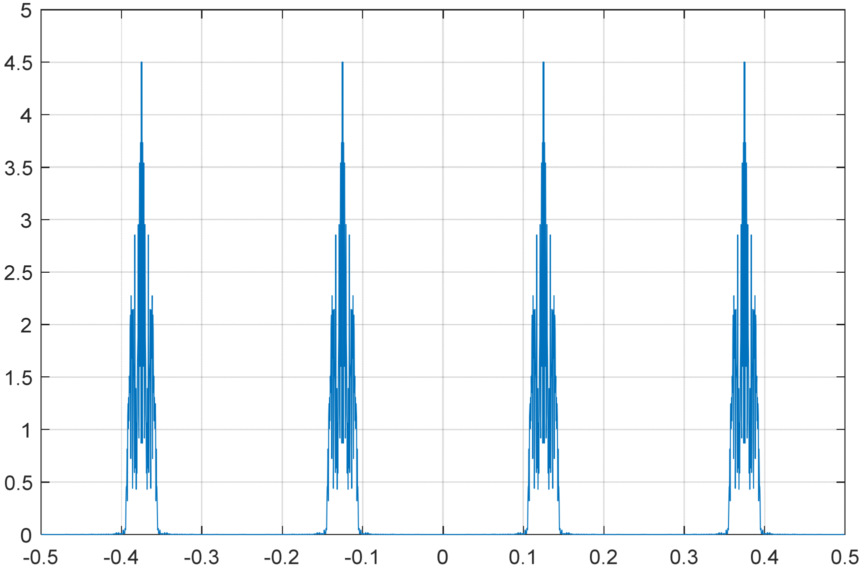


Group delay: transmit filter



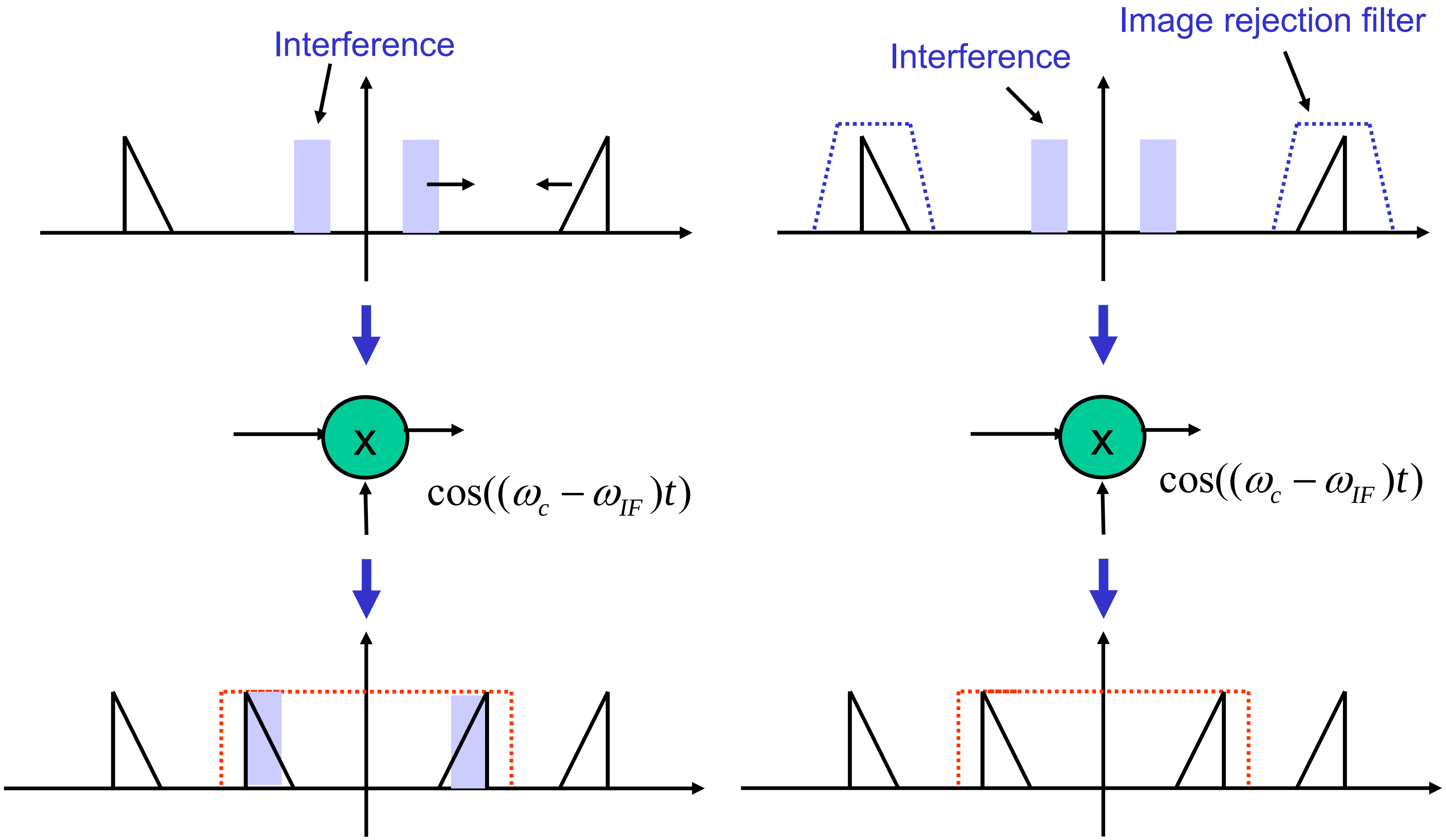
Group delay: receive filter





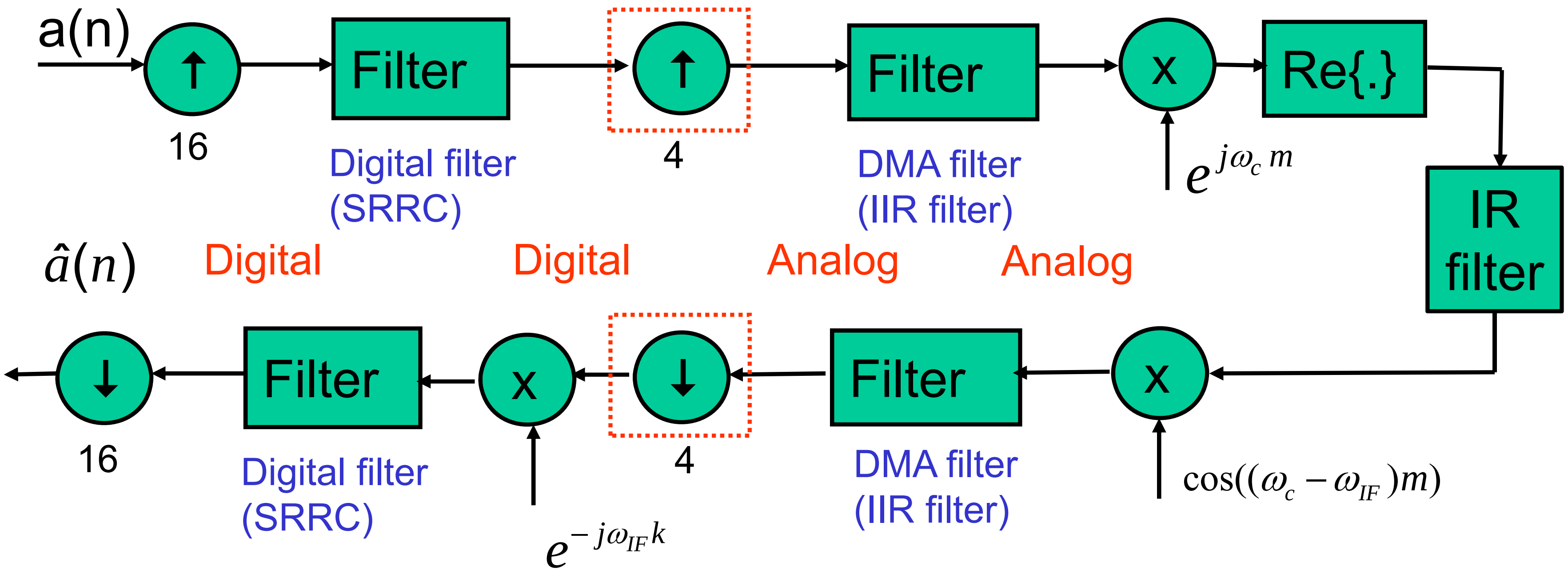
1/8

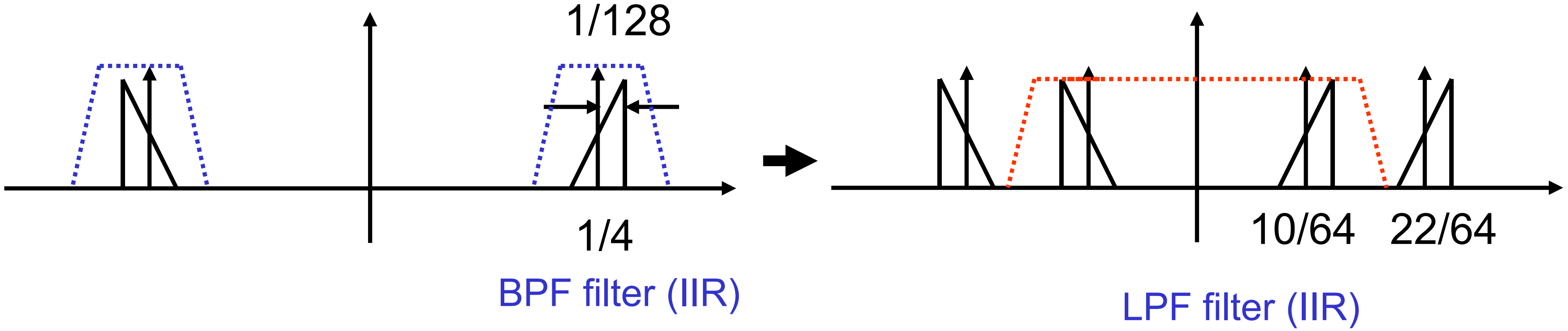
- Image problem for IF demodulation:


Interference
x
$\cos((\omega_c - \omega_{IF})t)$
Image rejection filter
Interference
x
$\cos((\omega_c - \omega_{IF})t)$

- Homework:
  - Let f_c be the frequency of carrier, f_i be that of the image signal, and f_{IF} be that of IF carrier. Calculate f_i.
  - Let f_c=16MHz, the symbol rate be 1MHz, f_{IF}=10MHz, the sampling rate of ADC be 16M, and that of the DMA filter be 64MHz.
  - Redesign the system in S9P3 to upcovert and downconvert the transmit signal, and recover the transmit symbols
  - Design an image rejection filter, and add the filter to the above system.
  - Add a sinusoidal image interference and conduct the simulation again.

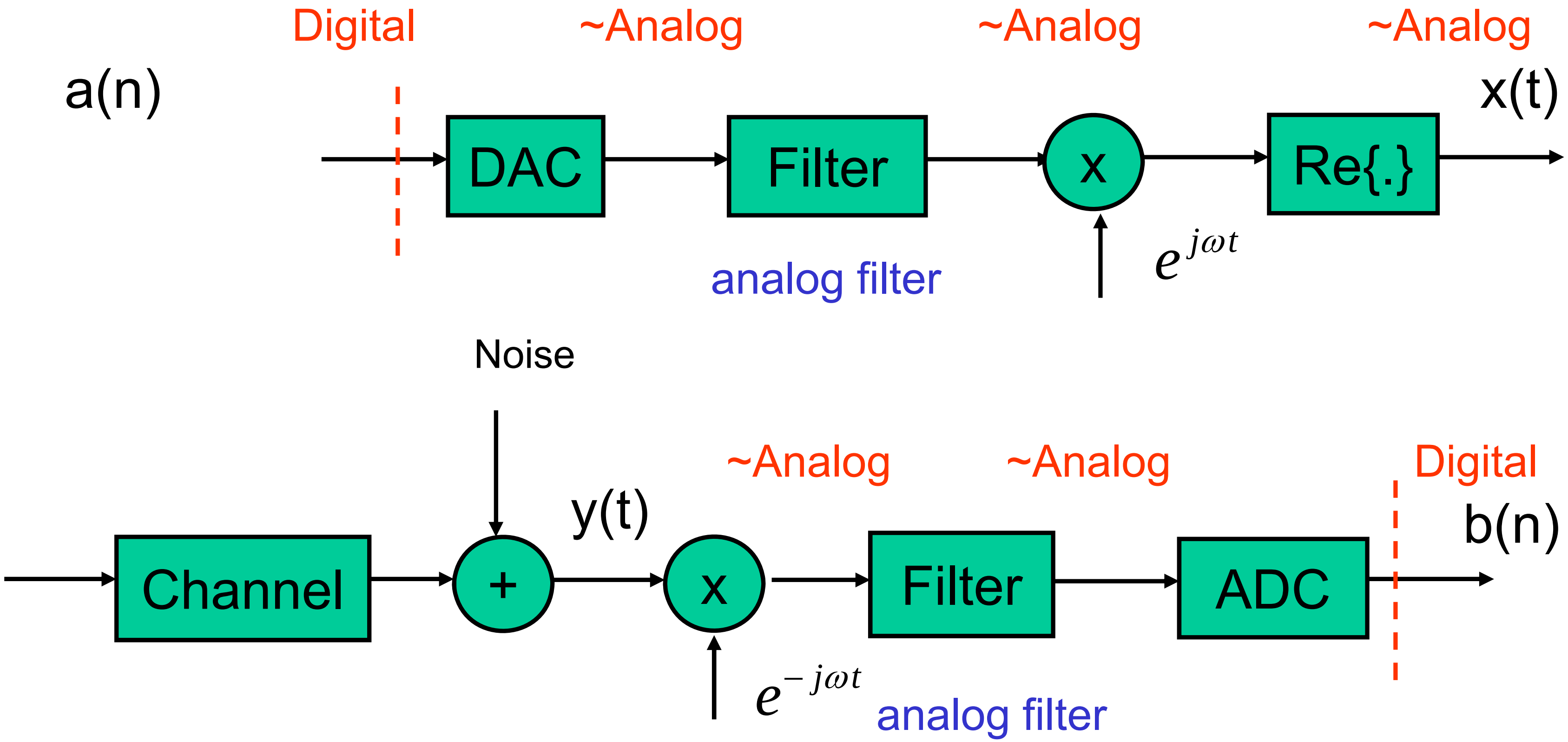
- Transceiver:





Lab. 10 RF Impairments

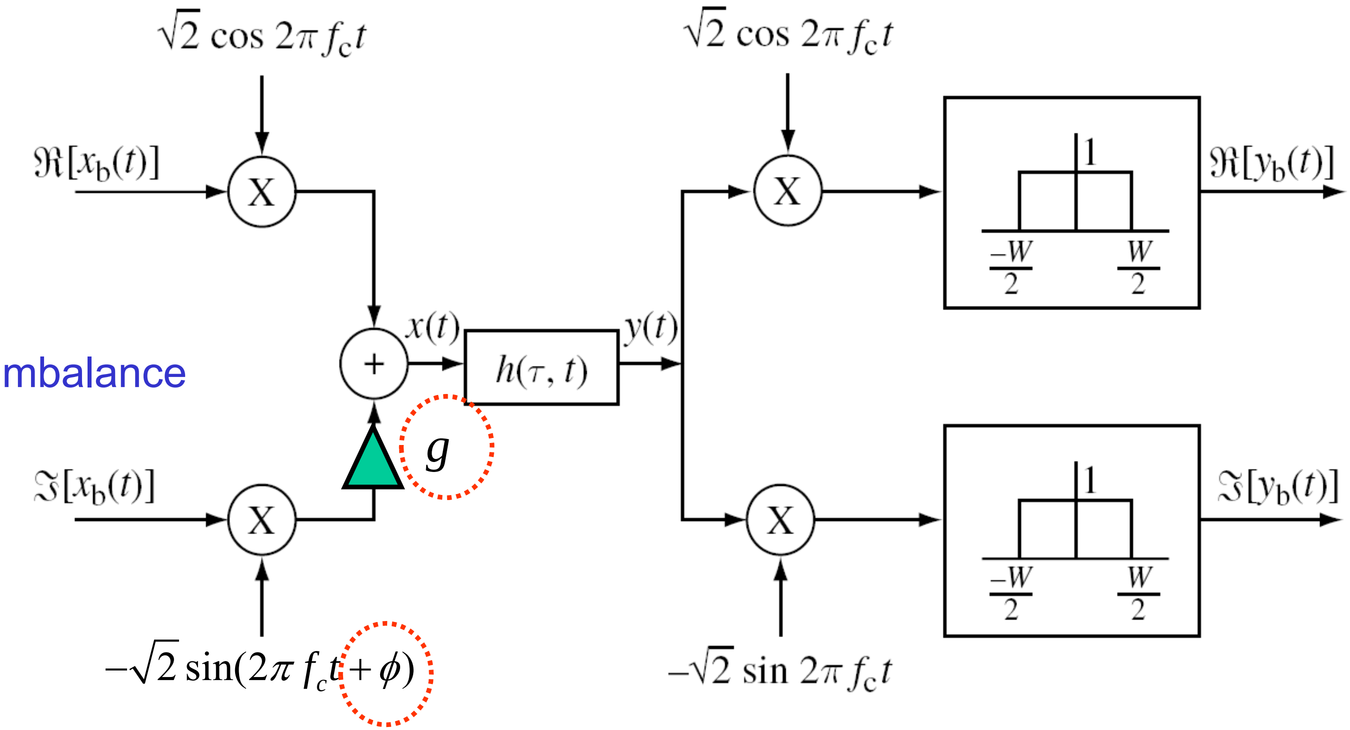
- A digital communication system:



- Impairments:
  - DAC
  - Transmit IQ imbalance
  - Phase noise of the mixer
  - PA nonlinearity
  - Channel effect
  - Noise
  - Receive IQ imbalance
  - DC offset
  - Phase noise of the mixer
  - ADC
  - Carrier frequency offset

- DAC
  - Quantize the output signal (reduce the number of bits)
  - Convert digital signal to analog signal
- IQ-imbalance
  - Amplitude imbalance
  - Phase imbalance

Transmit imbalance

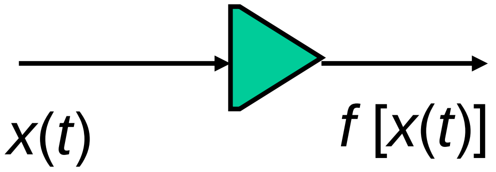


- Phase noise:

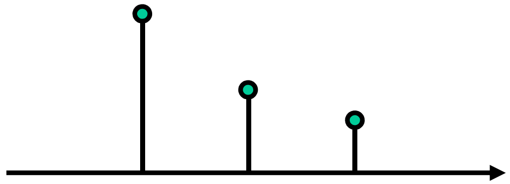
Lowpass random signal

$$\text{carrier}: \cos(2\pi ft + \rho)$$

- PA nonlinearity:

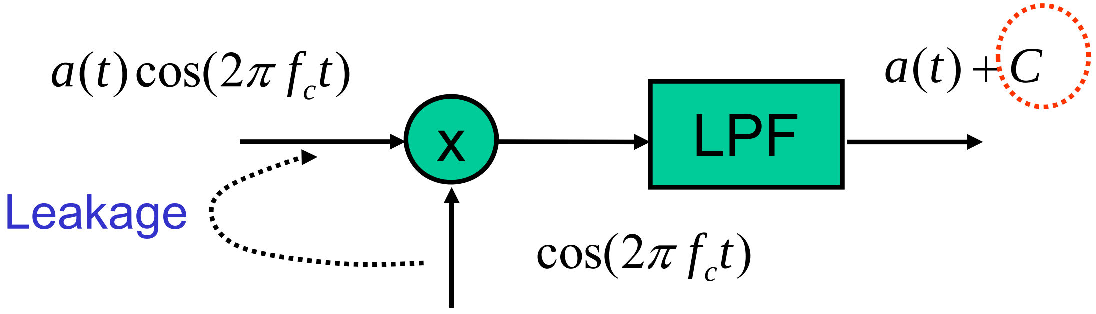


- Channel effect:

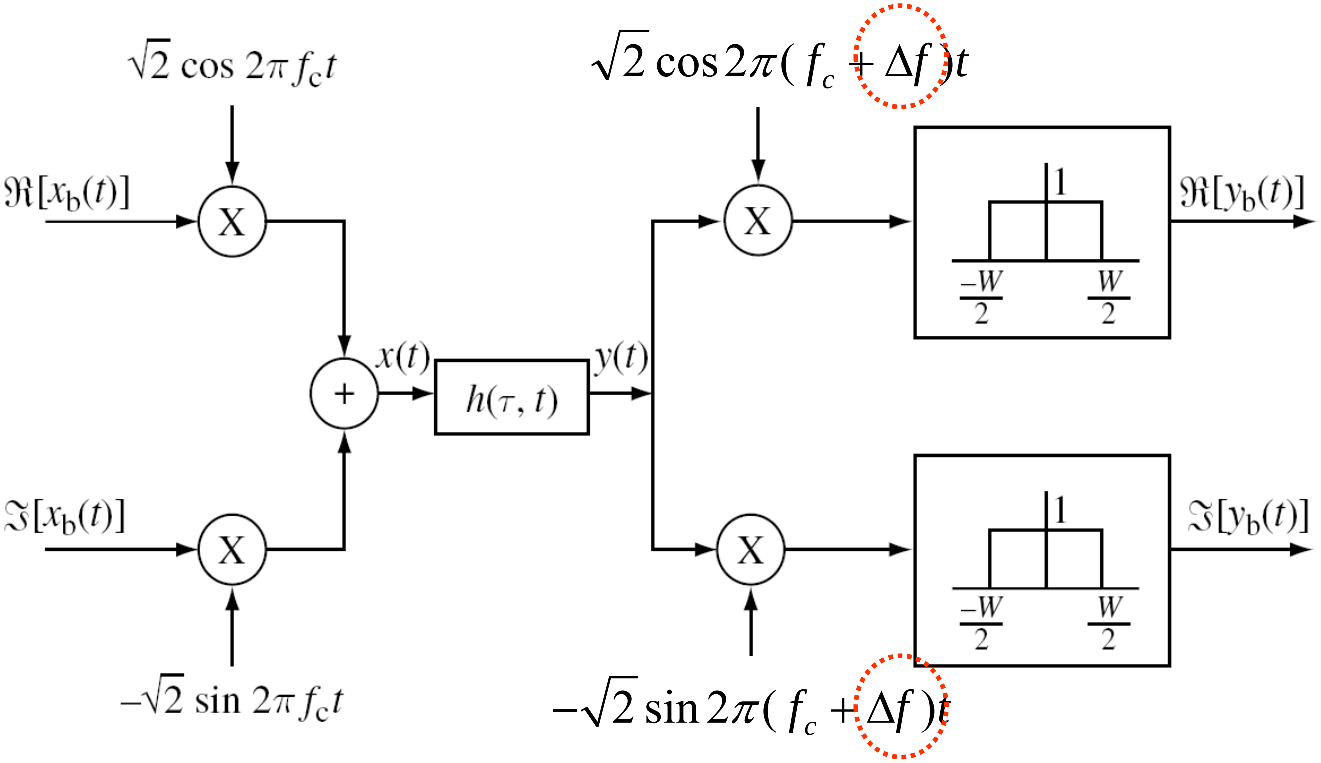


Multi-path

- DC offset:



- ADC:
  - – Convert continuous-time signal to discrete-time
  - – Quantize the discrete-time signal
- Carrier frequency offset (CFO):



- Transmit IQ-imbalance:

$$x(t) = a_I(t)\cos(\omega t) - a_Q(t)\boxed{g}\sin(\omega t \boxed{+\phi})$$

$$= a_I(t)\frac{e^{j\omega t}+e^{-j\omega t}}{2} + jga_Q(t)\frac{e^{j(\omega t+\phi)}-e^{-j(\omega t+\phi)}}{2}$$

(Δ)

- Demodulated signal (receiver does not have imbalance):

$$LPF\left\{2x(t)e^{-j\omega t}\right\} = a_I(t) + jga_Q(t)e^{j\phi}$$

* From (Δ)

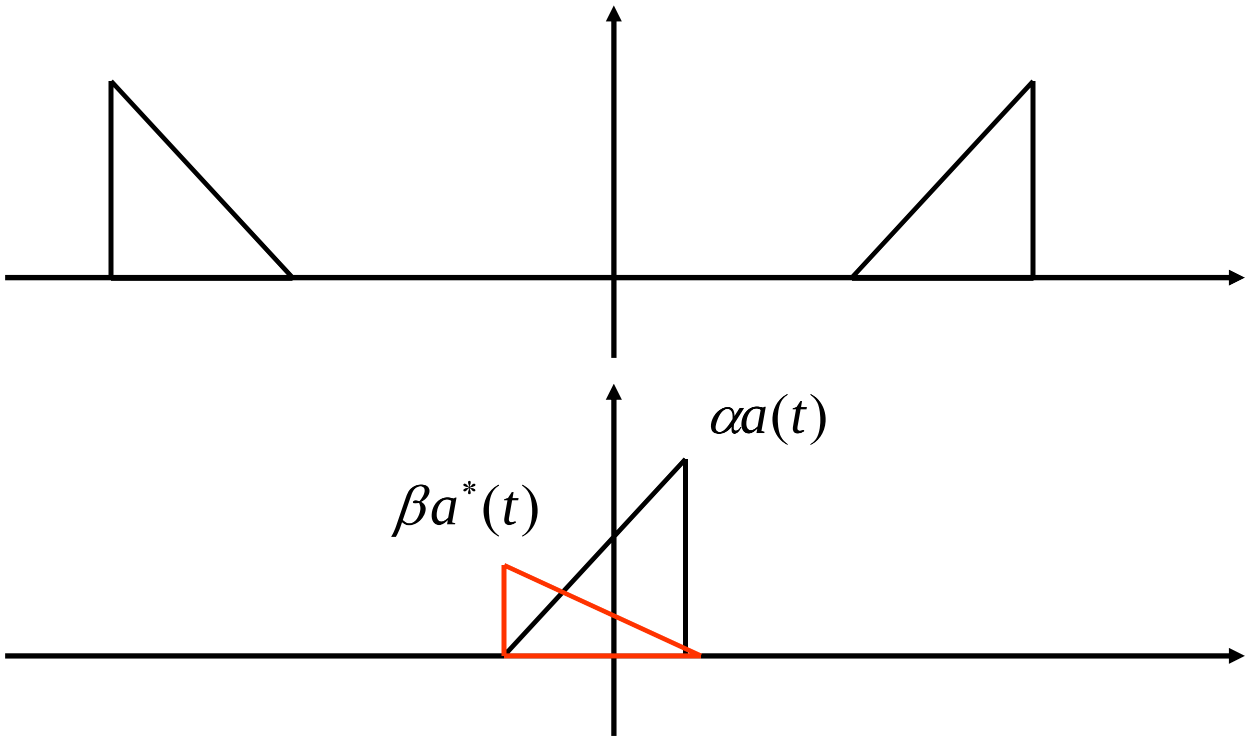
$$= \alpha\left[a_I(t) + ja_Q(t)\right] + \beta\left[a_I(t) - ja_Q(t)\right]$$

$$= \alpha a(t) + \beta a^*(t)$$

* $a(t) = a_I(t) + ja_Q(t)$

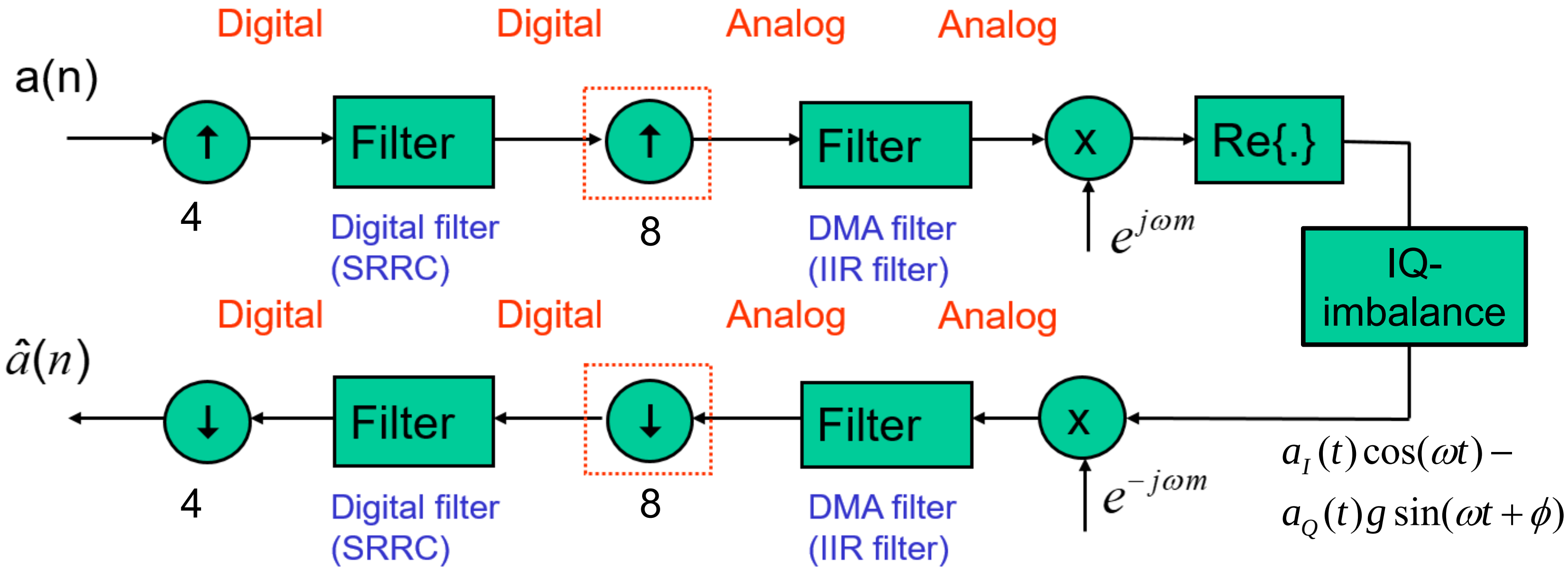
$$\alpha = \frac{1}{2}\left(1 + ge^{j\phi}\right), \quad \beta = \frac{1}{2}\left(1 - ge^{j\phi}\right)$$

- The spectrum (downconvert to baseband):

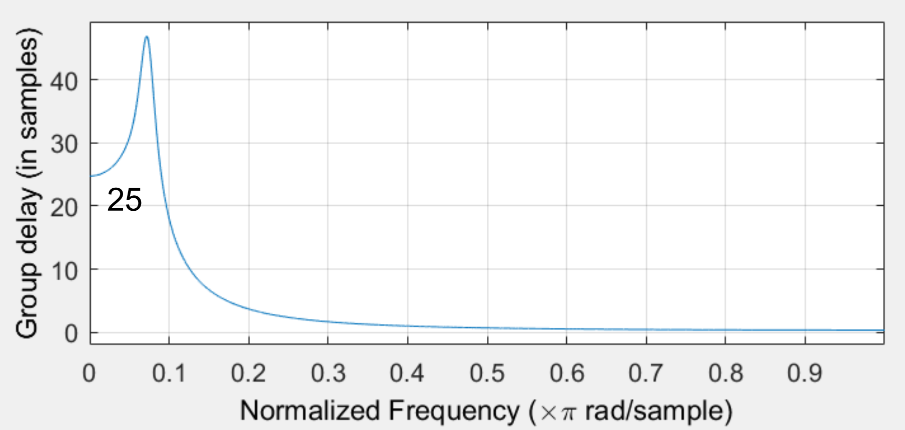


- IQ-imbalance results in a self interference. Note that high frequency components will interfere low frequency components and vice versa.

- Practice 1:
  - Given a complex signal (S10P1.mat) for transmission, use the architecture in S9P2 to simulate a system with transmit IQ-imbalance and observe its effect (redesign IIR filters).
    - Symbol rate: 1MHz, carrier: 8MHz
    - Gain imbalance: 1.2 / Phase imbalance: 15°
  - Compare the result with the one generated with $\alpha a(t) + \beta a^*(t)$

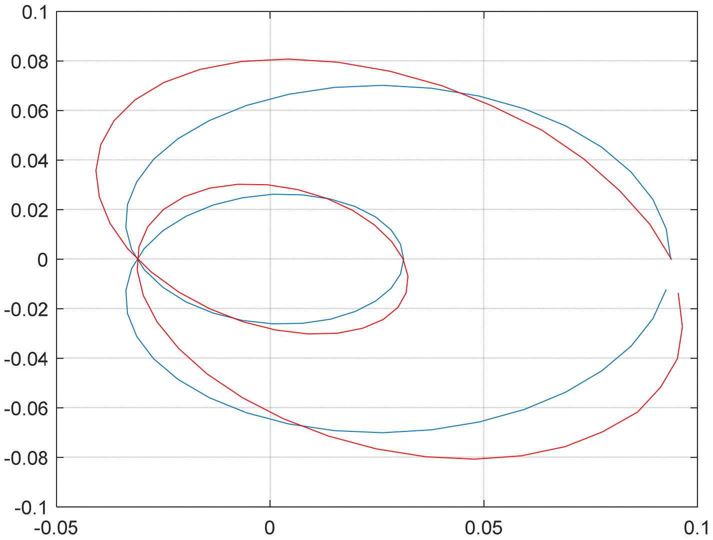


- Result:

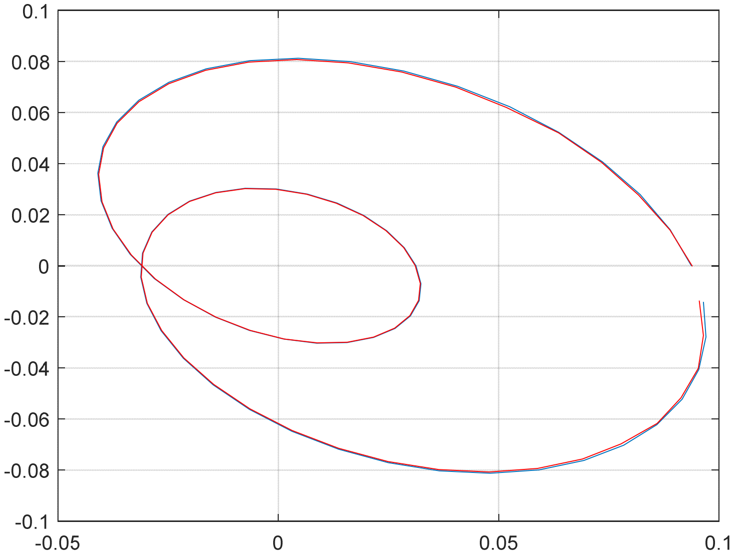


Delay: 2x25

Group delay: transmit filter



Transmitted and received signal



Compare with formulae calculated

- The IQ-imbalnced transmit signal:

$$
\begin{aligned}
x(t) &= a_I(t)\cos(\omega t) - a_Q(t) g \sin(\omega t + \phi) \\
&= [a_I(t) - g \sin\phi a_Q(t)]\cos(\omega t) - g\cos\phi a_Q(t)\sin(\omega t) \\
&= a_I^E(t)\cos(\omega t) - a_Q^E(t)\sin(\omega t)
\end{aligned}
$$

where

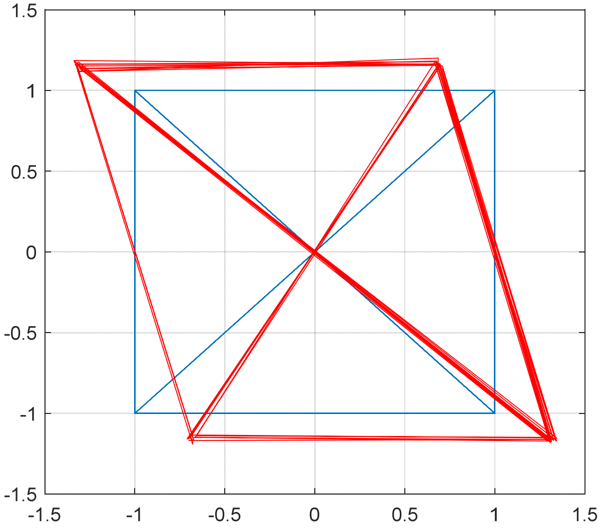
$$
\begin{bmatrix} a_I^E(t) \\ a_Q^E(t) \end{bmatrix} = \begin{bmatrix} 1 & -g\sin\phi \\ 0 & g\cos\phi \end{bmatrix} \begin{bmatrix} a_I(t) \\ a_Q(t) \end{bmatrix} \quad \Rightarrow \quad \mathbf{a}^E(t) = \mathbf{H}\mathbf{a}(t)
$$

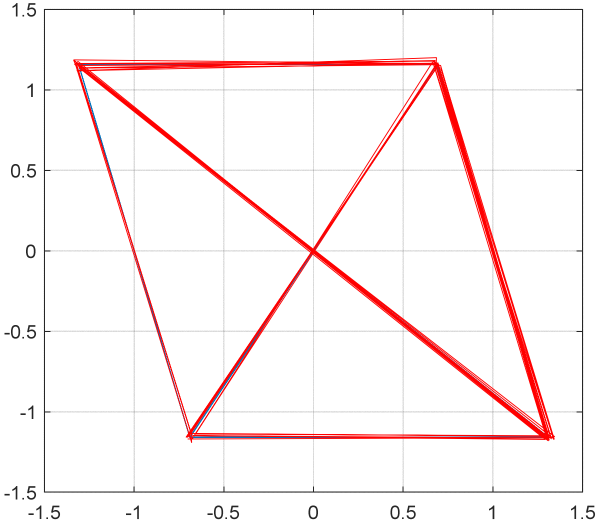
- Compensation of transmit IQ imbalance:
  - Use a signal b(t) for transmission

$$
\mathbf{b}(t) = \begin{bmatrix} b_I(t) \\ b_Q(t) \end{bmatrix} = \mathbf{H}^{-1}\mathbf{a}(t)
$$

- Practice 2:
  - Given an QPSK signal (S10P2.mat), redo Practice 1
  - Conduct the IQ-imbalance compensation in the transmitter and observe the demodulated result

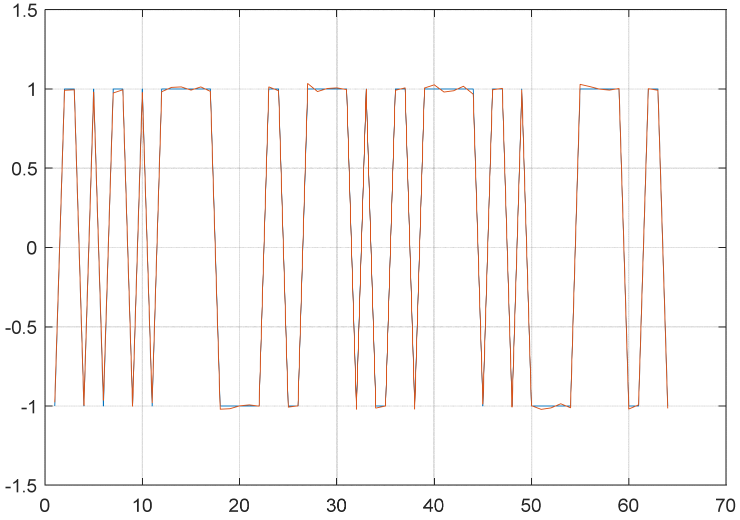
- Result:



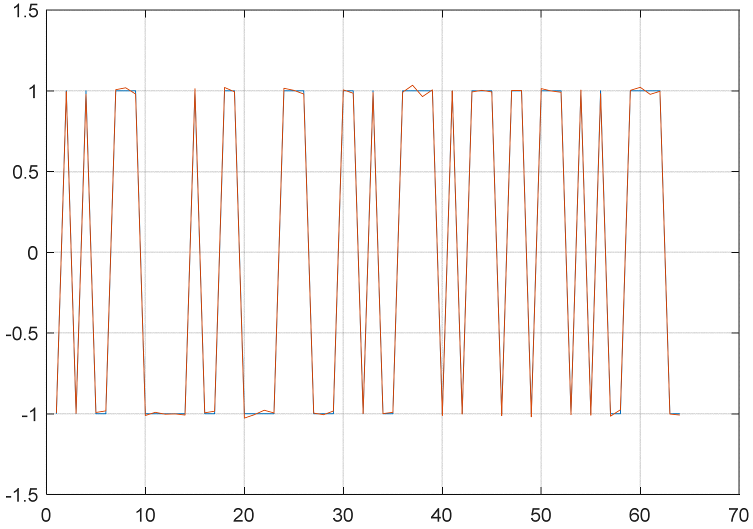


Compare with formulae calculated

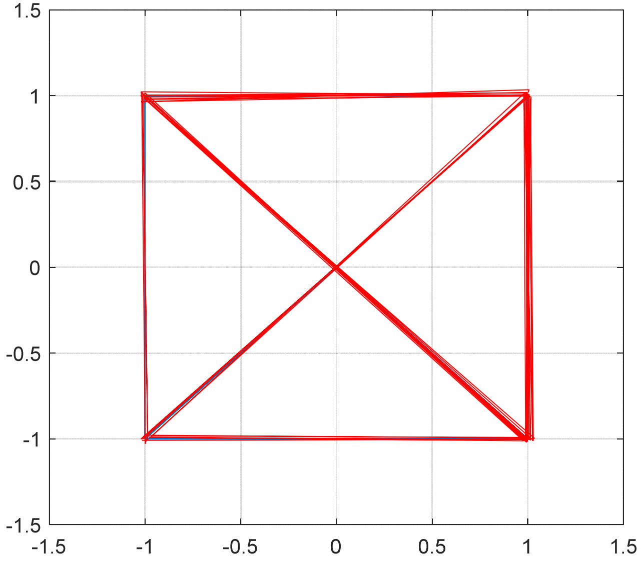
- Result:



Real



Imaginary

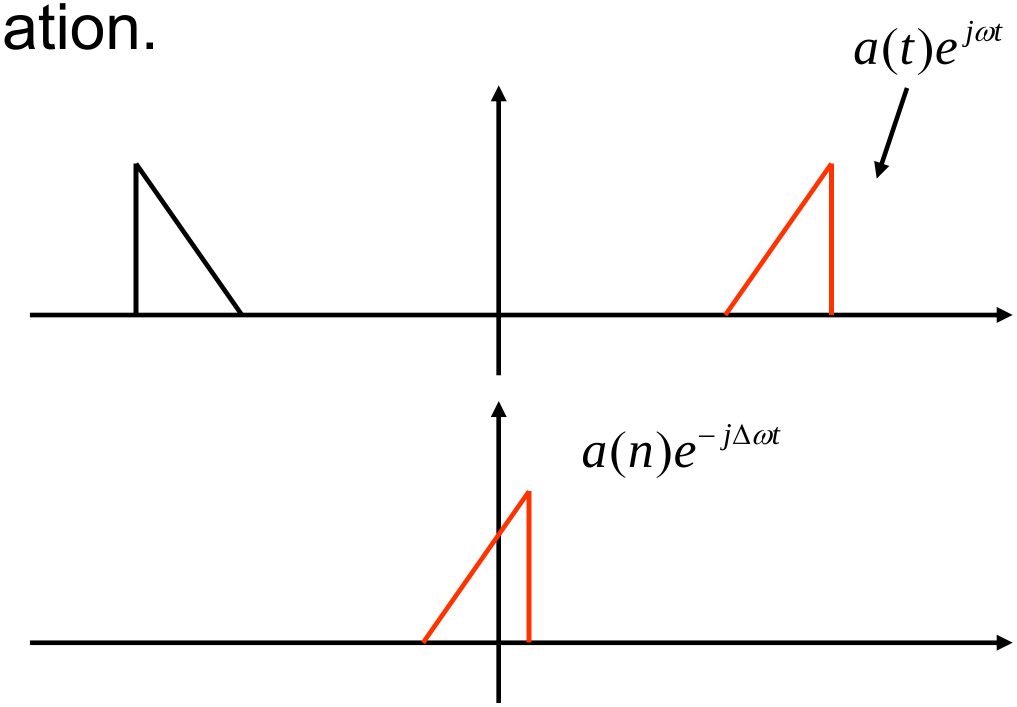


Compensated

- Carrier frequency offset (CFO):

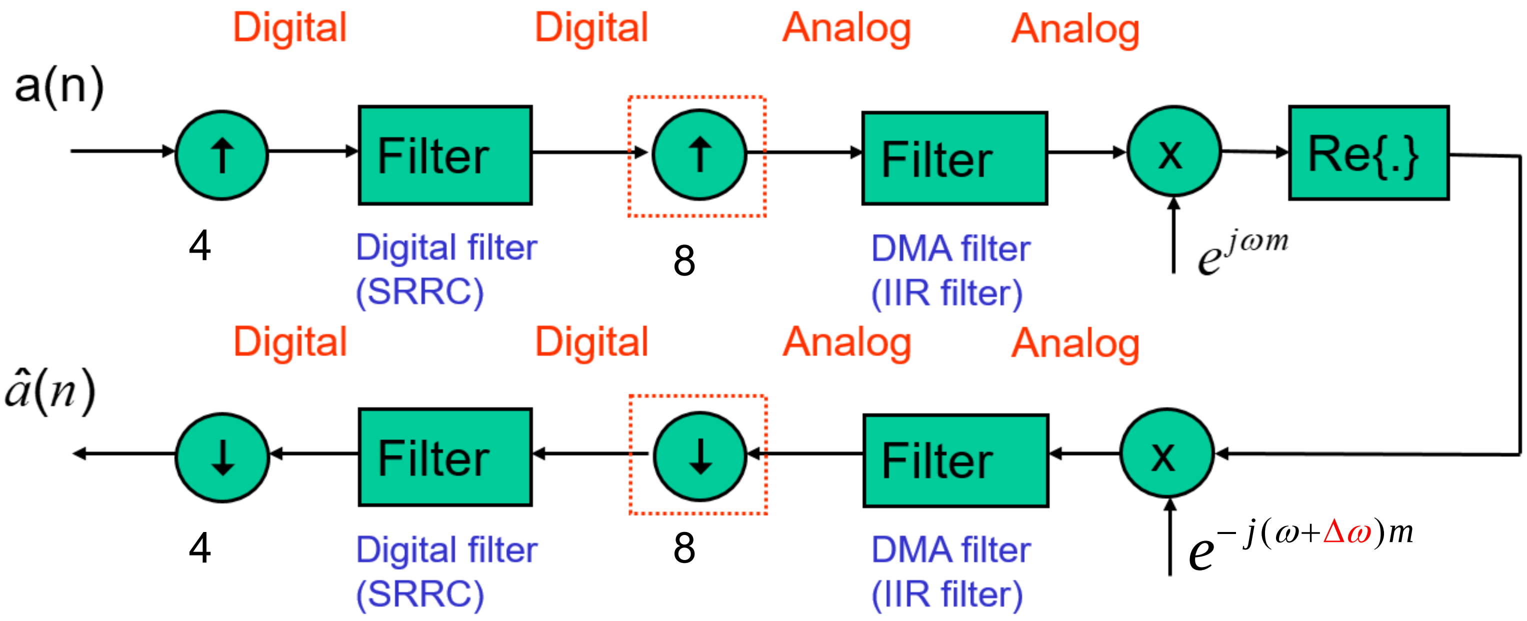
$$a(t)e^{j\omega t}e^{-j(\omega+\Delta\omega)t}=a(t)e^{-j\Delta\omega t}$$

- The CFO results in a constant-rate rotation on the signal constellation.

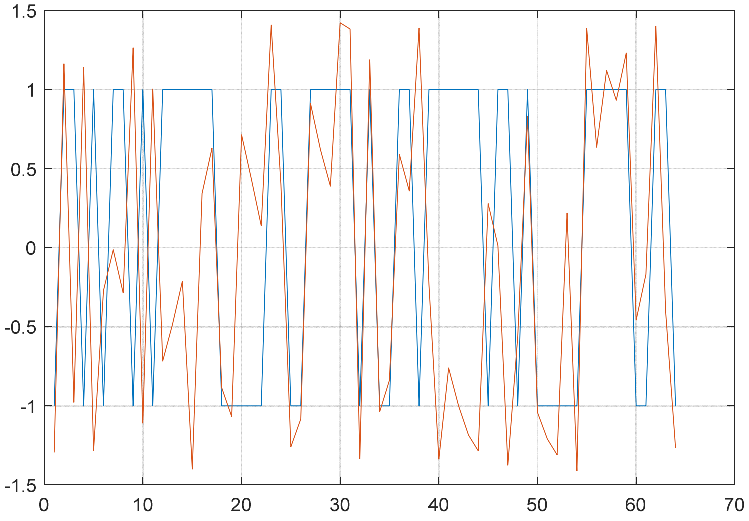


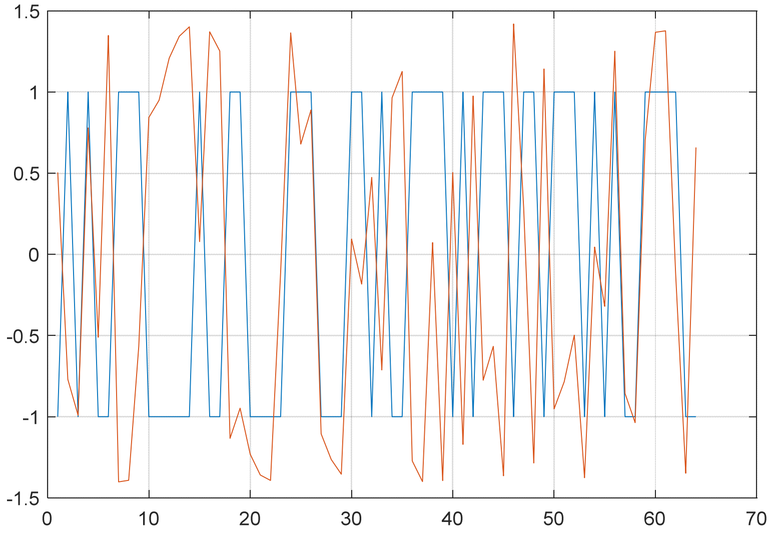
- CFO can be easily compensated by multiplying the received signal by $e^{j\Delta\omega t}$.

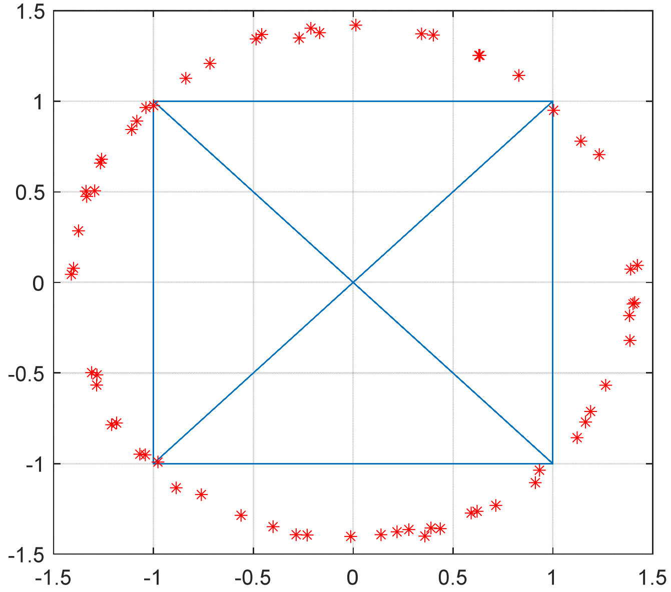
- Practice 3:
  - Given a complex signal (S10P2.mat) for transmission, use the system in Practice 1 to simulate a system with CFO
    - $\Delta\omega = 0.01$
  - Observe the constellation rotation effect.



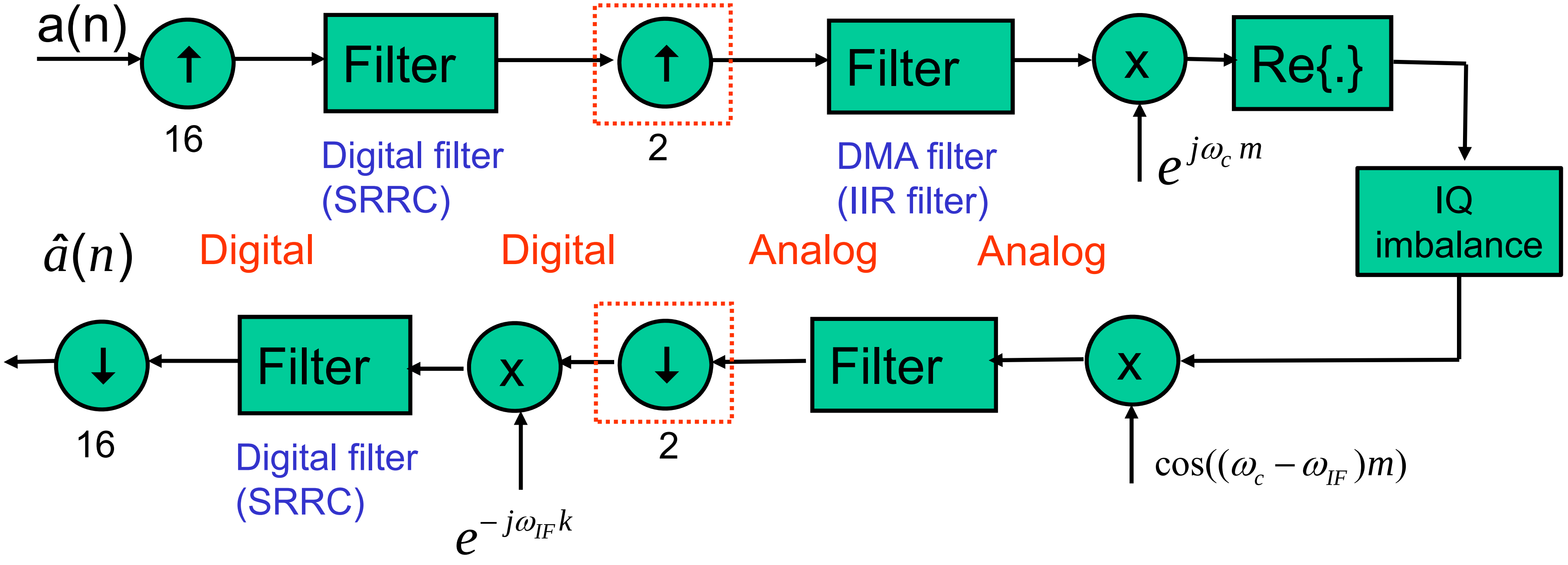
- Result:


1.5
1
0.5
0
-0.5
-1
-1.5
0
10
20
30
40
50
60
70


1.5
1
0.5
0
-0.5
-1
-1.5
0
10
20
30
40
50
60
70


1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5

- Homework:
  - Derive the effect of the transmit IQ-imbalance for the system with IF demodulation.
  - Use the system (filters, transmit signal, parameters etc.) in S9P3, simulate the IF-demodulated system with transmit IQ-imbalance (use QPSK signal).



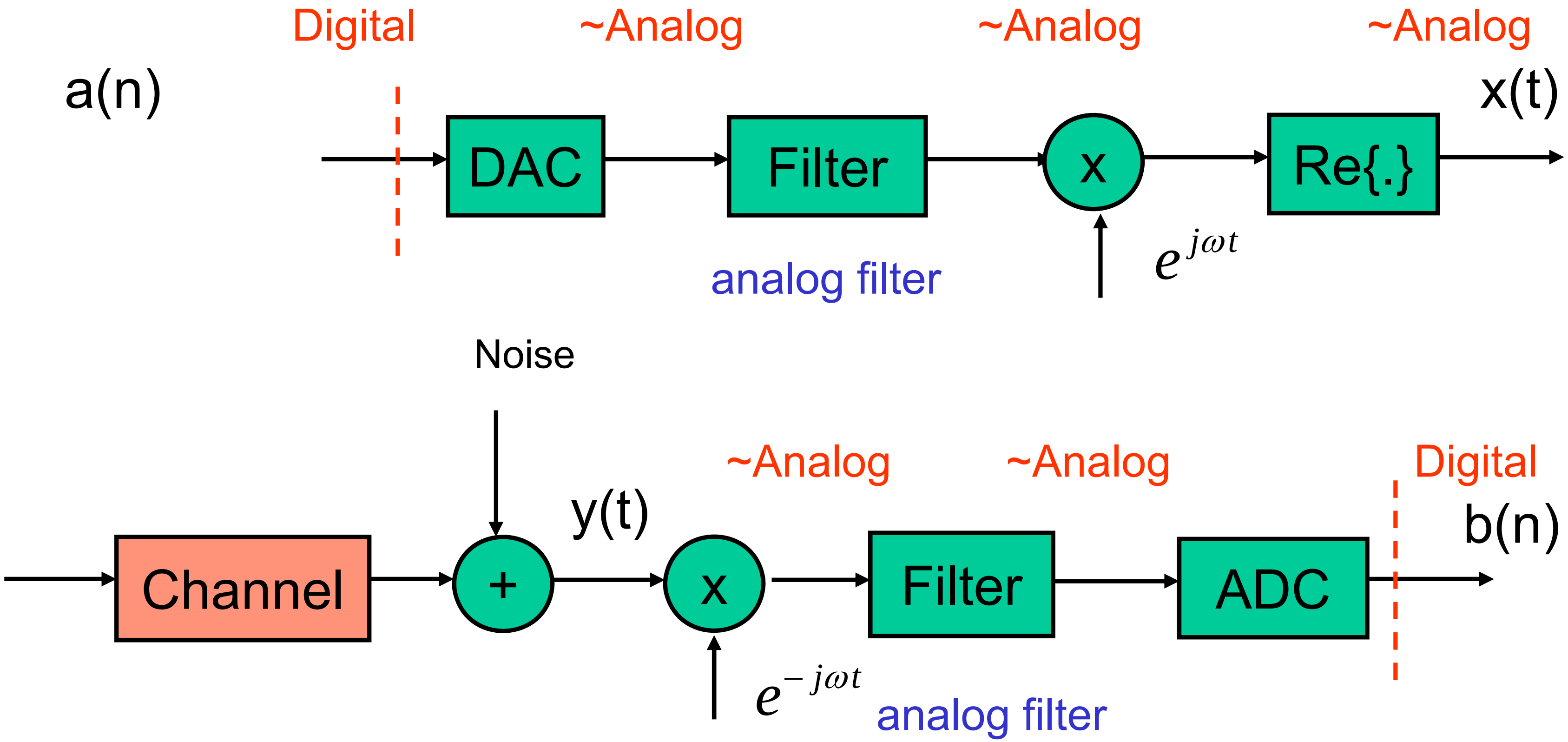
- Note that there is a phase delay problem in rx carriers. If not synchronized, the whole constellation will be rotated.

$$LPF\left\{2x(t)e^{-(j\omega t-\theta)}\right\}=\left[a_I(t)+jga_Q(t)e^{j\phi}\right]e^{j\theta}$$

- Reading assignment
  - z-transform, inverse z-transform

Lab 11. Equalization

- Channel effect:



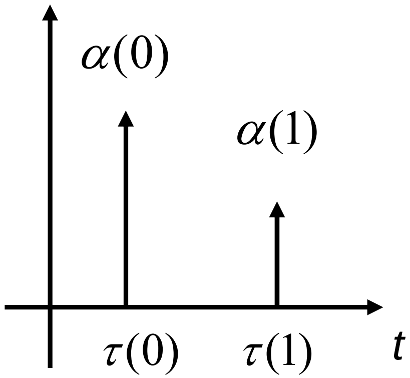
- Let

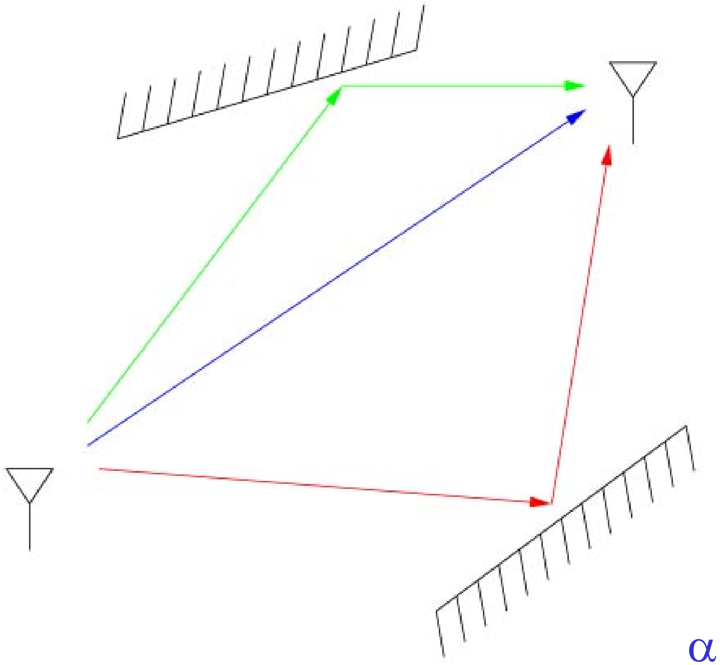
$$x(t) = \mathrm{Re}\left\{x_b(t)e^{j2\pi f_c t}\right\} \quad \text{(passband transmit)}$$

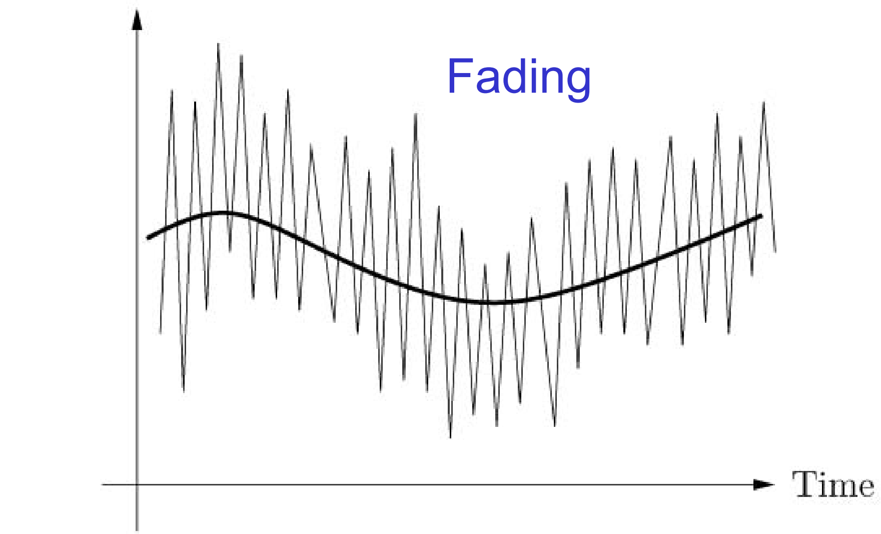
$$y(t) = \mathrm{Re}\left\{y_b(t)e^{2\pi f_c t}\right\} \quad \text{(passband receive)}$$

- Multi-path channel effect:

$$y(t) = \sum_i \alpha(i)x(t-\tau(i))$$







α(i): amplitude attenuation

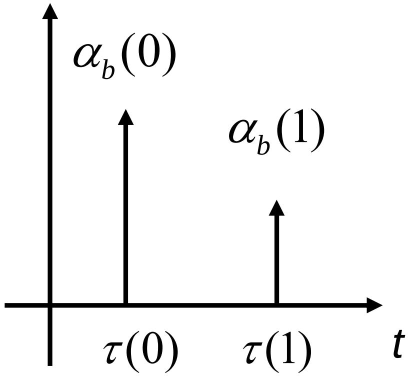
- Then,

$$y(t) = \mathrm{Re}\left\{ y_b(t) e^{j2\pi f_c t} \right\} = \sum_i \alpha(i)\,\mathrm{Re}\left\{ x_b(t-\tau(i)) e^{j2\pi f_c (t-\tau(i))} \right\}$$

$$= \mathrm{Re}\left\{ \left\{ \sum_i \alpha(i) x_b(t-\tau(i)) e^{-j2\pi f_c \tau(i)} \right\} e^{j2\pi f_c t} \right\}$$

$$= \mathrm{Re}\left\{ \left\{ \sum_i \alpha_b(i) x_b(t-\tau(i)) \right\} e^{j2\pi f_c t} \right\}$$

where $\boxed{\alpha_b(i) = \alpha(i) e^{-j2\pi f_c \tau(i)}}$



- Thus,

$$y_b(t) = \sum_i \alpha_b(i) x_b(t-\tau(i))$$

* Delays in passband are translated into baseband (complex exponentials)

- Let τ(i)=K_iT. After the ADC, we have

$$y_b(t) = \sum_i \alpha_b(i) x_b(t-\tau(i)) \quad \Rightarrow$$

$$y_b(mT) = \sum_i \alpha_b(i) x_b((m-K_i)T)$$

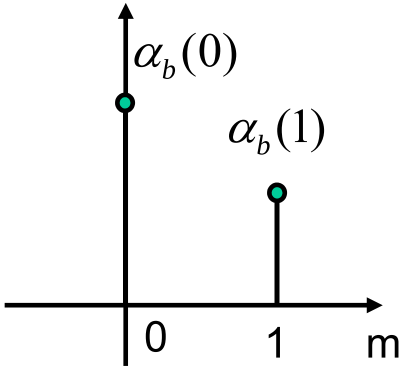
- The equivalent discrete-time signal is then

$$\boxed{y_b(m) = \sum_i \alpha_b(i) x_b(m-K_i)}$$

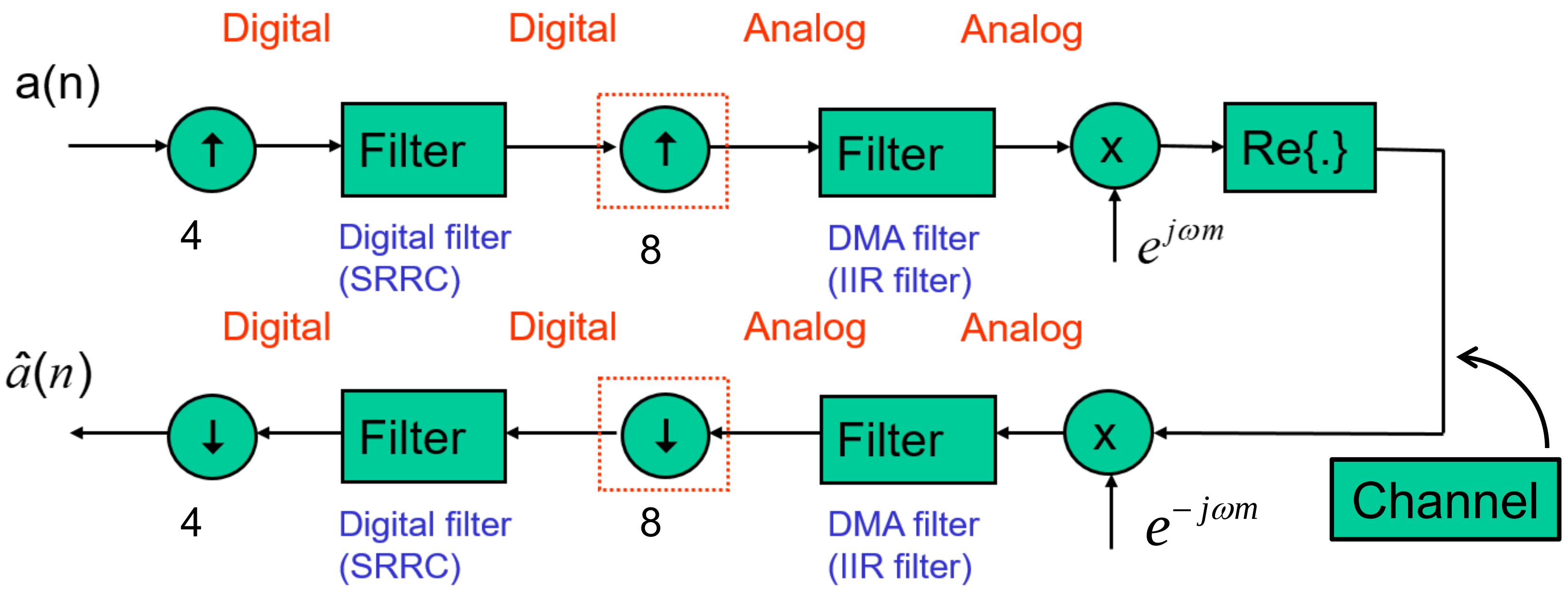
- For example: Let the number of path be 2 and K_0=0, K_1=1

$$y_b(m) = \alpha_b(0) x_b(m) + \alpha_b(1) x_b(m-1)$$

$$\Rightarrow \quad H(z) = \alpha_b(0) + \alpha_b(1) z^{-1}$$



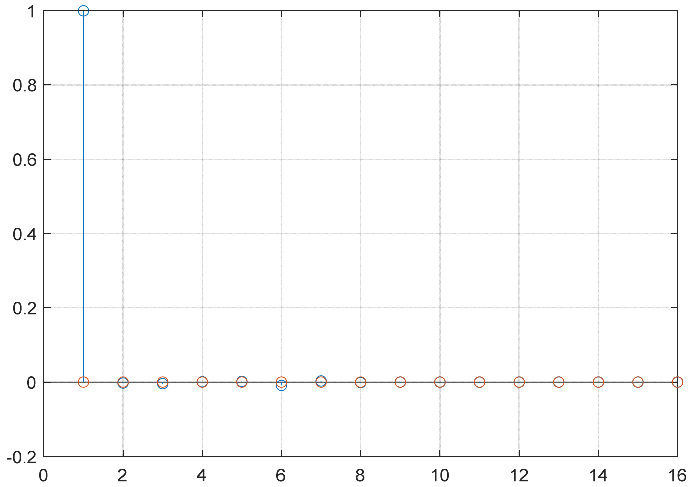
- Practice 1:
  - Find the equivalent baseband channel response of the system built in S10P1 (transmit an impulse, length 16).



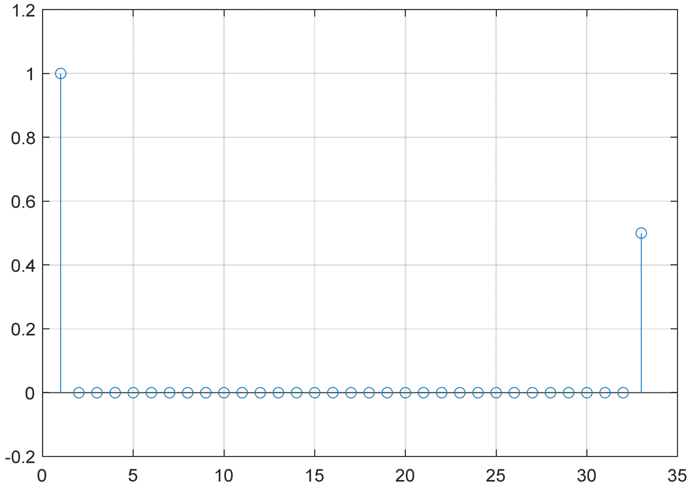
  - Add channel effect. Let the channel have two paths; the gain of the first path is one (without delay) and that of the second is 0.5 (one symbol period delay, i.e., 32 sampling periods). Find the equivalent baseband channel response.

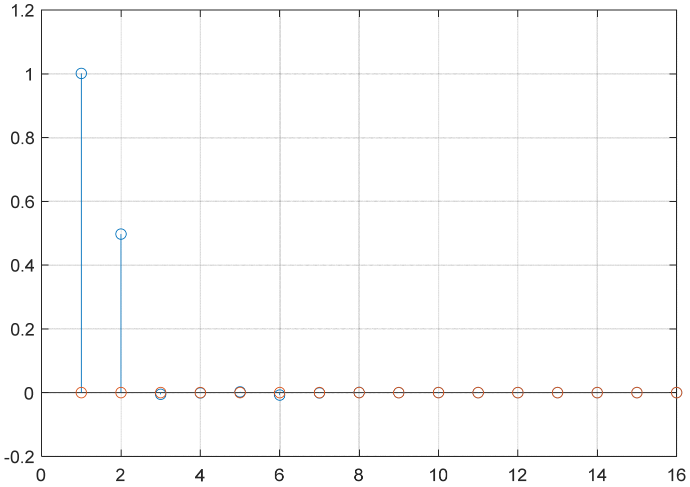
- – Let the time delay of the second path is 3.5 symbol periods (i.e. 32x3.5 sampling periods). Observe the baseband channel response again.

- Results:

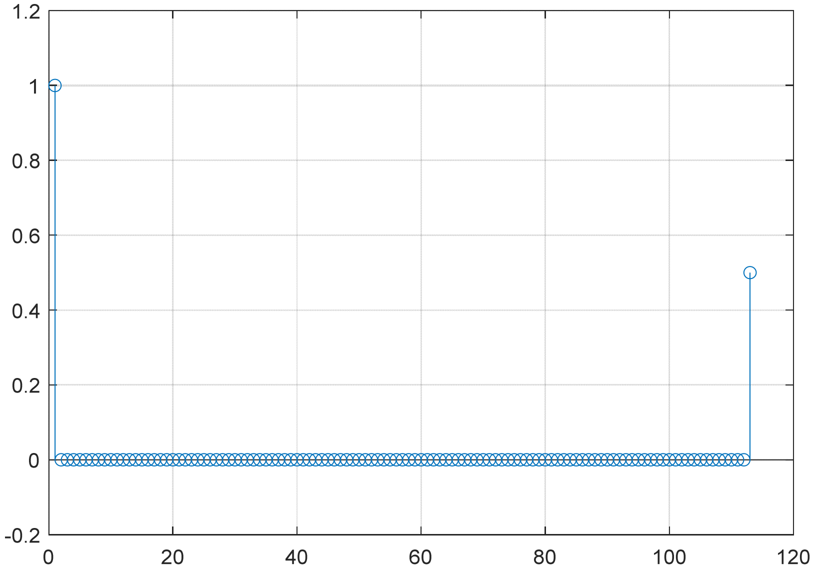


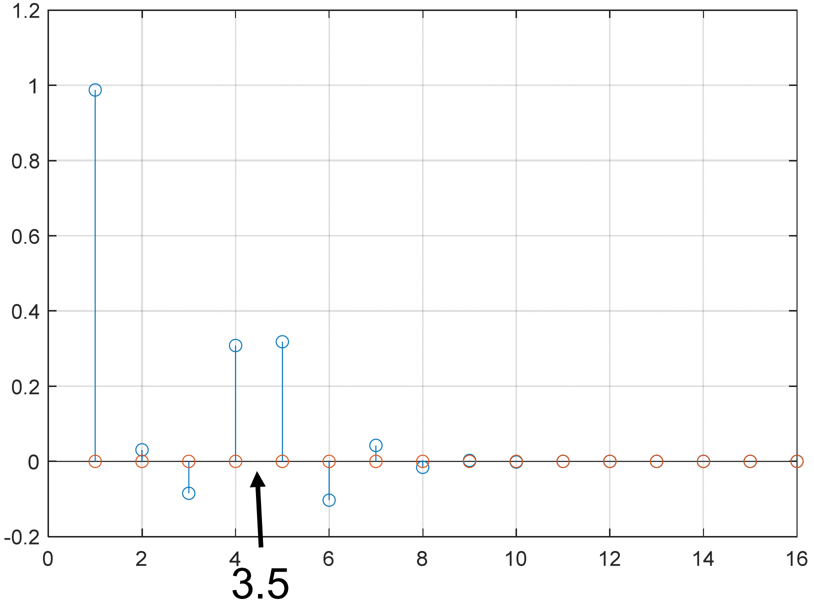
B: real
R: imaginary



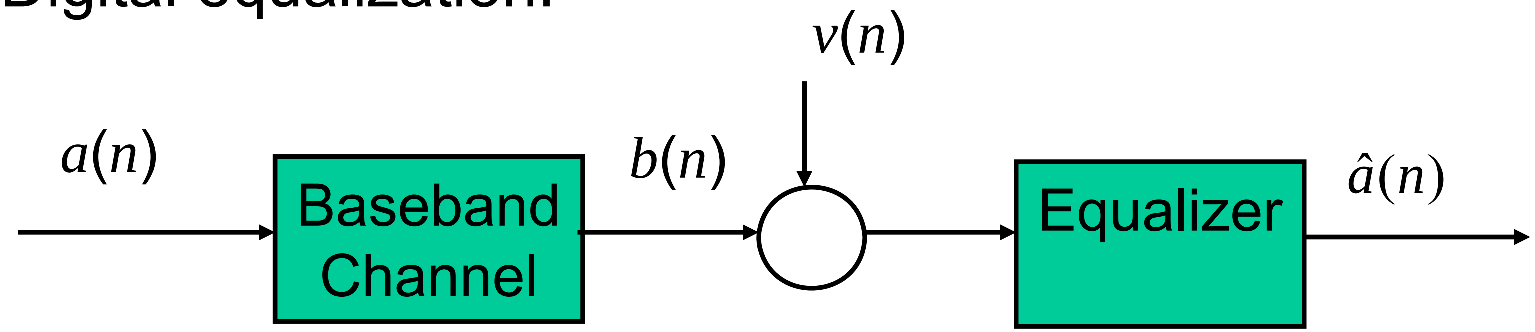


- Results:


1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
20
40
60
80
100
120


1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14
16
3.5

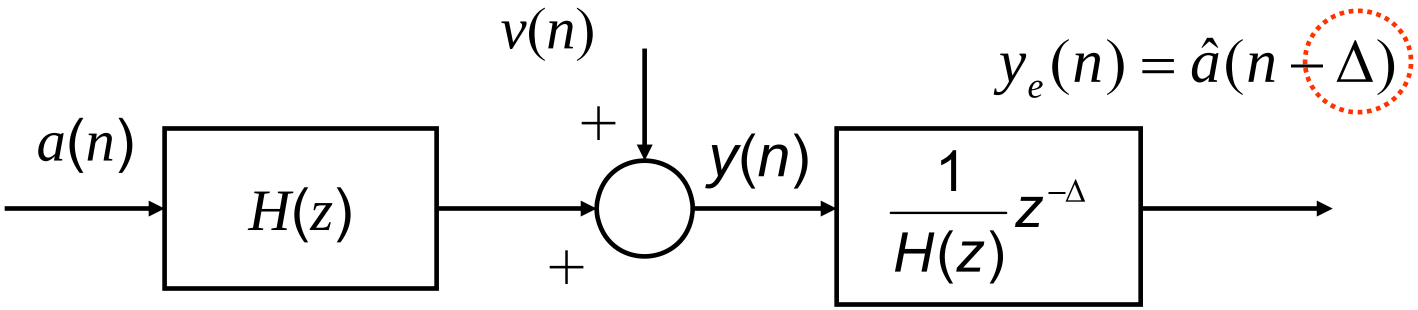
- Digital equalization:



$$b(n) = \sum_i a(n-i)\alpha_s(i) \qquad \hat{a}(n) = \sum_i b(n-i)w(i)$$

$\alpha_s(i)$: sampled channel response

- Zero-forcing (ZF) equalizer:



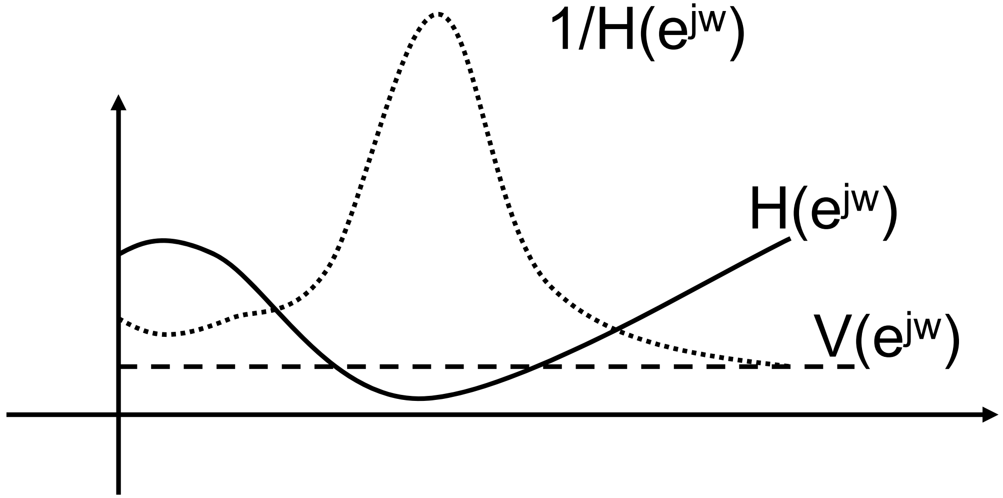
* The delay is for non-causal processing

- The ZF equalizer completely remove the ISI. However, noise can be greatly amplified.

$$Y(z)=H(z)A(z)+V(z)$$

$$Y_e(z)=\left[A(z)+\frac{V(z)}{H(z)}\right]z^{-\Delta}$$

$$Y_e(e^{j\omega})=\left[A(e^{j\omega})+\frac{V(e^{j\omega})}{H(e^{j\omega})}\right]e^{-j\omega\Delta}$$



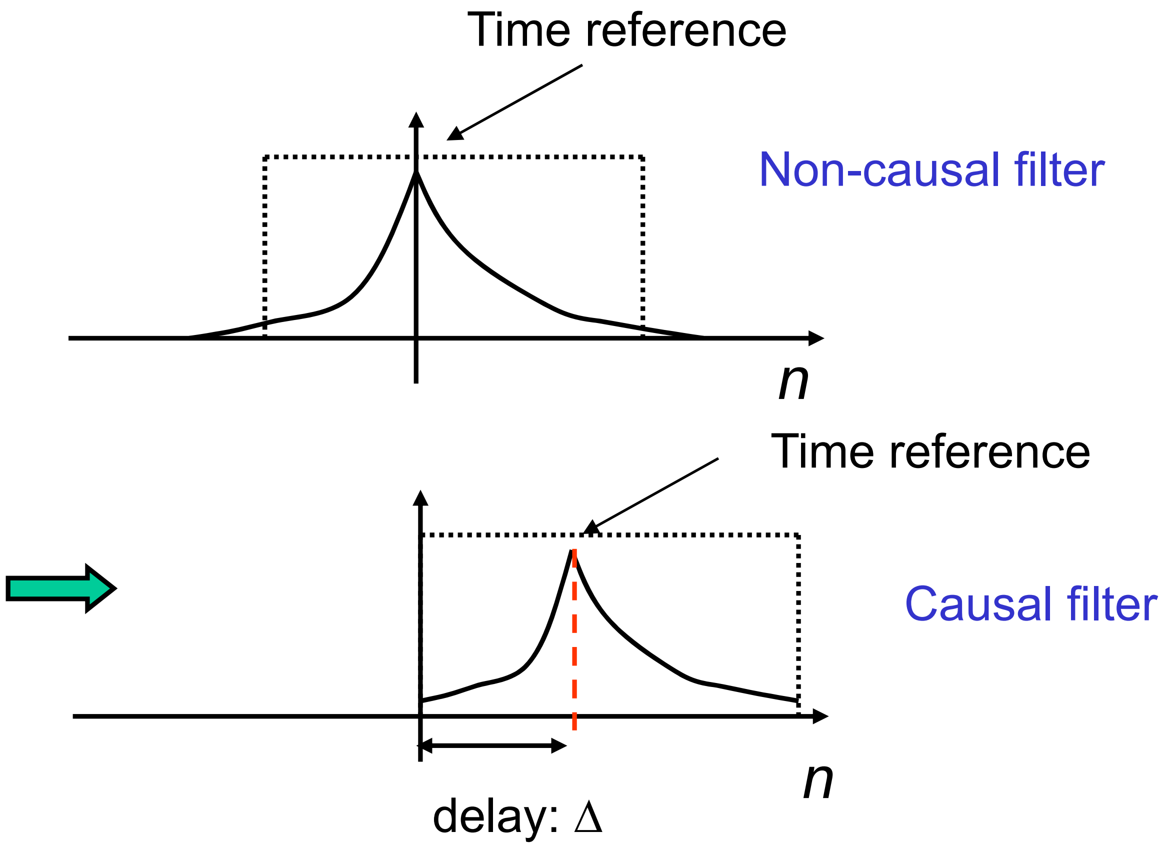
- Note that the non-causal processing capability is a distinct advantage of digital signal processing.
- For some H(z), the stable causal 1/H(z) does not exist. However, the stable non-causal 1/H(z) does exist.
- For example:

$$H(z) = 1 - 0.5z^{-1} \rightarrow \frac{1}{H(z)} = \frac{1}{1 - 0.5z^{-1}} \rightarrow w(n) = \begin{cases} (0.5)^n u(n) & \checkmark \\ -(0.5)^n u(-n-1) & \end{cases}$$

$$H(z) = 0.5 - z^{-1} \rightarrow \frac{1}{H(z)} = \frac{1}{0.5 - z^{-1}} = \frac{2}{1 - 2z^{-1}} \rightarrow w(n) = \begin{cases} (2)^{n+1} u(n) & \checkmark \\ -(2)^{n+1} u(-n-1) & \end{cases}$$

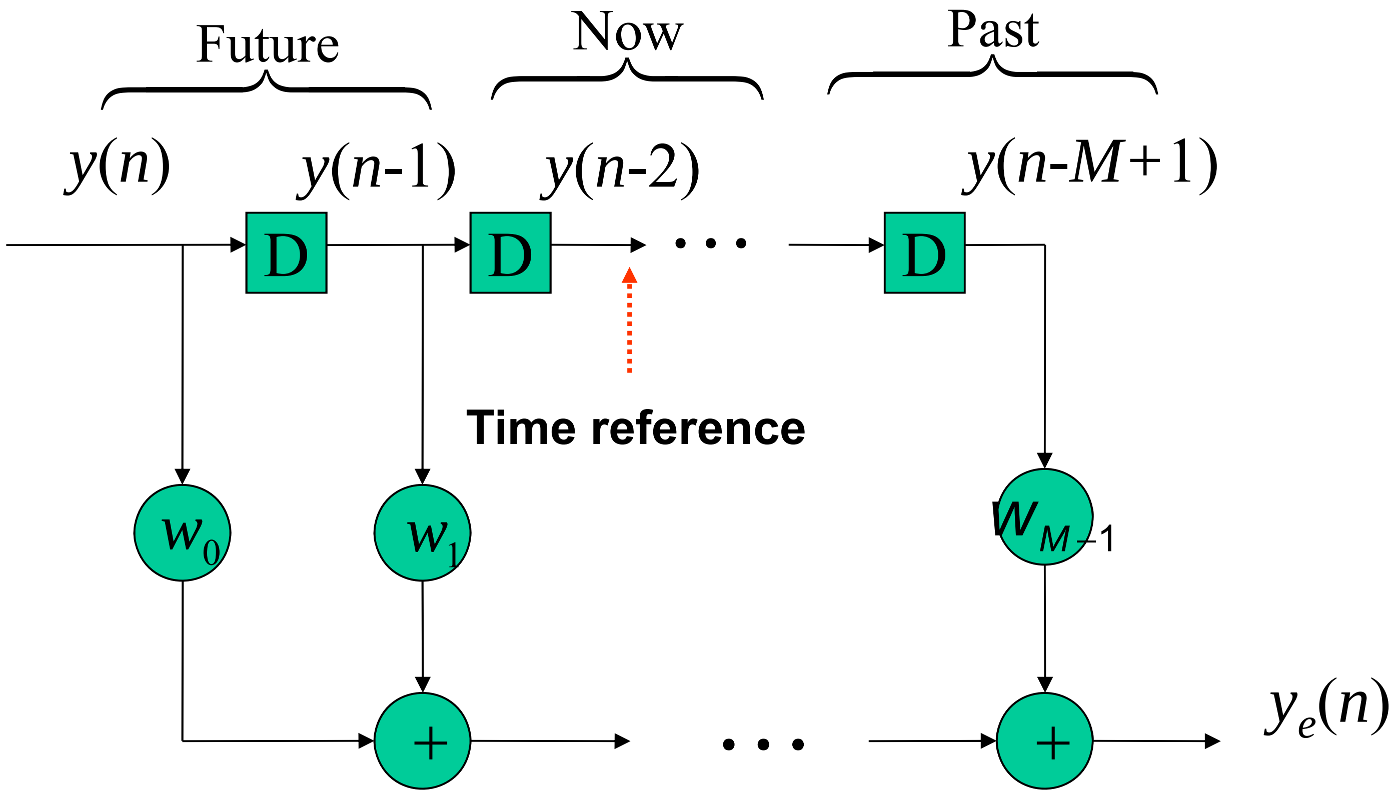
- We can realized the non-causal filter using truncation and delays.

- Non-causal FIR filtering:



- The filter output is then delayed.

- Non-causal FIR filtering:



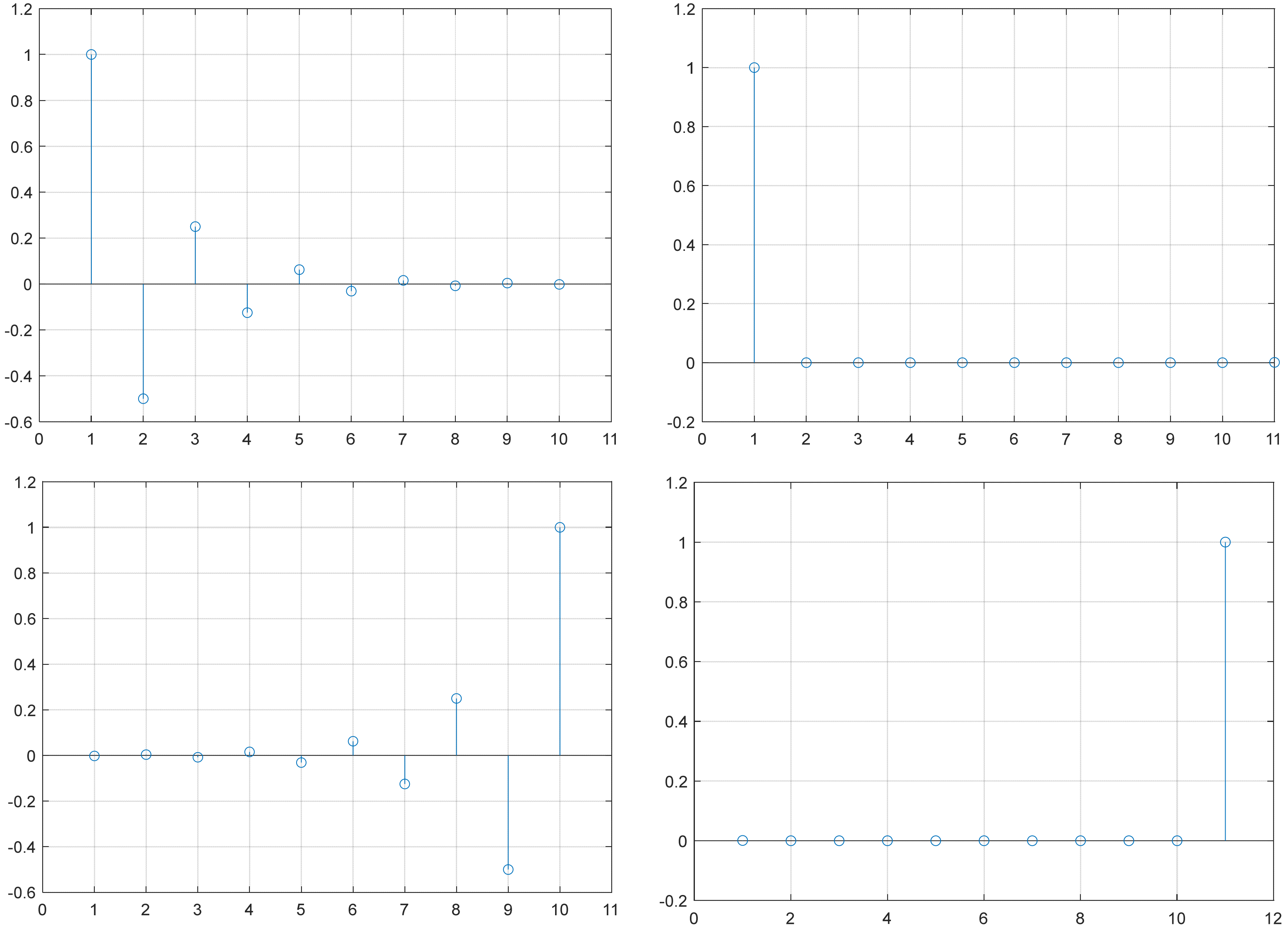
$$\mathbf{w} = [w_0 \ w_1 \ \ldots \ w_{M-1}]^T,$$

$$\mathbf{y}(n) = [y(n) \ y(n-1) \ \ldots \ y(n-M+1)]^T$$

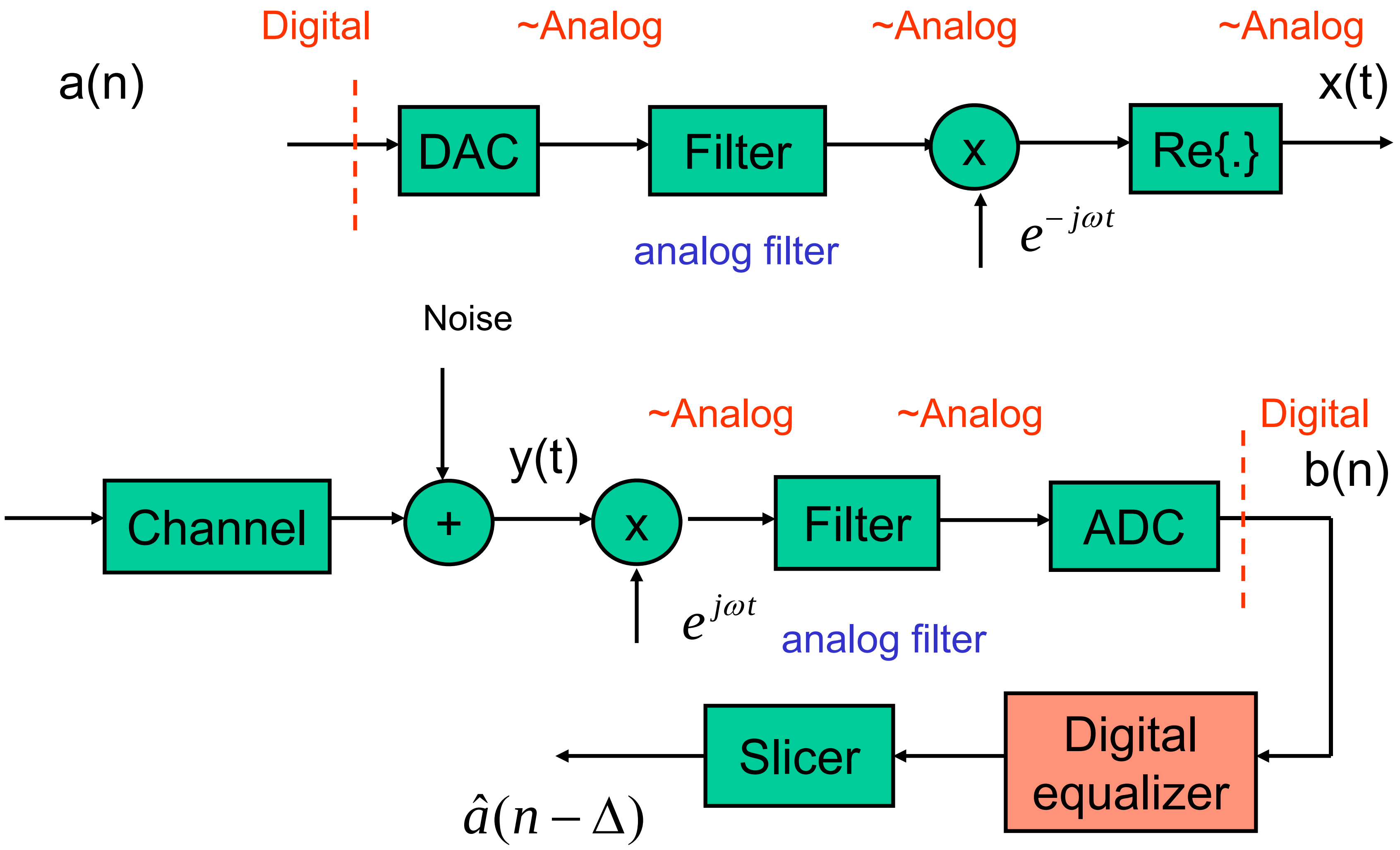
$$y_e(n) = w_n * u(n) = \mathbf{w}^T \mathbf{y}(n)$$

- Practice 2:
  - Consider two baseband channel responses [1, 0.5] and [0.5, 1]. Find the ZF equalizers of the channels (length 10).
  - Convolve the ZF equalizer with the channels to see if the channel is equalized.

- Result:


1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0 1 2 3 4 5 6 7 8 9 10 11
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0 1 2 3 4 5 6 7 8 9 10 11
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0 1 2 3 4 5 6 7 8 9 10 11
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0 2 4 6 8 10 12

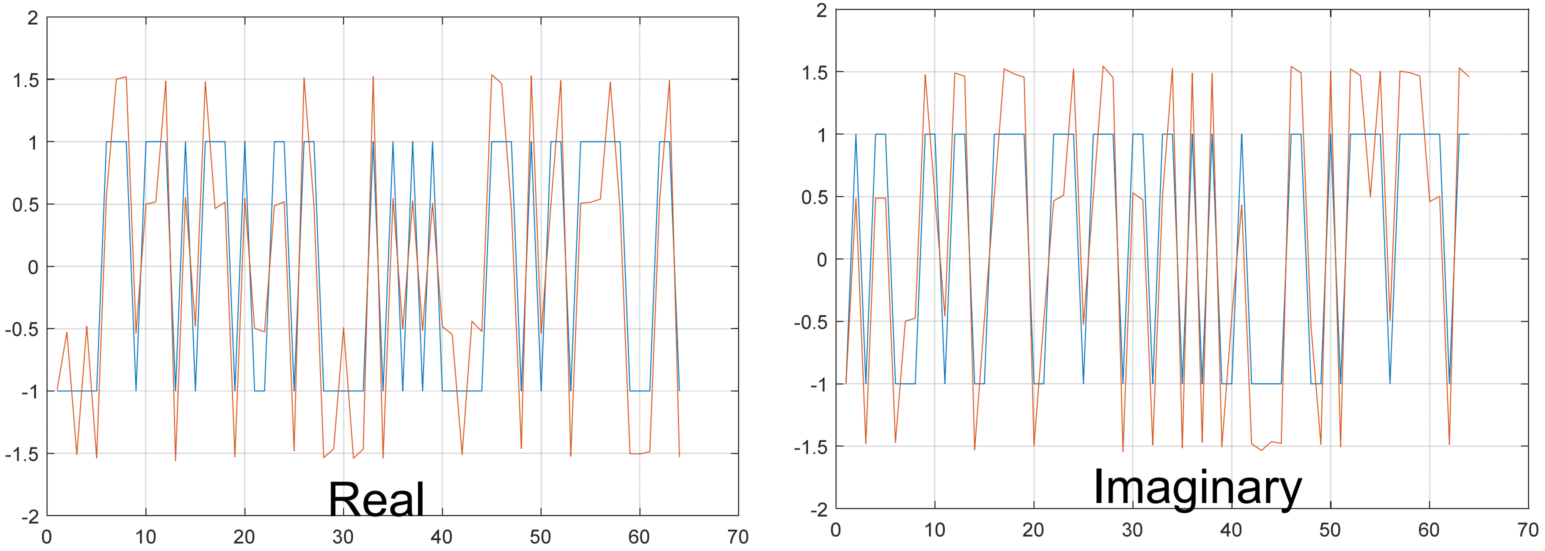
- Digital equalization:



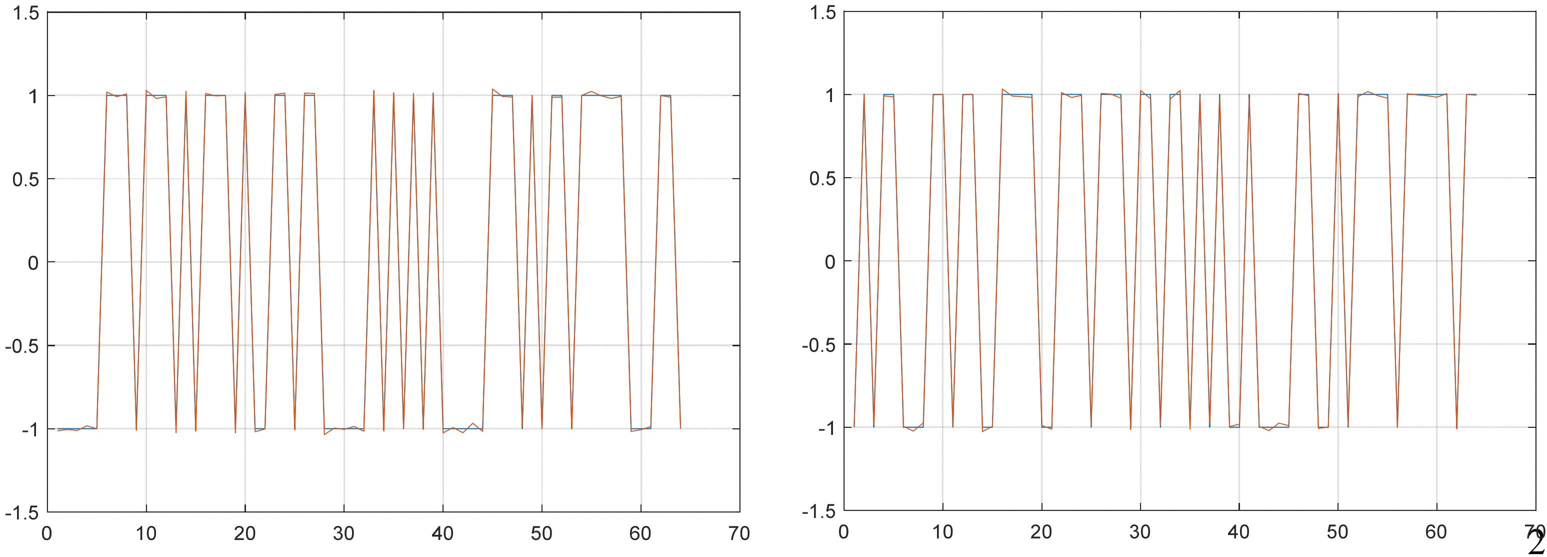
- Practice 3:
  - With the built system in Practice 1, assume the channel is the first two-path channel (i.e., path spacing is one symbol period), and obtain the recovered symbol sequence (given the symbol sequence S11P3.mat; QPSK).
  - Use the result obtained in Practice 2 to conduct digital ZF equalization recovering the transmit symbol signal.

- Result:

Without equalization



With equalization



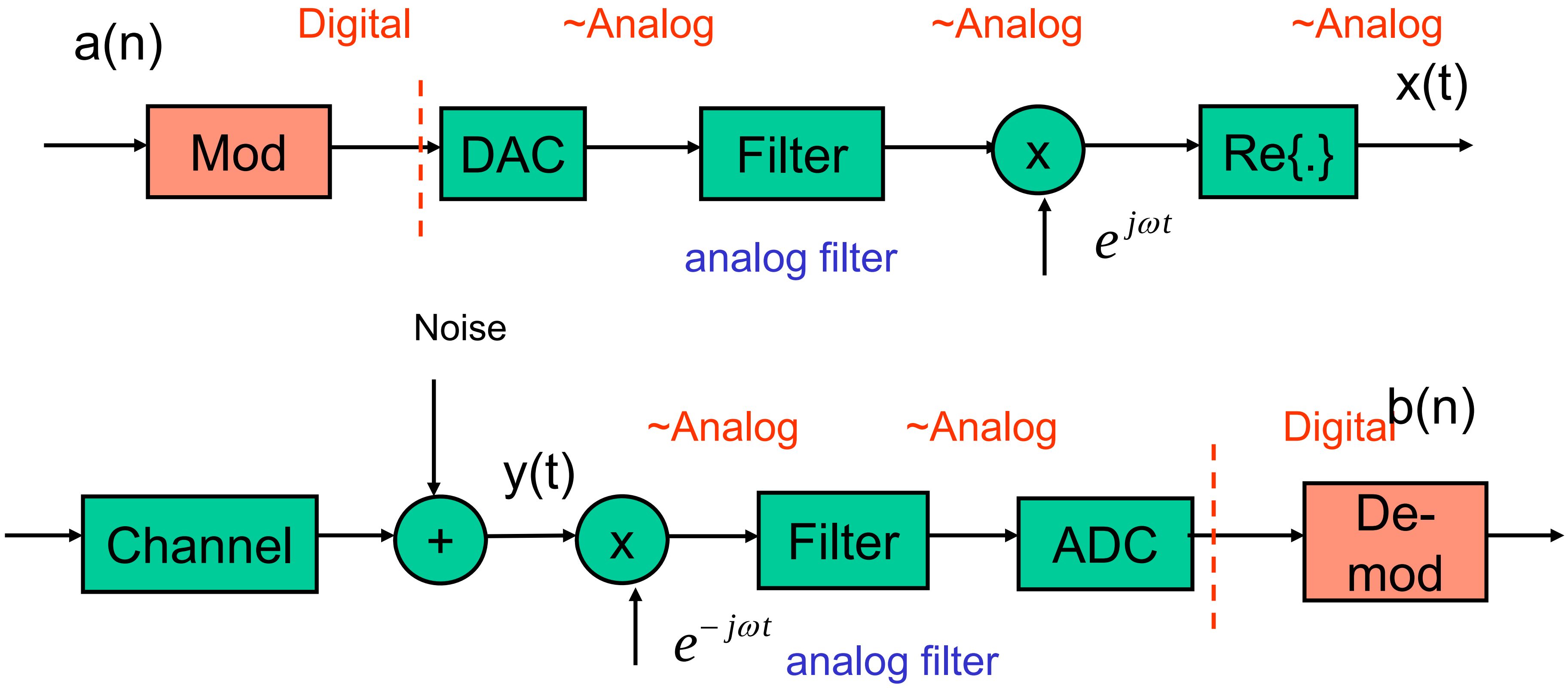
- Homework:
  - Assume that there are three paths ($\alpha(0)=0.3$, $\alpha(1)=1$, and $\alpha(2)=0.3$) in a wireless channel and the delay between any two paths is a symbol period. Find the equivalent baseband ZF equalizer.
  - With the built plateform in Practice 1 and the assumed channels, conduct digital ZF equalization to recovery the transmit symbol signal (QPSK).

Hint: (1) Find the roots of the transfer function of the channel
(2) Decompose the equalizer into two
(3) Truncate and delay the anti-causal one.

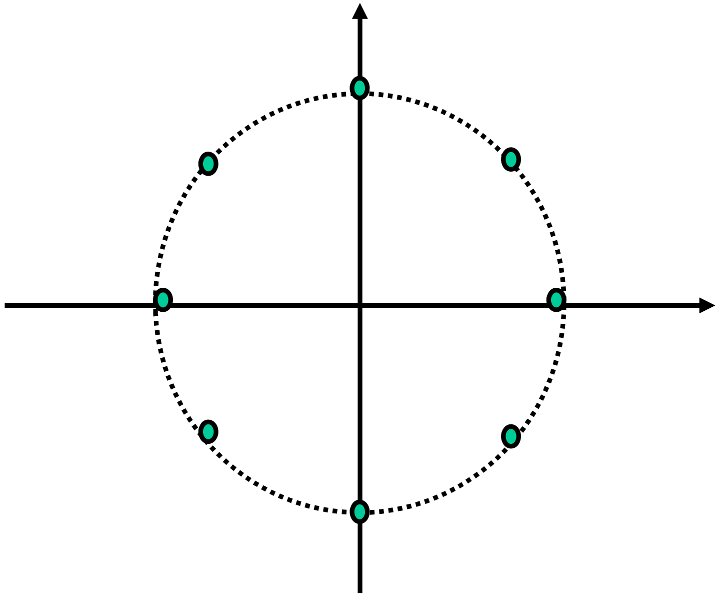
- Reading assignment:
  - CPFSK, MSK

Lab 12. Constant Envelop Modulation

- Digital communication system:



- Advantage of constant envelop modulation:
  - Efficiency of power amplifier can be maximized
- In QAM, only QPSK (maps two bits) has the constant envelop property.
- M-PSK:
  - e.g. 8-PSK (maps three bits)
- However, the higher M, the lower the efficiency.

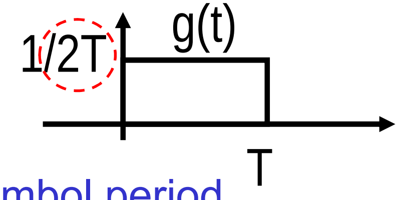


* Constant envelope

Continuous-phase frequency shift keying (CPFSK)

- Let a PAM signal be denoted as

$$s(t)=\sum_n a_n g(t-nT), \quad a_n=\pm1,\pm3,\cdots$$



T : symbol period

- A continuous-phase (CPFSK) modulated signal is expressed as

$$x(t)=\cos\left[2\pi f_c t+2\pi f_d (2T)\int_{-\infty}^{t} s(\tau)d\tau\right]$$

f_d : peak frequency deviation

- We can also express the signal as

$$x(t)=\cos\left[2\pi f_c t+\theta(t)\right]$$

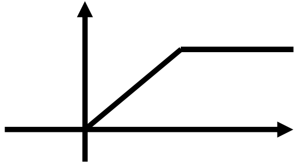
- Then, its phase in $nT \le t \le (n+1)T$, is

$$\theta(t) = 2\pi f_d T \sum_{n=-\infty}^{n-1} a_k + 2\pi(\underbrace{2 f_d T}_{h}) \underbrace{\left(\frac{t-nT}{2T}\right)}_{q(t-nT)} a_n$$

$$= \theta_n + 2\pi h a_n q(t-nT)$$

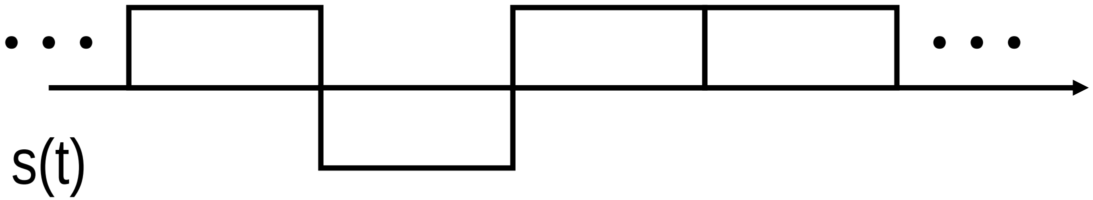
$$\boxed{h = 2 f_d T = 2\left(\frac{f_d}{f_s}\right)}$$

f_s: symbol rate

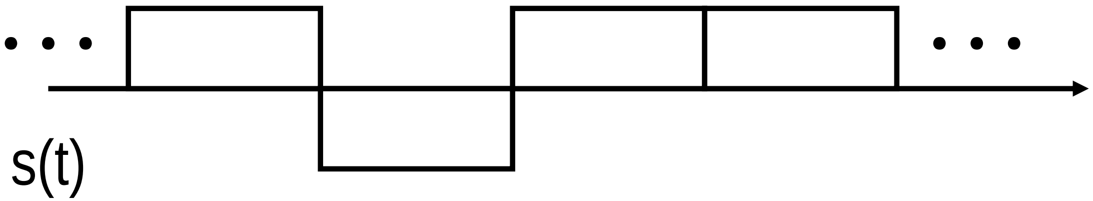


$$q(t) = \begin{cases} 0, & t < 0 \\ t/2T, & 0 \le t \le T \\ 1/2, & t > T \end{cases}$$

- The parameter h is called the *modulation index*.
- When a_n is binary and h=1/2, the CPFSK is called the minimum shift keying (MSK), indicating the *minimum* FSK frequency deviation having the *orthogonal* property.
- For binary case, we have a NRZ signal for transmission.



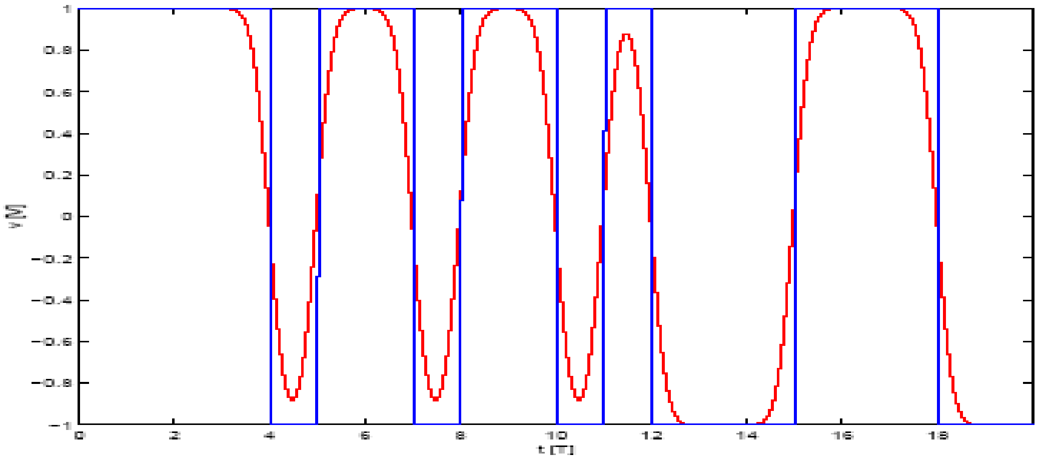
- If we apply a Gaussian filter to s(t) and then CPFSK modulation, we then have the GFSK.



* Gaussian filter

- Example:

B: s(t) R: $s_f(t)$



- The filtering operation can reduce the transmission bandwidth. However, the downside is reducing the system performance, and increasing the complexity of the receiver design.
- The GFSK signal becomes

$S_f(t)$: Gaussian filtered pulse

$$x(t) = \cos\left[2\pi f_c + 2\pi f_d (2T) \int_{-\infty}^{t} s_f(\tau) d\tau\right]$$

- The demodulation reverses the process. Let r(n) be the down-converted baseband received signal. What we have to do is
  - Take the phase of r(n)
  - Differentiate the phase of r(n)
  - Gaussian filtered pulse matching
  - Detect the transmit signal (as the ordinary NRZ signal)

- Let B be the 3dB bandwidth of the Gaussian filter. Its frequency and impulse responses can be expressed as

$$H(f) = \exp\left(-\frac{\ln 2}{2}\left(\frac{f}{B}\right)^2\right)$$

* exp(-ln2/2)=0.707

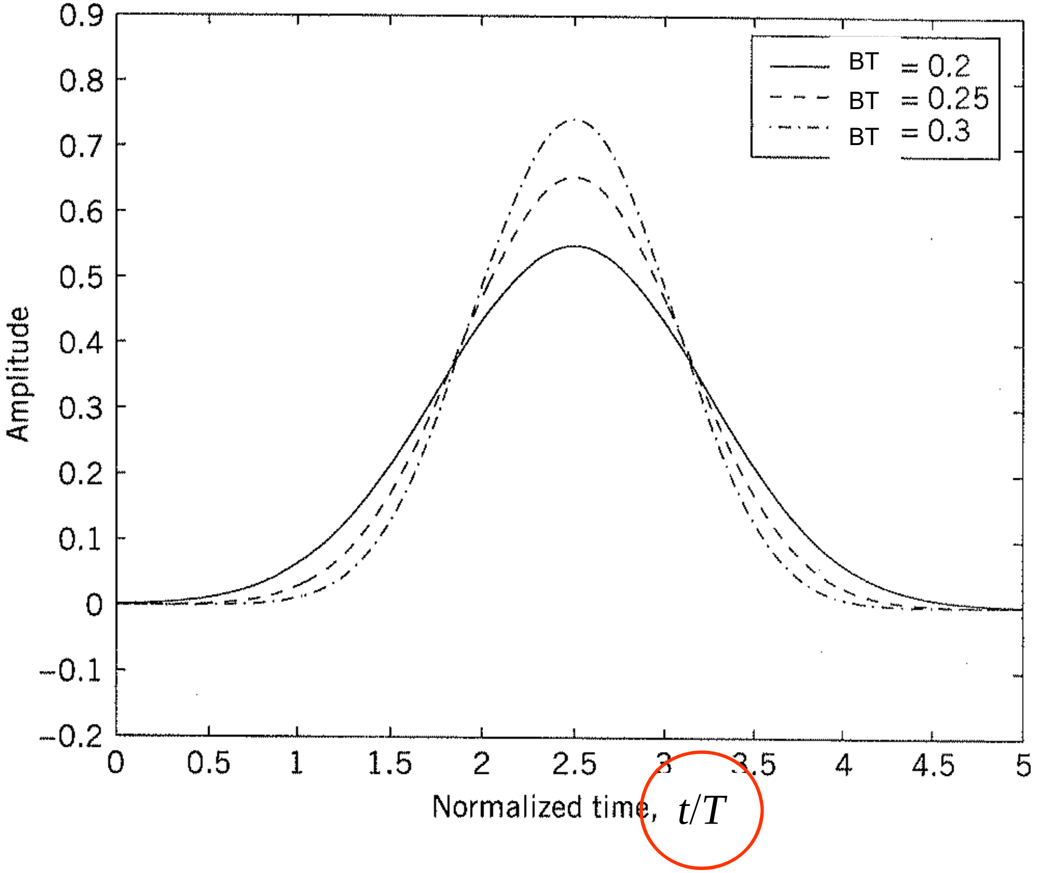
$$h(t) = \sqrt{\frac{2\pi}{\ln 2}}B\exp\left(-\frac{2\pi^2}{\ln 2}B^2t^2\right)$$

- The response of this Gaussian filter to a rectangular pulse of unit amplitude and duration T (symbol period) is given by

$$g(t) = \int_{-T/2}^{T/2} h(t-\tau)d\tau = \sqrt{\frac{2\pi}{\ln 2}}B\int_{-T/2}^{T/2}\exp\left(-\frac{2\pi^2}{\ln 2}B^2(t-\tau)^2\right)d\tau$$

- The response can be expressed as the difference between two complementary error functions as

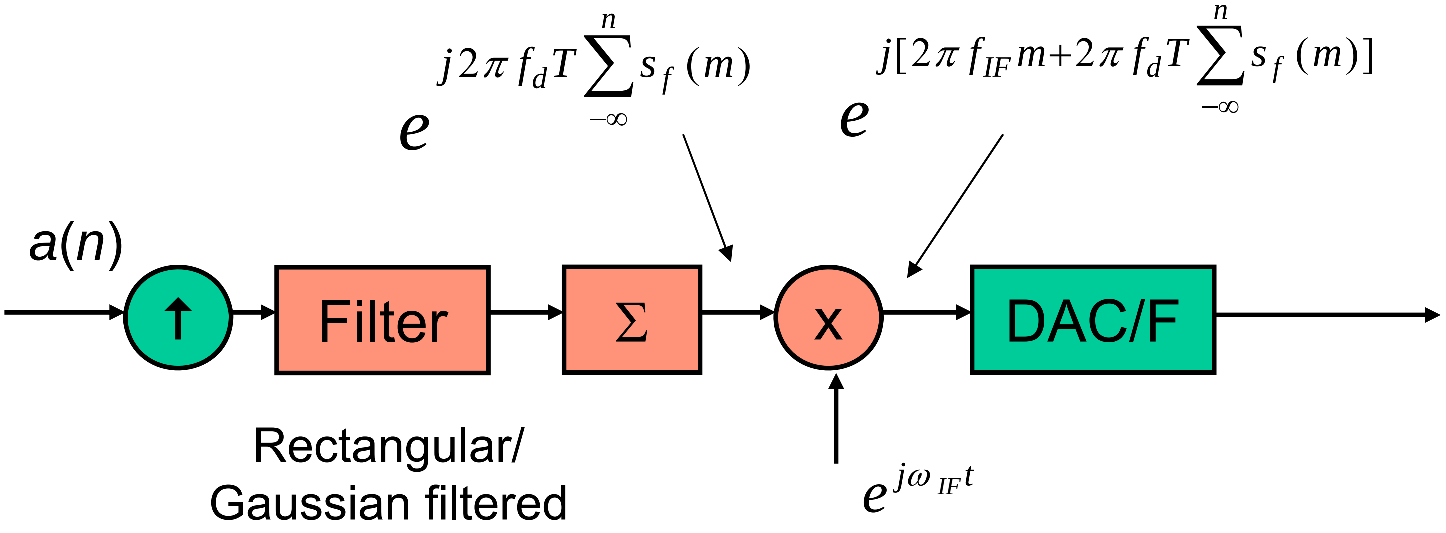
$$g(t) = \frac{1}{2}\left[erfc\left(\pi\sqrt{\frac{2}{\ln 2}}BT\left(\frac{t}{T}\right) - \frac{1}{2} \right) - erfc\left(\pi\sqrt{\frac{2}{\ln 2}}BT\left(\frac{t}{T}\right) + \frac{1}{2} \right) \right]$$



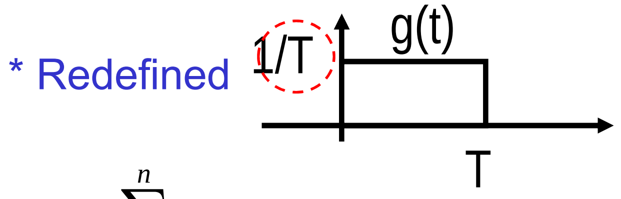
BT: design parameter

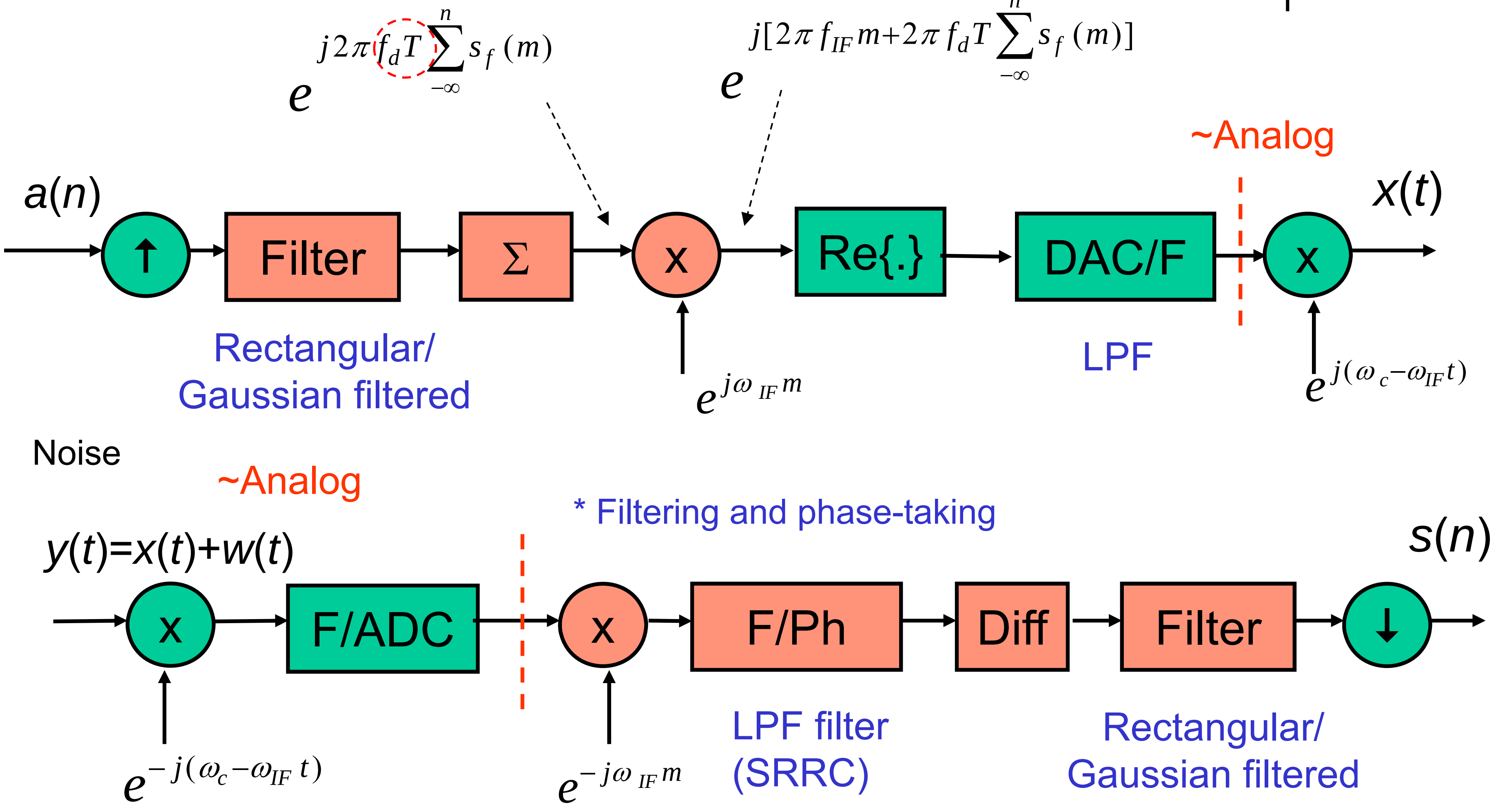
* If BT is the same, the normalized pulse is the same.

- In practice, the GFSK scheme is often digitally implemented with an low IF architecture.
  - The system has to be oversampled.
  - The integration is replaced with a summation.
  - The differentiation is replaced with a difference operation.


$e^{j2\pi f_d T \sum_{-\infty}^{n} s_f(m)}$
$e^{j[2\pi f_{IF} m + 2\pi f_d T \sum_{-\infty}^{n} s_f(m)]}$
$a(n)$
↑
Filter
Σ
x
DAC/F
Rectangular/
Gaussian filtered
$e^{j\omega_{IF} t}$

- The system:


* Redefined
1/T
g(t)
T


$e^{j2\pi f_d T \sum_{-\infty}^{n} s_f(m)}$
$e^{j[2\pi f_{IF} m + 2\pi f_d T \sum_{-\infty}^{n} s_f(m)]}$
~Analog
$a(n)$
↑
Filter
Σ
x
Re{.}
DAC/F
x
$x(t)$
Rectangular/
Gaussian filtered
$e^{j\omega_{IF} m}$
LPF
$e^{j(\omega_c - \omega_{IF} t)}$
Noise
~Analog
* Filtering and phase-taking
$y(t)=x(t)+w(t)$
$s(n)$
x
F/ADC
x
F/Ph
Diff
Filter
↓
$e^{-j(\omega_c - \omega_{IF} t)}$
$e^{-j\omega_{IF} m}$
LPF filter
(SRRC)
Rectangular/
Gaussian filtered

- Practice 1:
  - Let the BT of a Gaussian filter be 0.5. Plot its sampled waveform (M=16, S=1)

$$h(t) = \sqrt{\frac{2\pi}{\ln 2}} B \exp\left(-\frac{2\pi^2}{\ln 2} B^2 t^2 \right)$$

$$\Rightarrow h(n) = h(nT_{sp}) = C \exp\left(-\frac{2\pi^2}{\ln 2} \left(\frac{B}{f_{sp}} \right)^2 n^2 \right)$$

$$= C \exp\left(-\frac{2\pi^2}{\ln 2} \left(\frac{B}{Mf_s} \right)^2 n^2 \right) = C \exp\left(-\frac{2\pi^2}{\ln 2} \left(\frac{BT}{M} \right)^2 n^2 \right)$$

C : normalization constant,

M: oversampling factor

* T_{sp}: sampling period

* f_{sp}: sampling freq.

$* \ f_{sp} = Mf_s$

$* \ \frac{B}{f_s} = BT$

* T : symbol period

* f_s: symbol rate

  - Write the program as a function (gaufilter(M, BT, S))

- Practice 2:
  - Let the parameters of a GFSK system be given as:
    - BPSK symbol rate :1MHz
    - f_d =150KHz
    - f_{IF} =2MHz
    - BT=0.5
    - Sampling rate of DAC/ADC :16MHz
    - Sampling rate of analog signal: 64MHz
  - Design a DMA filter.
  - Use SRRC filter as the LPF for IF demodulated signal.
  - Construct the transmitter and the receiver (ignoring the analog carriers). The input is given in S12P2.mat.
  - Observe demodulated phase signals (filtered/unfiltered).

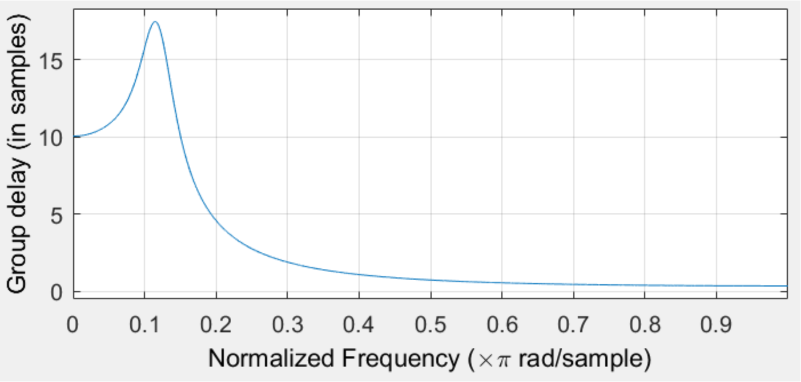
- Phase accumulation for modulation:

$$\theta(nT+t)=\theta(nT)+2\pi f_d T\left(\frac{t}{T}\right)a_{n+1}, \qquad 0<t\le T$$

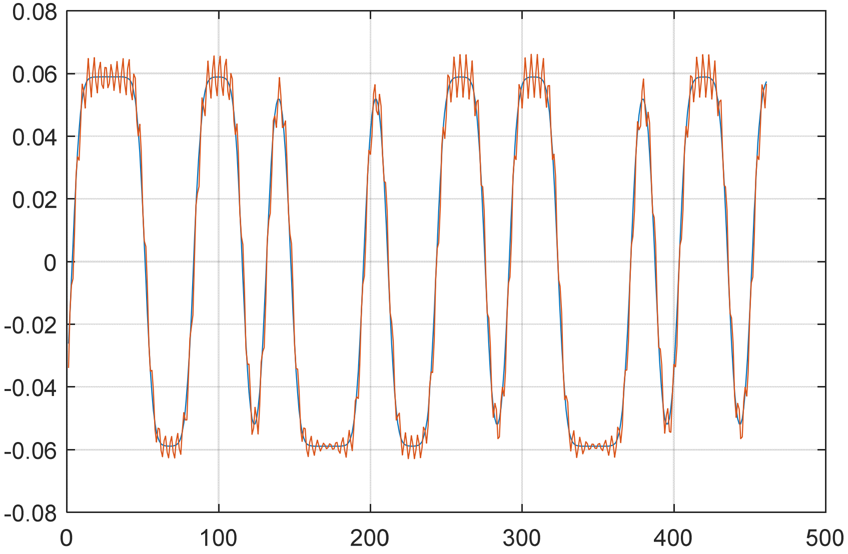
$$\Rightarrow \theta(nT+\frac{m}{M}T)=\theta(nT)+2\pi\left(\frac{f_d}{f_s}\right)\frac{m}{M}a_{n+1}=\theta(nT_b)+2\pi m\left(\frac{f_d}{Mf_s}\right)a_{n+1}, \quad 0<m\le M$$

* M : oversampled factor

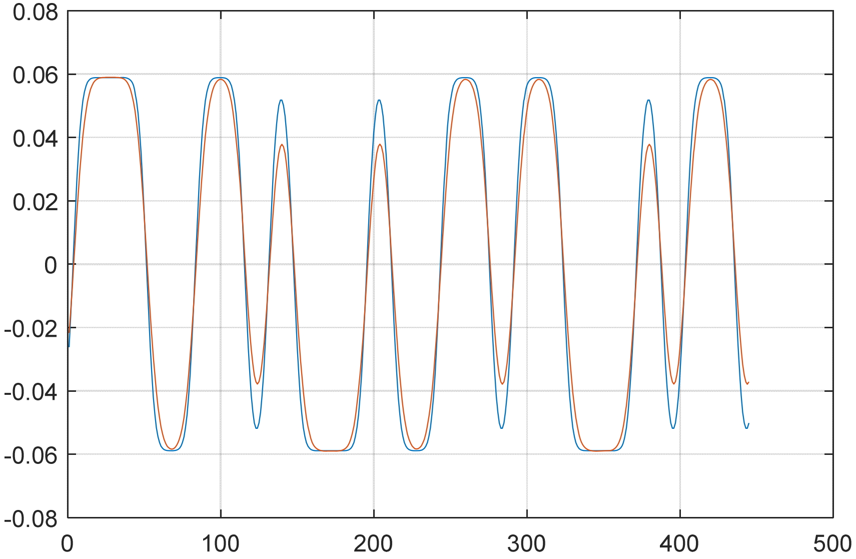
- Results:



Group delay: IIR filter



Before Gaussian filtering



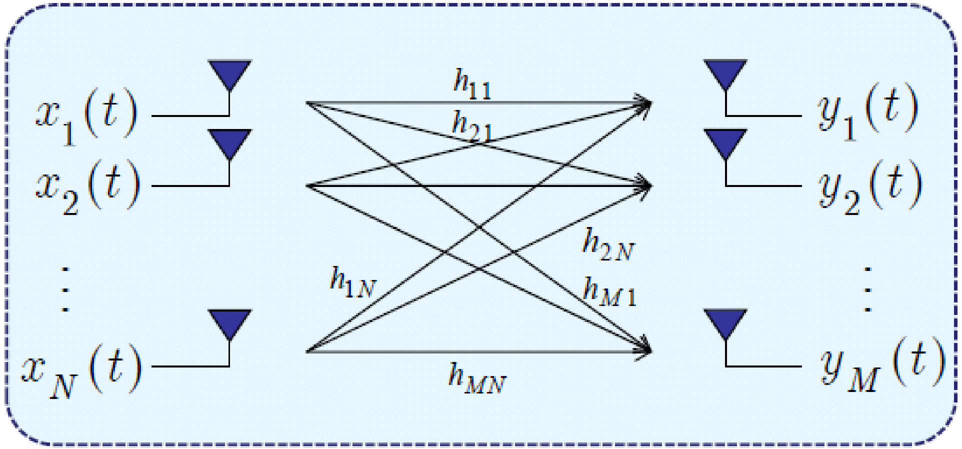
After Gaussian filtering

- Homework:
  - Add noise and observe the performance of the system constructed in Practice 2.
  - Change the frequency deviation to 300KHz and observe the performance of the system.
  - Compare the transmit signal spectrums of the CPFSK system with the two frequency deviations.

Lab. 13 MIMO Transmission

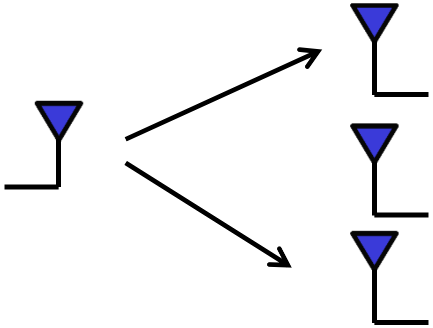
- Conventional communication systems only have one transmit/receive antenna.
- With multiple transmit/receive antennas, a multiple-input-multiple-output (MIMO) system can be formed, and the system capacity can be effectively improved.

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_N(t) \end{bmatrix}$$



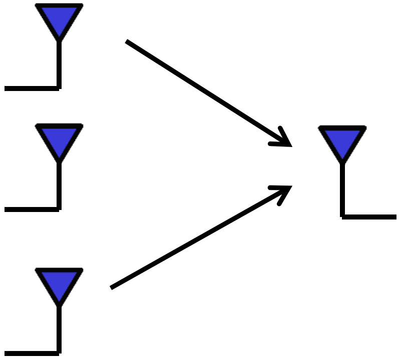
$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_M(t) \end{bmatrix}$$

- For single-input-multiple-output (SIMO) systems, i.e., n_T=1, we have



$$\begin{bmatrix} y_1(n) \\ y_2(n) \\ \vdots \\ y_{n_R}(n) \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_{n_R} \end{bmatrix} x(n)$$

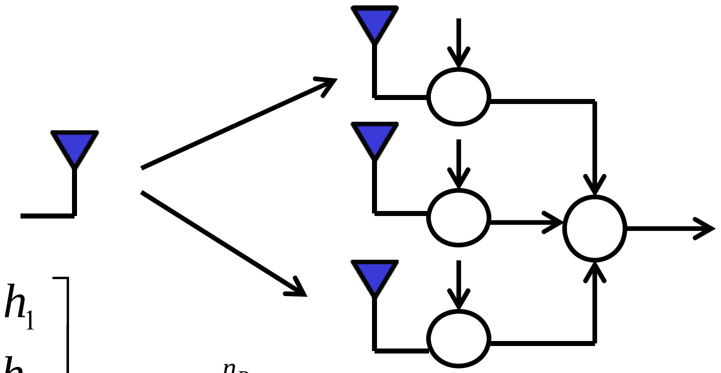
- For multiple-input-single-output (MISO) systems, i.e., n_R=1, we have



$$y(n) = \left(h_1 + h_2 + \cdots + h_{n_T}\right) x(n)$$

- The optimum transmit/receive schemes are called beamforming.

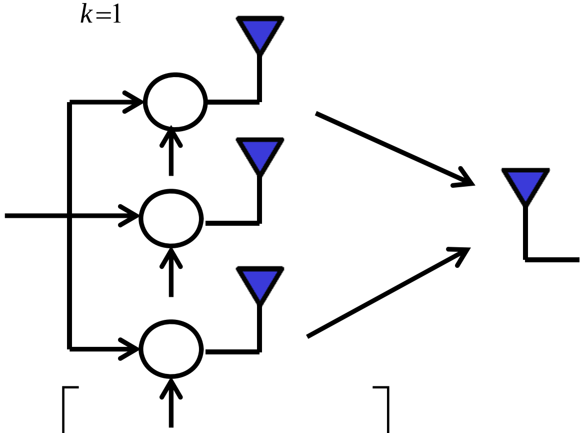
- Receive beamforming:



$$\begin{bmatrix} h_1^* & h_2^* & \cdots & h_{n_R}^* \end{bmatrix}\begin{bmatrix} y_1(n) \\ y_2(n) \\ \vdots \\ y_{n_R}(n) \end{bmatrix} = \begin{bmatrix} h_1^* & h_2^* & \cdots & h_{n_R}^* \end{bmatrix}\begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_{n_R} \end{bmatrix} x(n) = \sum_{k=1}^{n_R} |h_k|^2 \, x(n)$$

* Assume it is noise free

- Transmit beamforming:

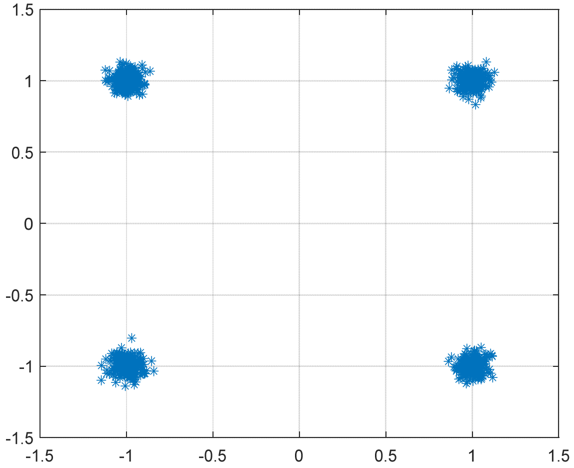


$$y(n) = (\ \ )\left[\frac{h_1^*}{\sqrt{\sum_{k=1}^{N} |h_k|^2}} x(n)\right] h_1 + \left[\frac{h_2^*}{\sqrt{\sum_{k=1}^{N} |h_k|^2}} x(n)\right] h_2 + \cdots + \left[\frac{h_N^*}{\sqrt{\sum_{k=1}^{N} |h_k|^2}} x(n)\right] h_{n_T}$$

$$= \sqrt{\sum_{k=1}^{n_T} |h_k|^2}\, x(n)$$

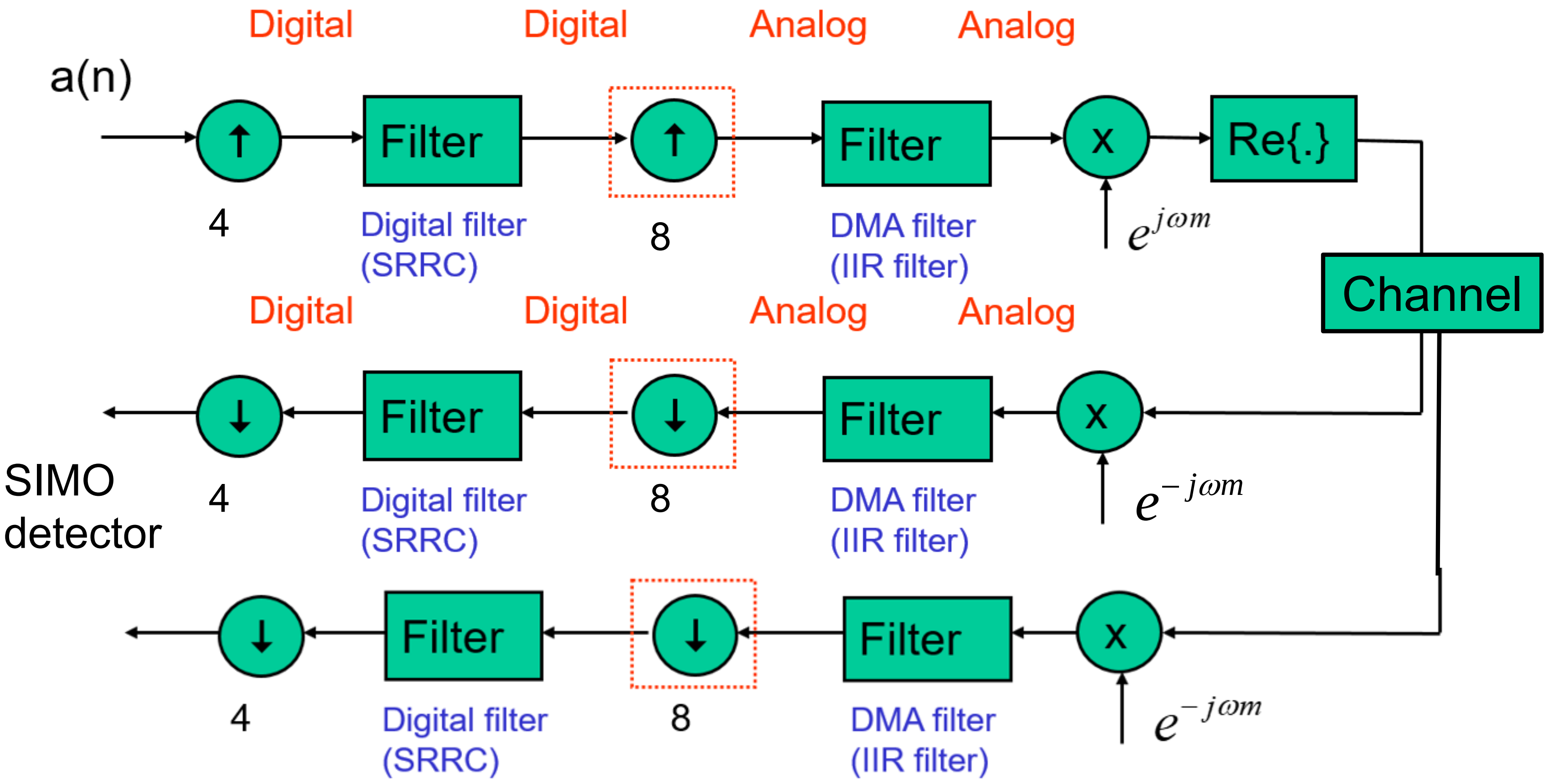
- Practice 1:
  - Given a QPSK sequence and a 1x2 baseband channel (S13P1.mat). Let the sequence pass the channel and be added AWGN with standard deviation of 0.1 (in each dimension). Conduct receive beamforming at the receiver
  - Conduct ZF equalization and calculate the standard deviation of the error

- Result:

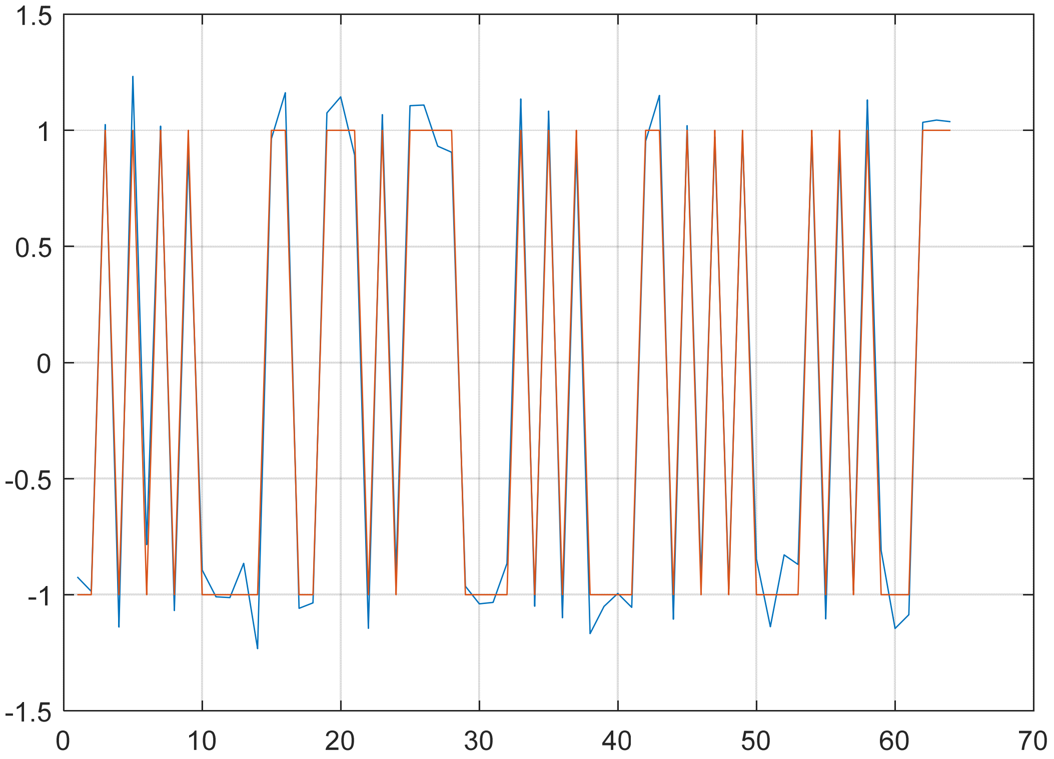


Sd_noise=0.1
Sd_err=0.0426

- Practice 2:
  - Extend the system in S10P1 to have a 1x2 channel and add AWGN with σ=0.03. Conduct the receive beamforming for the system (S13P2.mat).



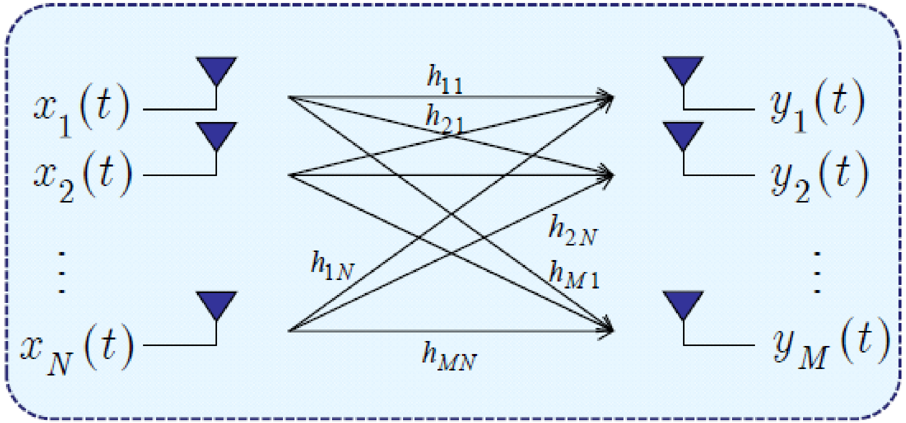
- Result:



- In general transmit beamforming is more difficult to conduct since the channel state information is required.
- In MIMO systems, the antennas are separated far enough such that there is no correlation between antennas.
- In rich scattering environments, this assumption is easier to hold.
- Using vector and matrix representations, we can make the signal transmission directional (in a vector space) such that simultaneous transmission of multiple bit streams becomes possible (pointing to different directions).

- MIMO system:

$$\mathbf{x}(t)=\begin{bmatrix} x_1(t) \\ \vdots \\ x_N(t) \end{bmatrix}$$



$$\mathbf{y}(t)=\begin{bmatrix} y_1(t) \\ \vdots \\ y_M(t) \end{bmatrix}$$

- Then we can have

Directions

$$\begin{bmatrix} y_1(n) \\ y_2(n) \\ \vdots \\ y_{n_R}(n) \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1N} \\ h_{21} & h_{22} & \vdots & h_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n_R1} & h_{n_R2} & \cdots & h_{n_RN} \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \\ \vdots \\ x_{n_T}(n) \end{bmatrix} = \begin{bmatrix} h_{11} \\ h_{21} \\ \vdots \\ h_{n_R1} \end{bmatrix} x_1(n) + \begin{bmatrix} h_{11} \\ h_{21} \\ \vdots \\ h_{n_R1} \end{bmatrix} x_2(n) + \cdots + \begin{bmatrix} h_{1n_T} \\ h_{21} \\ \vdots \\ h_{n_Rn_T} \end{bmatrix} x_{n_T}(n)$$

- Now, each transmit signal is represented by a M-dimensional vector and they have different directions.

- The optimum detector is the maximum likelihood (ML) detector:

$$p(\mathbf{x} \mid \mathbf{y}) \propto \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{y}-\mathbf{H}\mathbf{x}\right\|^2\right)$$

$$\mathbf{y}{=}\mathbf{H}\mathbf{x}{+}\mathbf{w} \Rightarrow \min_{\mathbf{x}}\left\|\mathbf{y}-\mathbf{H}\mathbf{x}\right\|^2$$

- Observations:
  - The computational complexity grows exponentially with the number of antennas.
  - Many suboptimum detectors can be used to approximate the ML.

- Zero-forcing (ZF) detector (de-correllator) :

$$\mathbf{y}=\mathbf{H}\mathbf{x}+\mathbf{w} \Rightarrow \mathbf{H}^{-1}\mathbf{y}=\mathbf{x}+\mathbf{H}^{-1}\mathbf{w} \Rightarrow \tilde{\mathbf{y}}=\mathbf{x}+\tilde{\mathbf{w}}$$

$$\mathrm{var}(\tilde{w}_1) \sim \frac{|h_{22}|^2+|h_{21}|^2}{|h_{11}h_{22}-h_{21}h_{12}|^2} N_0$$

* Null interference
* Equalize the channel
* 2x2

- The advantage of the ZF detector is that it is easy to implement.
- The disadvantage is that the noise can be significantly amplified, resulting in poor performance.

- MMSE detector:

$$\mathbf{y}=\mathbf{Hx}+\mathbf{w} \Rightarrow \min_{\mathbf{w}} E\left\{\left(\mathbf{x}-\mathbf{Wy}\right)^H\left(\mathbf{x}-\mathbf{Wy}\right)\right\}$$

- The solution:

$$* \rho = \frac{\sigma_x^2}{\sigma_w^2}$$

$$\mathbf{W}=\mathbf{H}^H\left(\mathbf{HH}^H+\rho^{-1}\mathbf{I}\right)^{-1}, \quad \text{if } n_T > n_R$$

$$\mathbf{W}=\left(\mathbf{H}^H\mathbf{H}+\rho^{-1}\mathbf{I}\right)^{-1}\mathbf{H}^H, \quad \text{if } n_T \le n_R$$

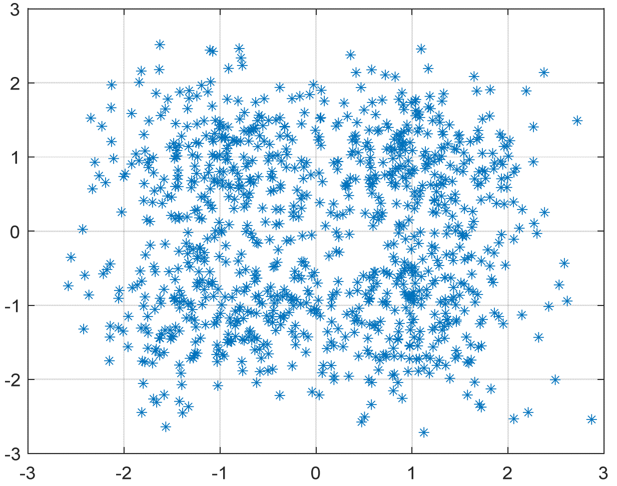
- The SINR:

$$\gamma_k=\frac{1}{\left(\mathbf{I}+\rho\mathbf{H}^H\mathbf{H}\right)^{-1}_{kk}}, \quad \text{if } n_T \le n_R$$

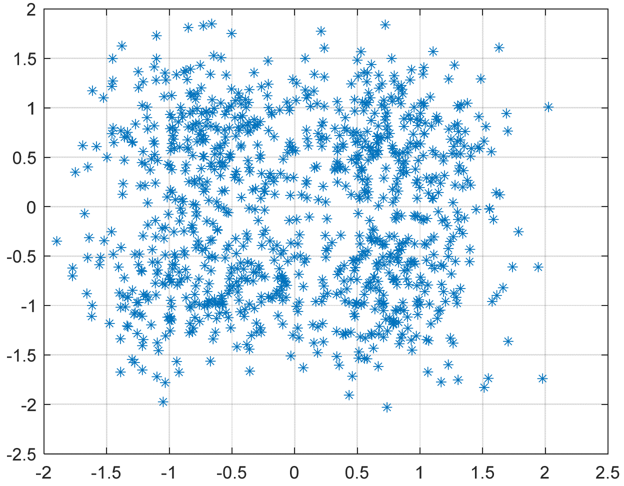
- The performance of the MMSE detector is better than that of the ZF.

- Practice 3:
  - Consider a 2x2 baseband system.
  - Given a sequence and a channel matrix (S13P3.mat), add AWGN with σ=0.3 and conduct the ZF/MMSE detection for the system.
  - Calculate MSE of both detectors.

- Results:



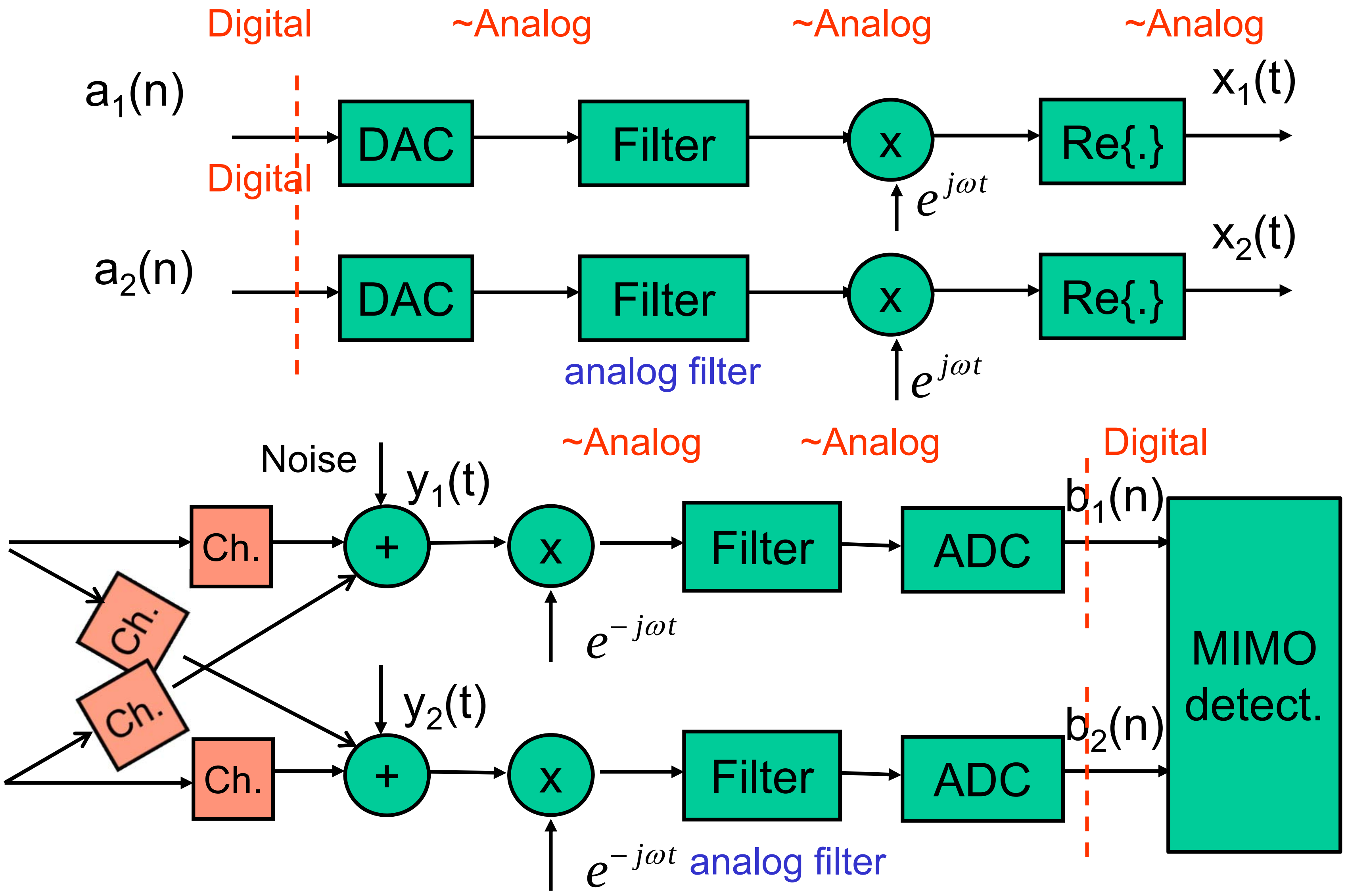
MSE=0.397



MSE=0.285

- Homework:
  - Given two sequences and a 2x2 channel (S13HW.mat), extend the built plateform in Practice 2 to a 2x2 system and conduct ZF detection to recovery transmit symbols.

- 2x2 system:


Digital
~Analog
~Analog
~Analog
$a_1(n)$
DAC
Filter
x
Re{.}
$x_1(t)$
Digital
$e^{j\omega t}$
$x_2(t)$
$a_2(n)$
DAC
Filter
x
Re{.}
analog filter
$e^{j\omega t}$
Noise
$y_1(t)$
~Analog
~Analog
Digital
$b_1(n)$
Ch.
+
x
Filter
ADC
MIMO detect.
Ch.
$e^{-j\omega t}$
Ch.
$y_2(t)$
$b_2(n)$
Ch.
+
x
Filter
ADC
$e^{-j\omega t}$
analog filter

Lab. 14 Fixed-point Implementation

- Interface to real-world: DAC/ADC
- DAC: time→ discrete to continuous, value→ quantization
- ADC: time→ continuous to discrete, value→ quantization
- The precision of the DAC/ADC (number of bits for quantization) directly impacts the system complexity.
- Floating-point/fixed-point representation of a number:

Decimal point : depends on the exponent

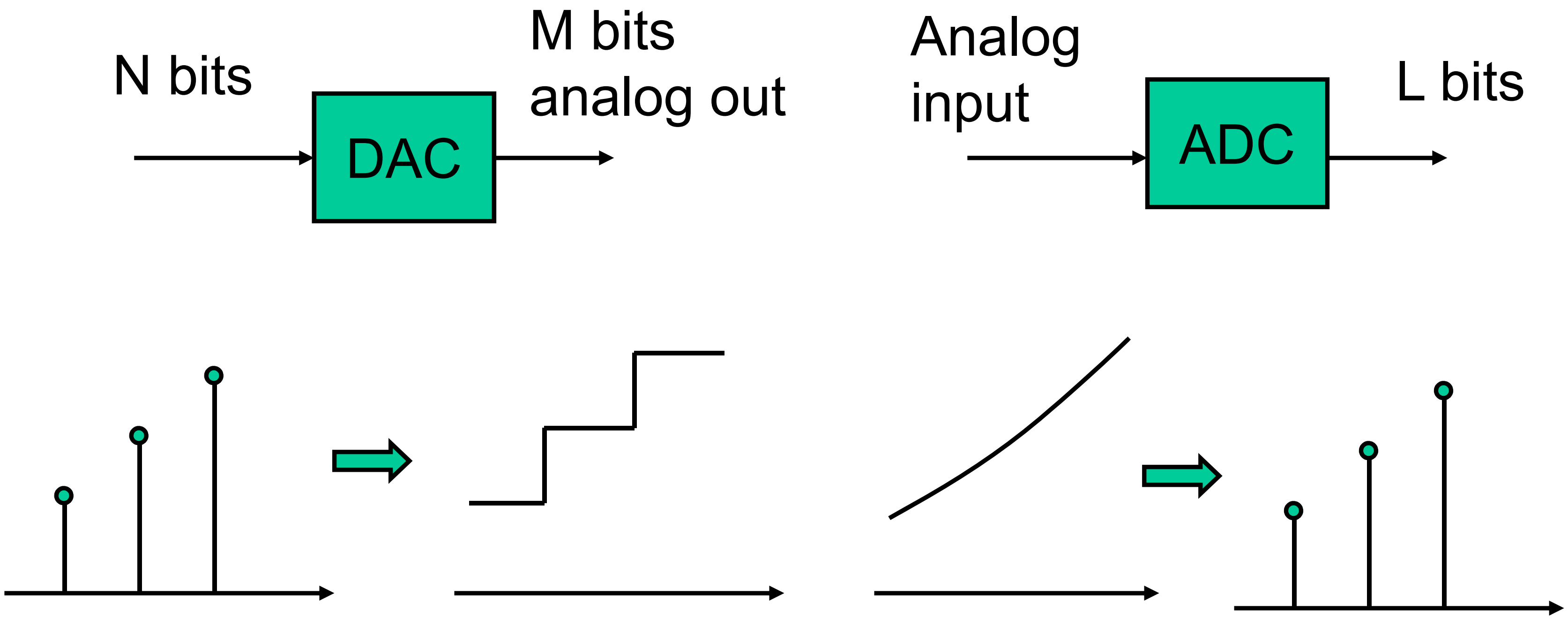
Floating-point: 1.2134 x 10^4
Fixed-point: 12134.1

Decimal point : fixed

* F.P. : For a signal, its decimal point can be changed dynamically.

- Almost all real-world communication systems use fixed-point implementation (d.p. cannot be change dynamically).

- DAC/ADC:



N bits

DAC

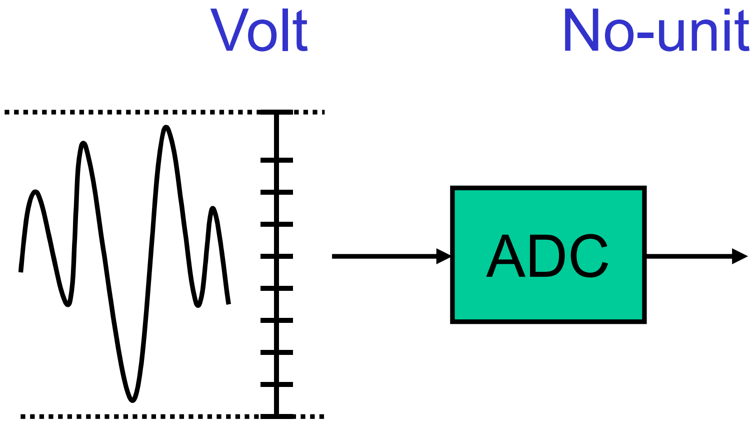
M bits
analog out

Analog
input

ADC

L bits

- Parameters of DAC/ADC:
  - Sampling rate
  - Number of quantization levels
  - Dynamic range

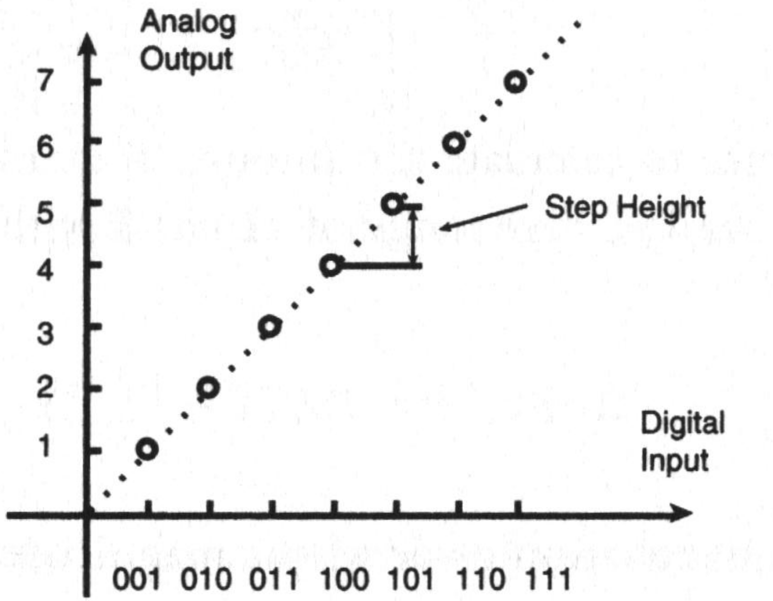


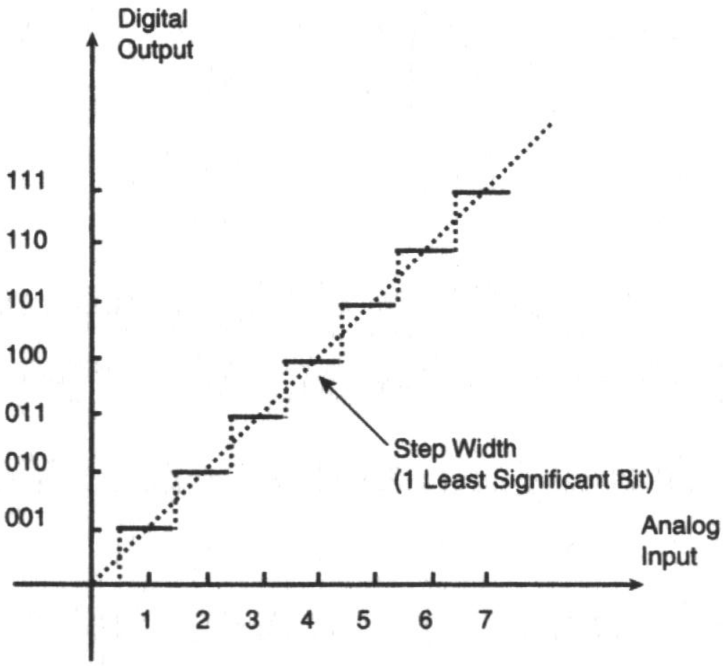
Volt

No-unit

ADC

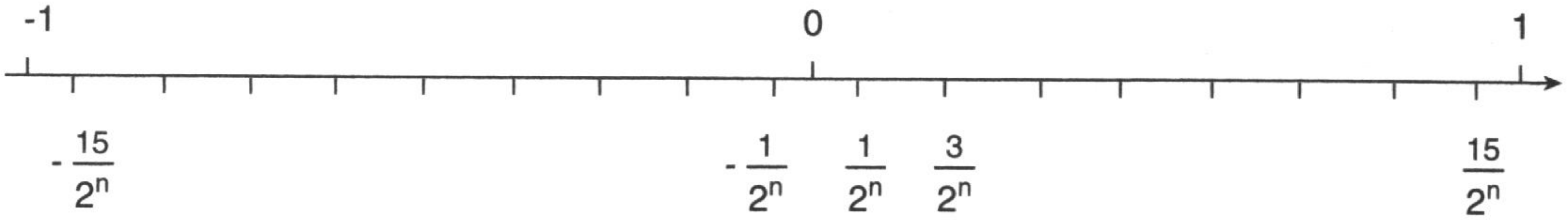
- Ideal DAC and ADC:





- Calculation of quantization noise:

$$\text{Modeling}: \underbrace{\bar{x}}_{\text{quantized}} = \underbrace{x}_{\text{original}} + \underbrace{w}_{\text{noise}}$$



- An N-bit ADC can have a quantization error from -1/2^N to 1/2^N (2/2^{N}x1/2).
- The average quantization power is (uniformly distributed)

$$\sigma_q^2 = \frac{1}{\frac{1}{2^N} - \left(-\frac{1}{2^N}\right)} \int_{-\frac{1}{2^N}}^{\frac{1}{2^N}} x^2 dx = \left.\frac{2^{N-1}}{3}\right|_{-\frac{1}{2^N}}^{\frac{1}{2^N}} = \frac{1}{3}\left[\frac{1}{2^N}\right]^2$$

- Given that the dynamic range of ADC and DAC is between -1 and 1, then the quantization noise power is

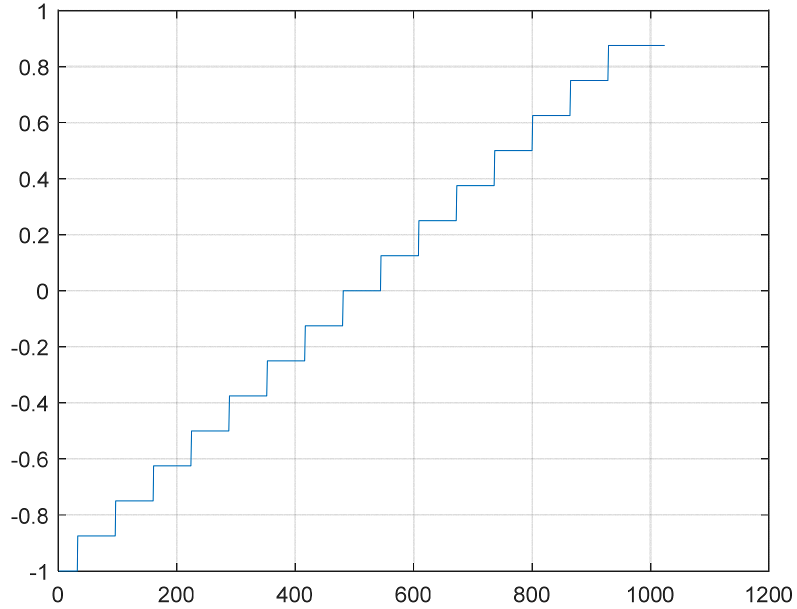
$$\sigma_q^2 = -10\times\log_{10}3 - 20\times N\times\log_{10}2 = \boxed{-4.77 - 6.02N \text{ dB}}$$

- Note that the quantization noise of the ADC should not becomes the main noise source in the system.

- Practice 1:
  - Given a ramp-up signal (S14P1.mat), conduct the ADC operations (number of bits: 4, dynamic range [-1,1]), and observe the output.
  - Given an uniformly distributed signal, conduct ADC operations (nb: 4, dr: [-1,1]) and calculate signal-to-quantization-noise-ratio (SQNR).
  - Verify the 6dB rule of thumb (theoretical SQNR).
  - Rewrite the program as a function (inputs: the value to be quantized, the number of bits, the dynamic range, output: the quantized result).
  - Given a white Gaussian signal, conduct ADC operation (nb: 7, dr: 3), and calculate the SQNR again.

$$\mathbf{SQNR} = 10\log_{10}\frac{E\left\{x^2(n)\right\}}{E\left\{\ [x(n)-x_Q(n)]^2\right\}}$$

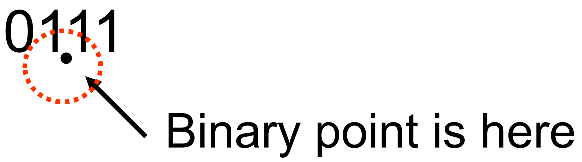
- Result:



SQNR_{th}=24.0788dB   SQNR=23.4147dB

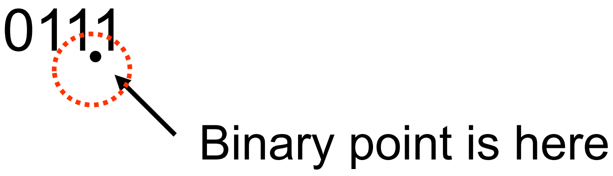
SQNR=37.3483

- Fixed-point representation
  - Dynamic range (DR)→ the position of decimal point
  - Number of quantization level (NQL) → the number of bits (NOB) to store the signal.
- For example: DR=±2, NQL=16 vs. DR= ±4, NQL=16



01.11

Binary point is here



011.1

Binary point is here

$$\textbf{Max}: \quad 0111 \rightarrow 1+\frac{1}{2}+\frac{1}{4}=1\frac{3}{4}$$

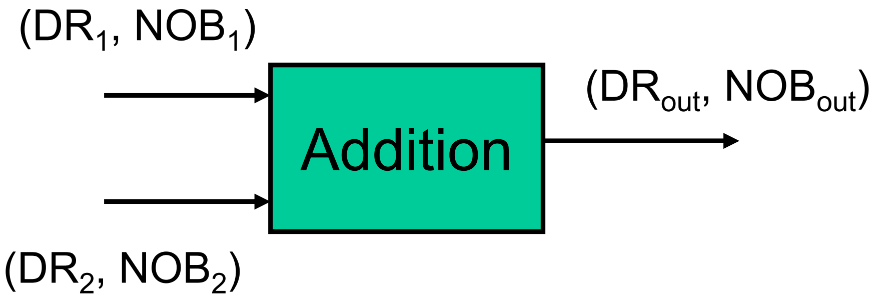
$$\textbf{Min}: \quad 1000 \rightarrow 0111+1 \rightarrow -2$$

$$\textbf{Max}: \quad 0111 \rightarrow 2+1+\frac{1}{2}=3\frac{1}{2}$$

$$\textbf{Min}: \quad 1000 \rightarrow 0111+1 \rightarrow -4$$

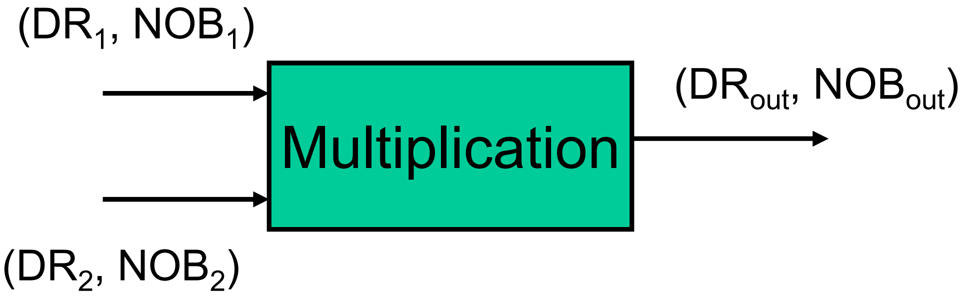
- Fixed-point operations:
  - In real circuits, addition and multiplication are two most common operations implemented.
  - What happened when two fixed-point numbers are added/ multiplied?

- Addition:



  - In general, $DR_{out}=DR_1+DR_2$ and $NOB_{out} << NOB_1+NOB_2$ (depends on DR_1 and DR_2 and output requirement)

- Multiplication:

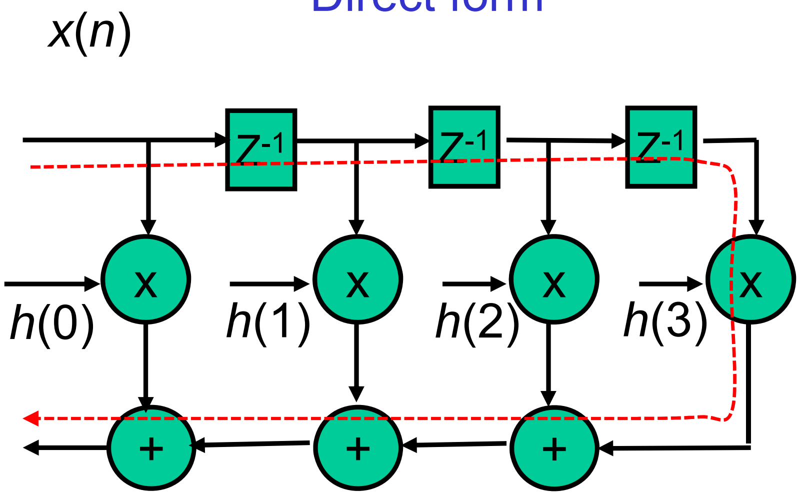


  - In general, $DR_{out}=DR_1 x DR_2$ and $NOB_{out}<NOB_1+NOB_2$ (depends on output requirement)

- Note that before conducting fixed-point simulations, the architecture of the operation has to be defined.
- For example: the filtering operation

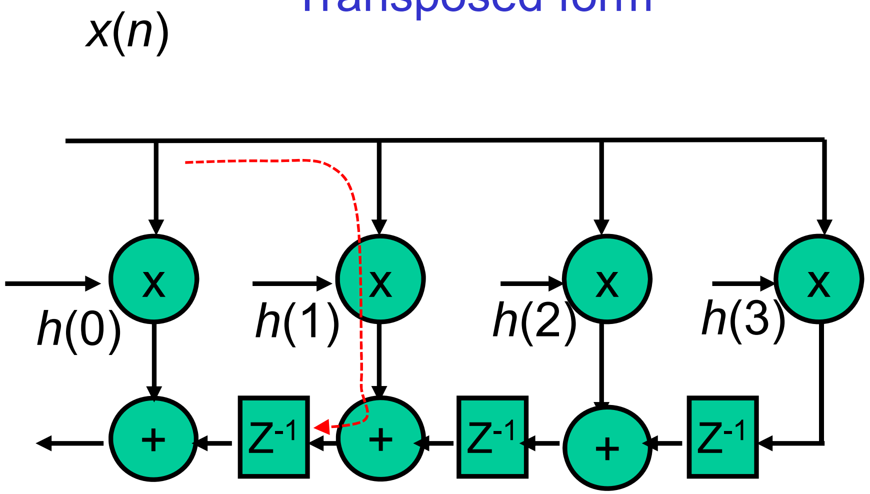
$$y(n) = \sum_{i=0}^{3} h(i)x(n-i)$$

Direct form



Critical path: 1 multiplier and 3 adders

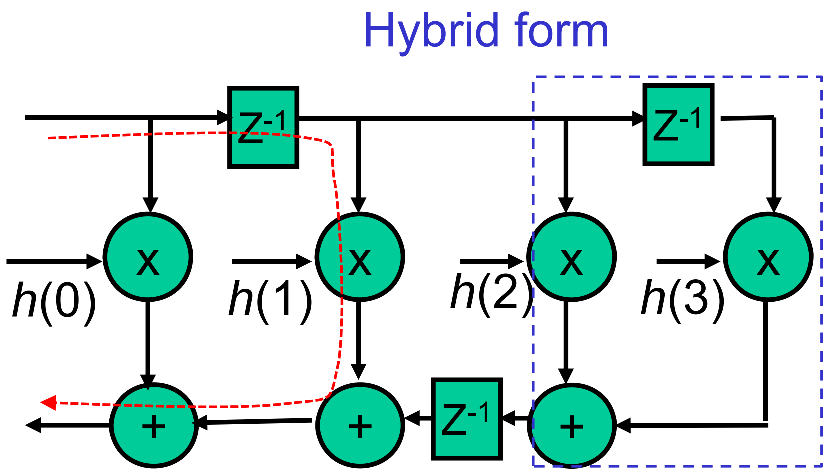
Transposed form



Critical path: 1 multiplier and 1 adder

Fixed-point simulation: determining the number of bits to represent each signal

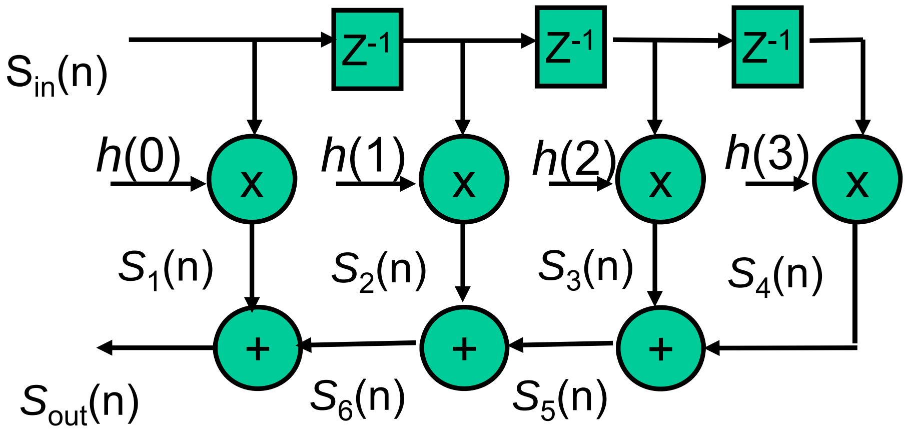
- Another architecture (hybrid form):



Critical path: 1 multiplier and 2 adders

- To conduct fixed-point simulations, the DR and the NOB for each signal has to be determined.
- The procedure:
  - Conduct floating-point simulations to have the knowledge of the distribution of each signal (histogram).
  - Determine the DR of each signal.
  - Choose the number of bits for each signal.

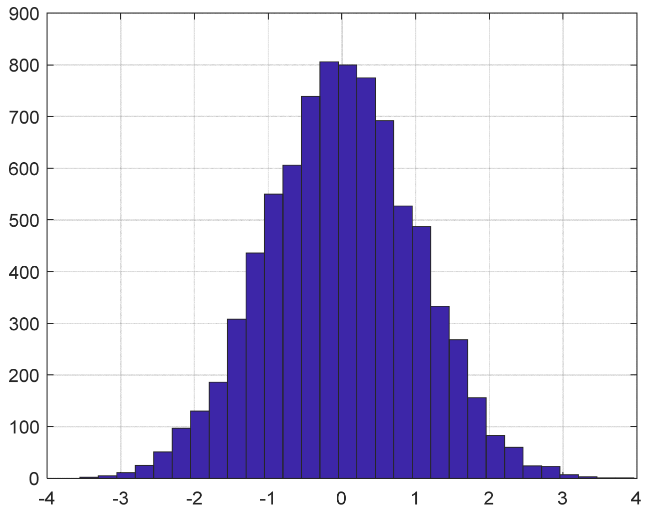
- Practice 2:
  - Given a filter with four coefficients, use the direct form architecture to conduct floating-point simulations, and observe the DR of each signal with histograms (S_{in}(.): white Gaussian with variance one).



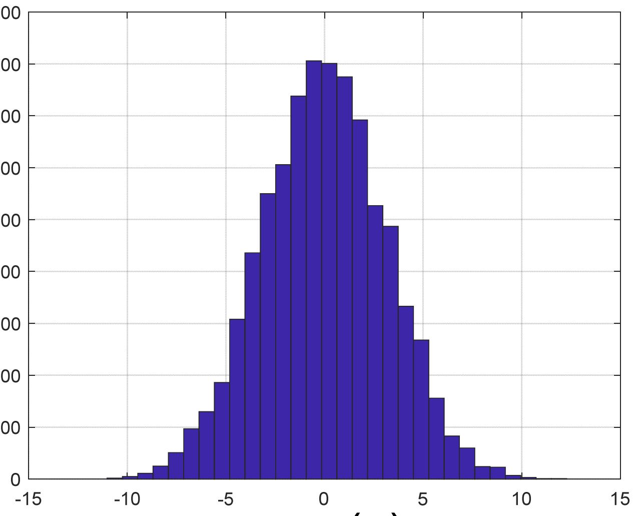
h(n)=[0.82, 3.1, 1.2, 0.53]

* 8 signals

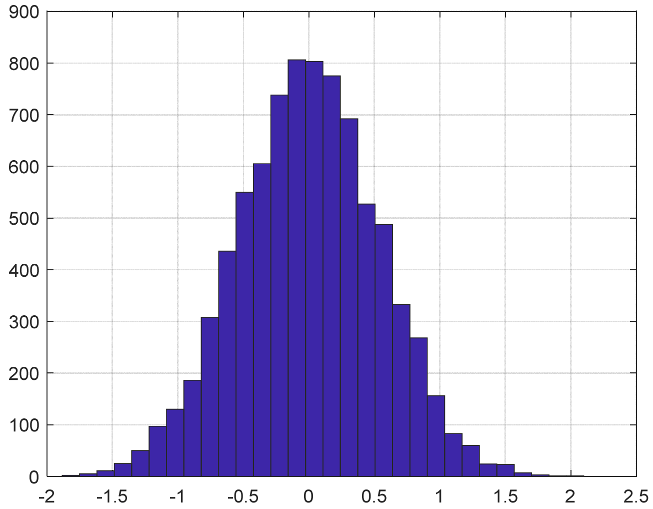
- Result:



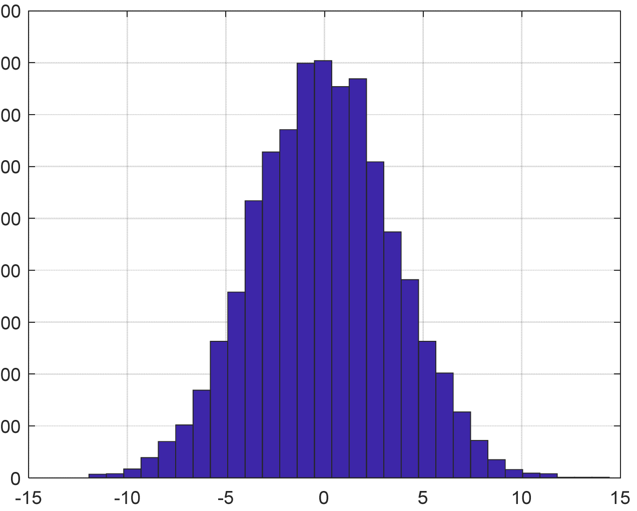
$s_{in}(n)$



$s_2(n)$



$s_4(n)$



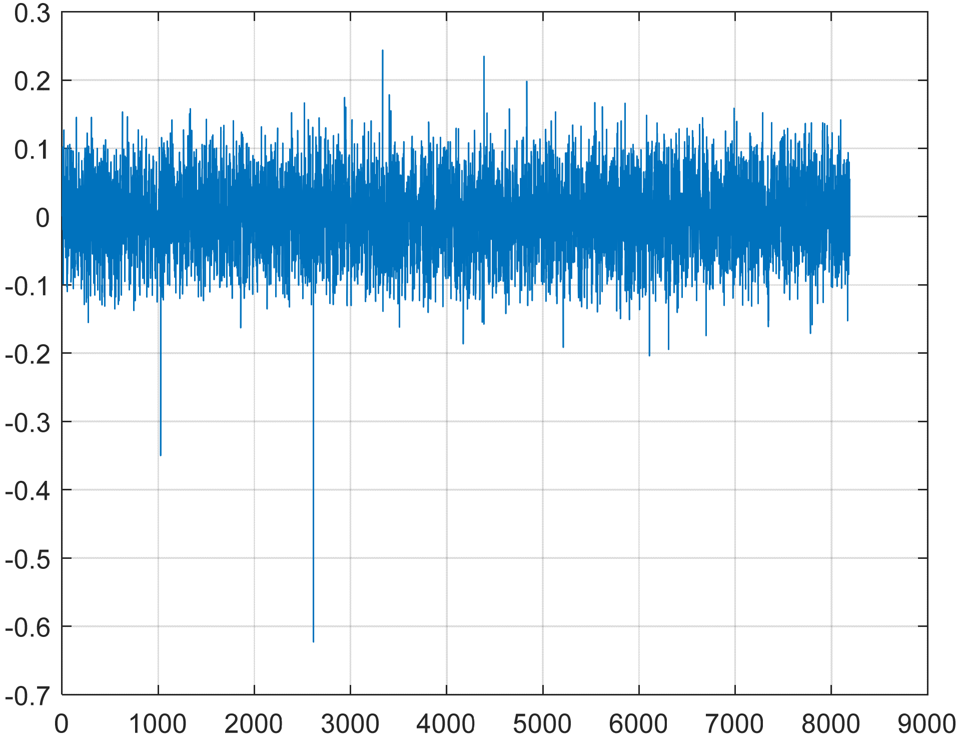
$s_{out}(n)$

- Practice 3:
  - Quantize the coefficients and all signals by properly choosing NOBs and DRs (note: has to be 2^n).
    - Let the input be quantized with NOB=8 and DR=4
    - The required SQNR at out is 35 dB
    - The goal is to meet the SQNR requirement with the least number of bits used in signals
  - Conduct fixed-point simulations for the filter and calculate the SQNR.

* If the DR is between -2^{n-1} and 2^{n-1}, the quantized result can be mapped the binary representation directly.

* With chosen DR, some signal may be clipped!

- Result:

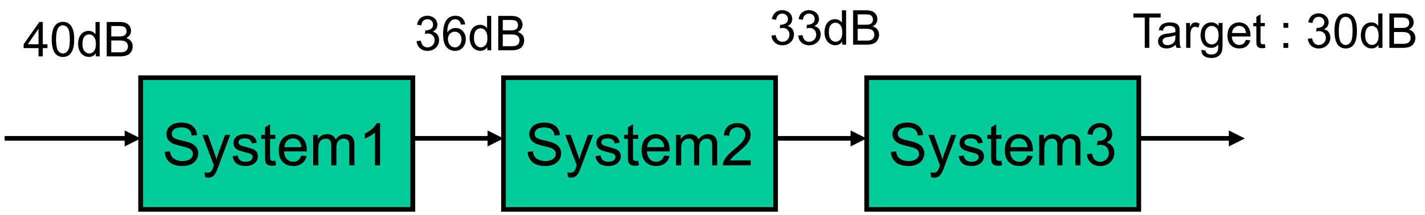


$s_{out}(n)$-Q[$s_{out}(n)$]

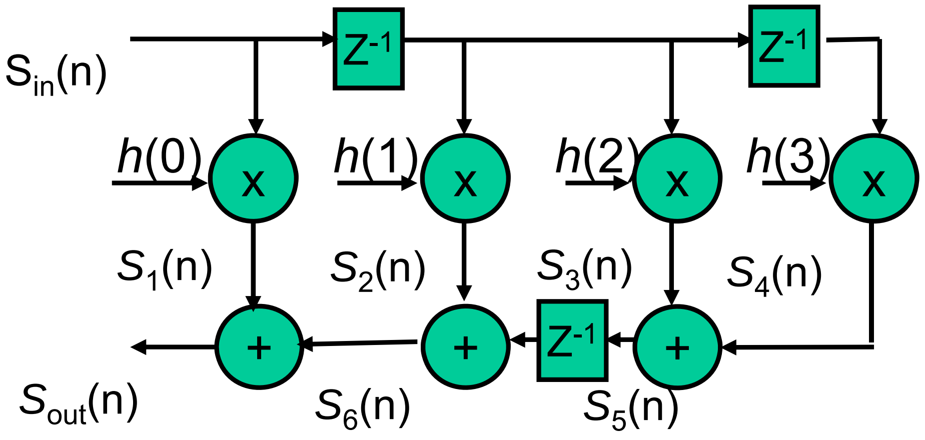
SQNR=35.7626dB

- For a systems with many blocks, how to determine the DR and NOB for each block ?
  - Determine the performance target of the system (MSE, SNR, or BER according to floating-point simulations or specifications).
  - Determine the acceptable performance loss (or acceptable target) of the fixed-point system.
  - Distribute the loss to each subsystem.
  - For each subsystem, we can then determine the NOBs of the signals and parameters inside the subsystem.

- e.g.

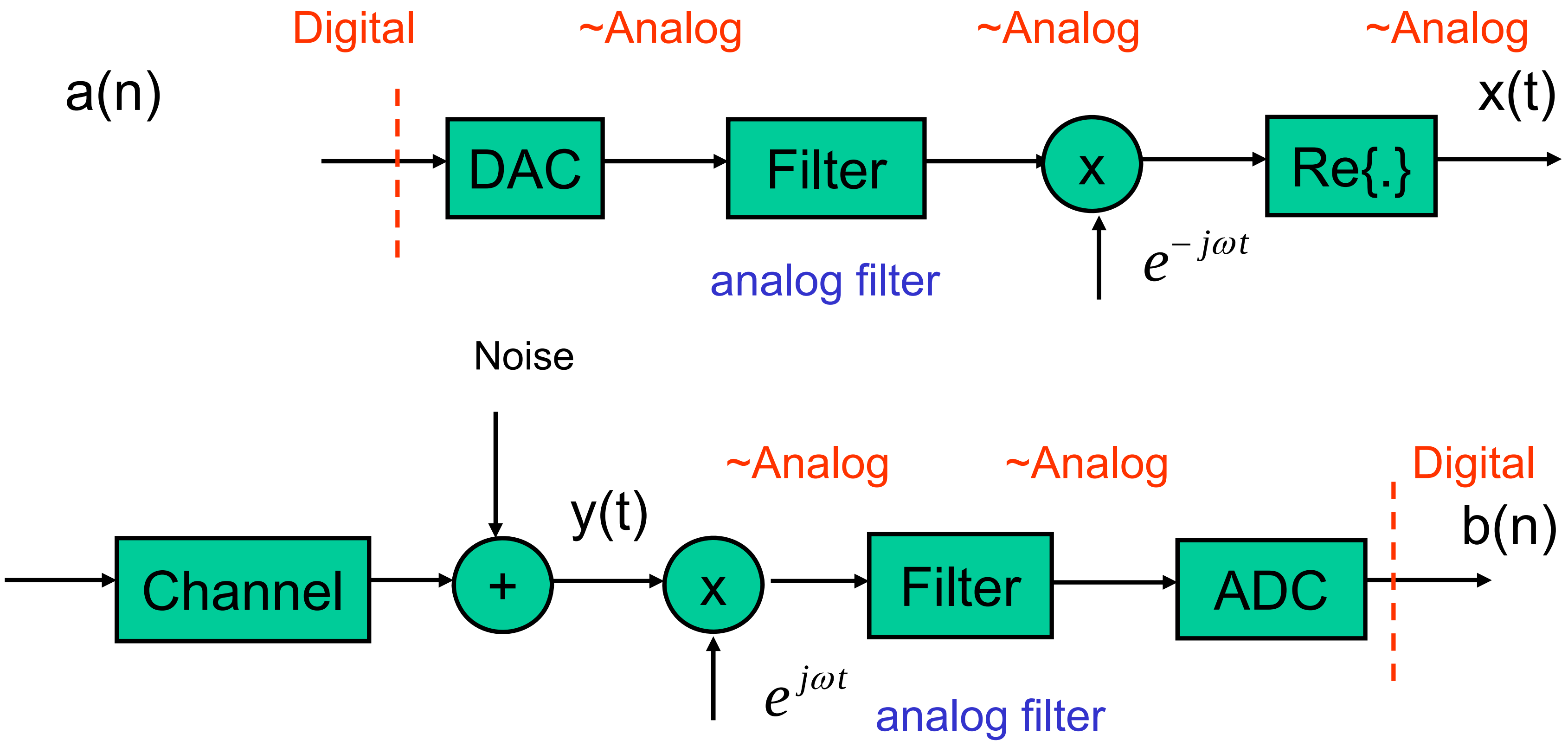


- Homework:
  - With the filter coefficients given in Practice 2 and the hybrid structure given below, conduct floating-point simulations first to determine the DR of each signal and then fixed-point simulations to determine the NOB for each signal.
    - Let the input be quantized with NOB=8 and DR=4.
    - The required SQNR at out is 35 dB.
    - The goal is to meet the SQNR requirement with the least number of bits used in signals.



Lab. 15 Testing

- A digital communication system:



- Analog/RF Impairments:
  - DAC
  - Transmit IQ imbalance
  - Phase noise of the mixer
  - PA nonlinearity
  - Channel effect
  - Noise
  - Receive IQ imbalance
  - DC offset
  - Phase noise of the mixer
  - ADC
  - Carrier frequency offset
- Digital processing:
  - Precision (fixed-point processing)
  - Receiver algorithms

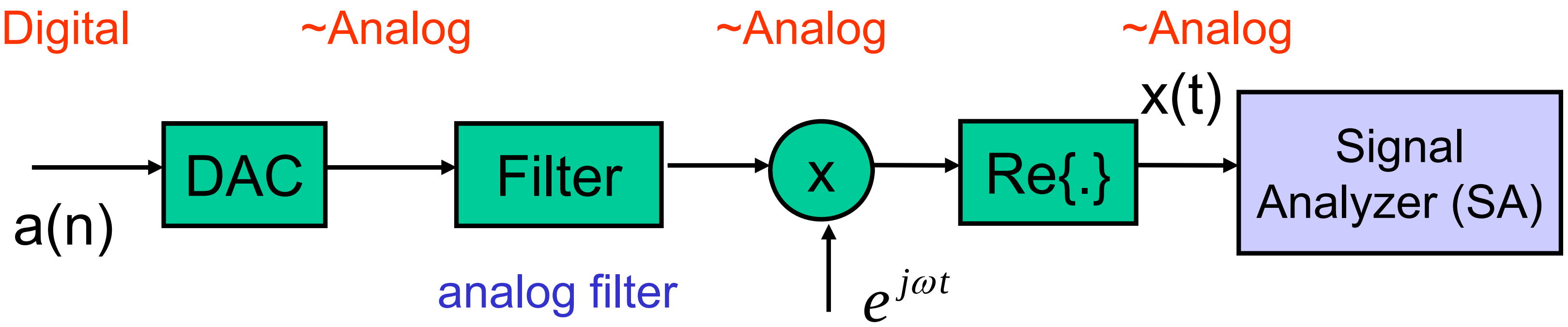
- How to test a communication system?
  - Transmitter/receiver performance (separated)
  - Combine all impairments (combined)
- Transmitter:
  - Transmit signal distortion (for receiver)
  - Spectrum characteristic (for interference control)
- Indices for transmitter
  - Error vector magnitude (EVM)
  - Spectrum mask

* EVM is usually used for QAM signals.

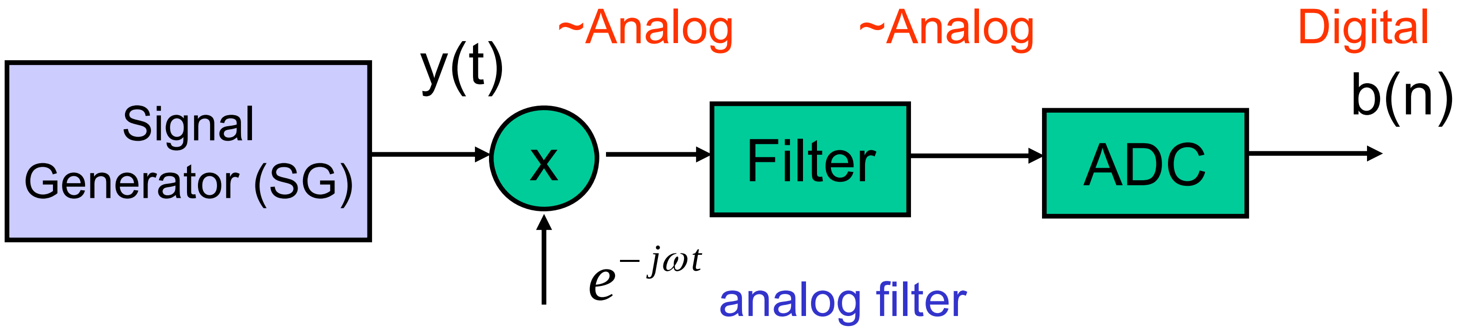
- Index for receiver:
  - Sensitivity
  - Interference resistance performance

- Test equipment:

Transmitter test:



Receiver test:



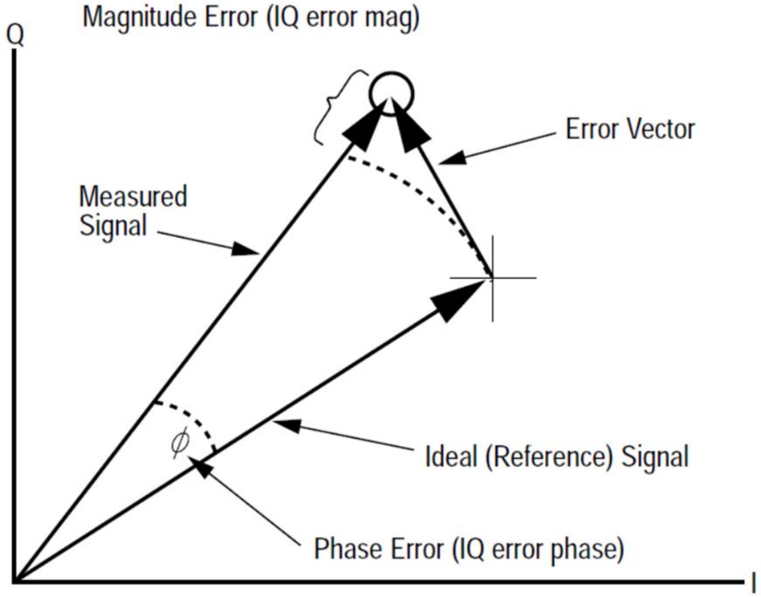
- Case study: Bluetooth 1.0
  - Band : 2.4GHz
  - Bandwidth: 2MHz
  - Data rate : 1Mb/s
  - Modulation: GFSK
- Parameters for Bluetooth
  - Modulation index: $0.28 < h < 0.35$
  - Standard deviation of Gaussian filter: BT=0.5
- Functions to be conducted by transmitter :
  - NRZ signal generation
  - Gaussian filtering
  - Phase integration
  - IF modulation
  - Filtering (fit spectrum mask)

- Function to be conducted by receiver:
  - Automatic gain control (AGC)
  - IF down converting
  - Gaussian filtering
  - Phase extraction/differentiation
  - Packet detection
  - Symbol timing
  - Carrier frequency offset (CFO) estimation/correction
  - Sampling frequency offset/timing correction
  - Detection

- EVM:
  - A signal sent by an ideal transmitter would have all constellation point at the ideal locations
  - For real-world transmitter, however, the constellation points will deviate from the ideal locations in a random fashion.
  - EVM is a measure of how far the constellation points are from the ideal locations

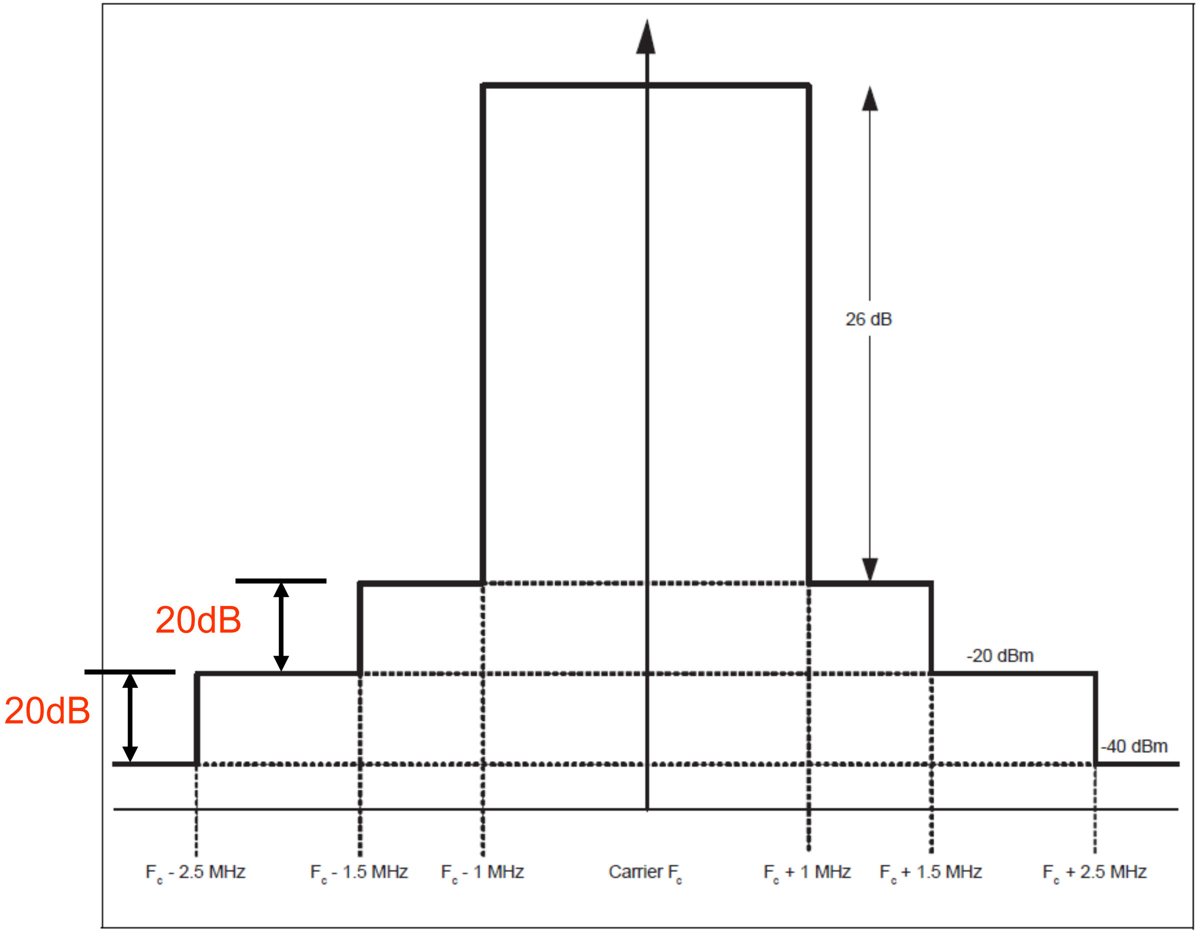
$$\mathbf{EVM(dB)} = 10\log_{10}\frac{E\left\{\left|a(n)-\hat{a}(n)\right|^2\right\}}{E\left\{\left|a(n)\right|^2\right\}}$$

$$\mathbf{EVM(\%)} = \sqrt{\frac{E\left\{\left|a(n)-\hat{a}(n)\right|^2\right\}}{E\left\{\left|a(n)\right|^2\right\}}}\times 100\%$$



* -30dB = 3.16%
* -40dB = 1.00%

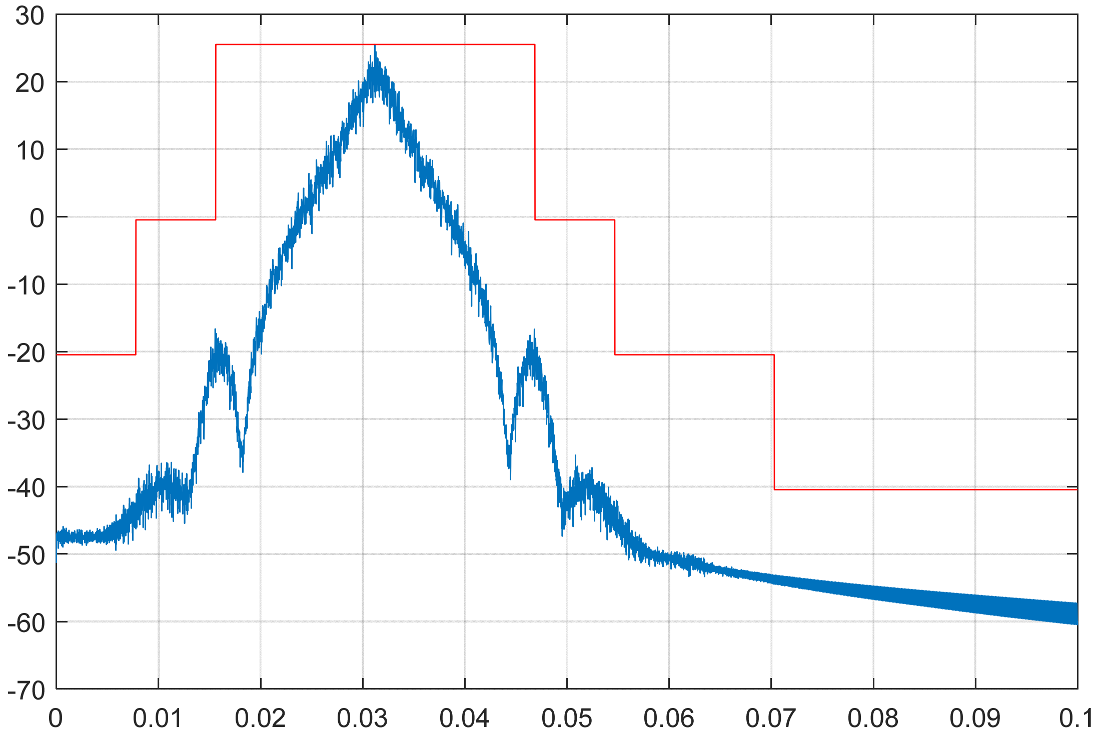
- Spectrum mask:
  - Confine the spectrum used for transmission (interference control for the Bluetooth system)



- Practice 1:
  - Use the GFSK system developed in S12P2 as a Bluetooth system.
  - Generate a Bluetooth transmission signal and check if its spectrum fits the requirement of the spectrum mask (write a function to plot the spectrum mask and the transmit signal spectrum).

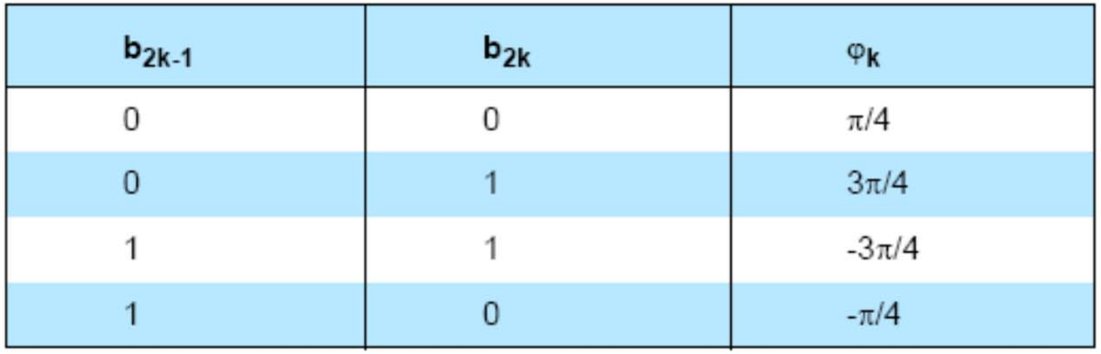
* See spectrum: [px,f]=pwelch(x,[],[],[],1);plot(f,10*log10(px));

- Result:

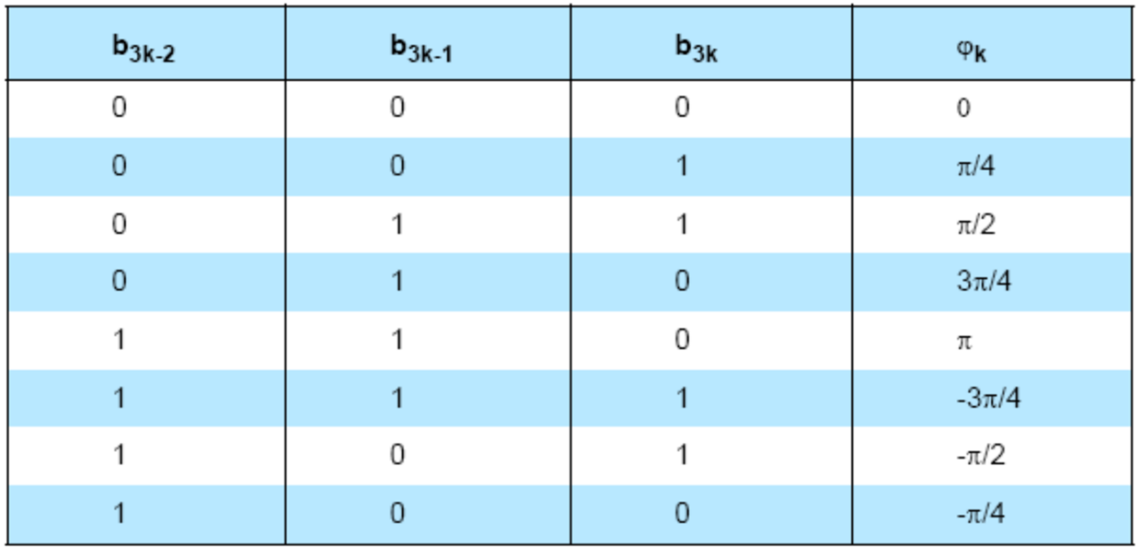


30
20
10
0
-10
-20
-30
-40
-50
-60
-70
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1

- Bluetooth EDR (enhanced data rate, 2Mbps and 3Mbps)
- Modulation:
  - $\pi/4$ DPSK
  - 8PSK

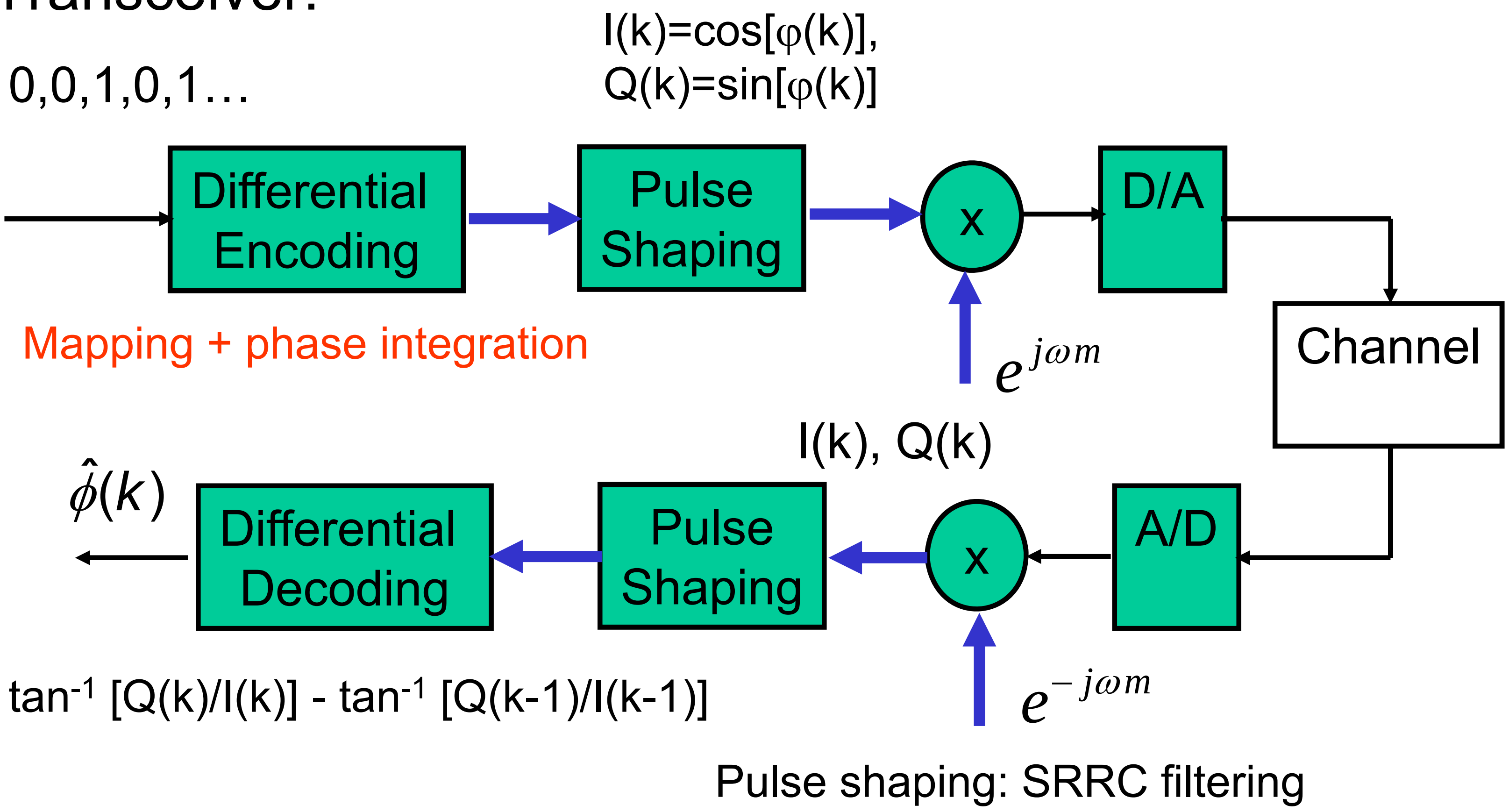


b_{2k-1}	b_{2k}	φ_k
0	0	$\pi/4$
0	1	$3\pi/4$
1	1	$-3\pi/4$
1	0	$-\pi/4$



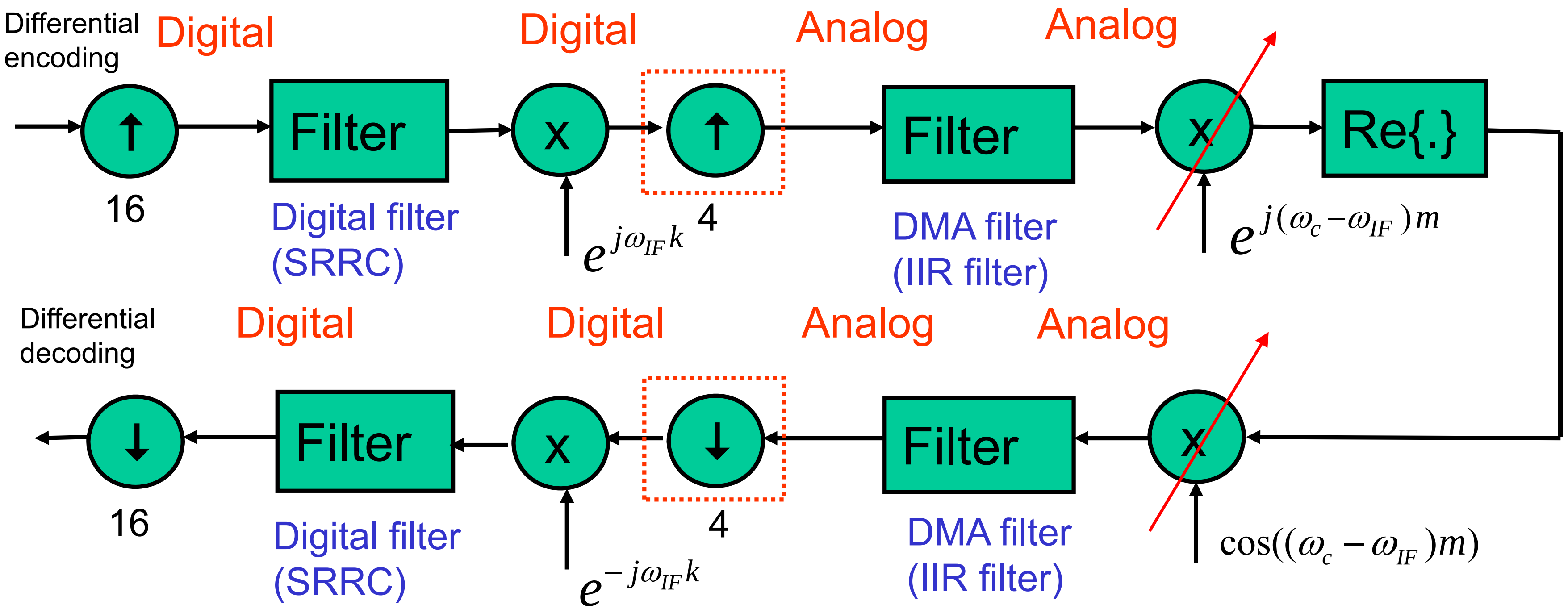
b_{3k-2}	b_{3k-1}	b_{3k}	φ_k
0	0	0	0
0	0	1	$\pi/4$
0	1	1	$\pi/2$
0	1	0	$3\pi/4$
1	1	0	π
1	1	1	$-3\pi/4$
1	0	1	$-\pi/2$
1	0	0	$-\pi/4$

- Pulse shaping filter:
  - Squared root raised cosine (SRRC) filter
  - Roll-off factor=0.4
- Transceiver:

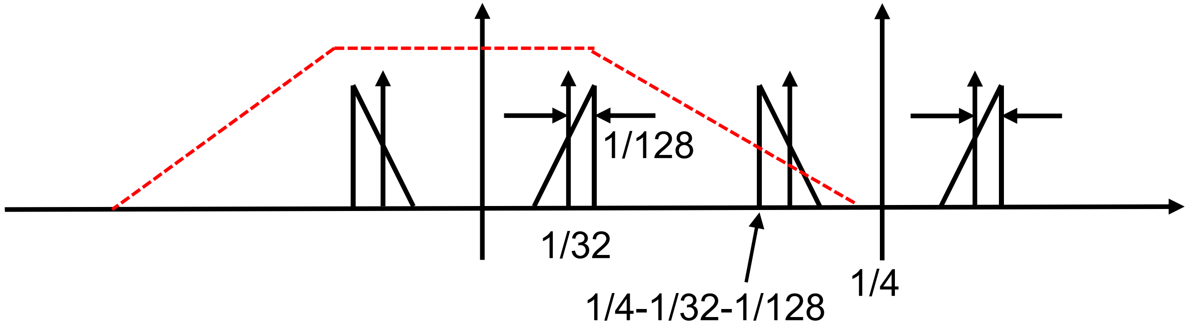


- Practice 2:
  - Design and implement a Bluetooth EDR system for the 2Mpbs mode (parameters are same as those in S12P2).
    - IF architecture/ Symbol rate: 1M/ IF frequency: 2M/ Tx oversampling rate: 16/ Analog signal sampling rate: 64
  - Check the spectrum mask.
  - Calculate the EVM.

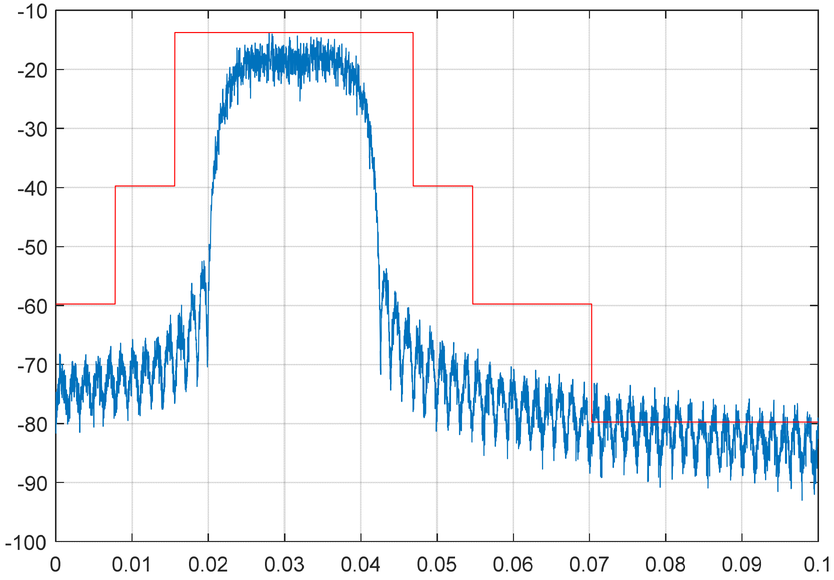
* Ignore analog carrier

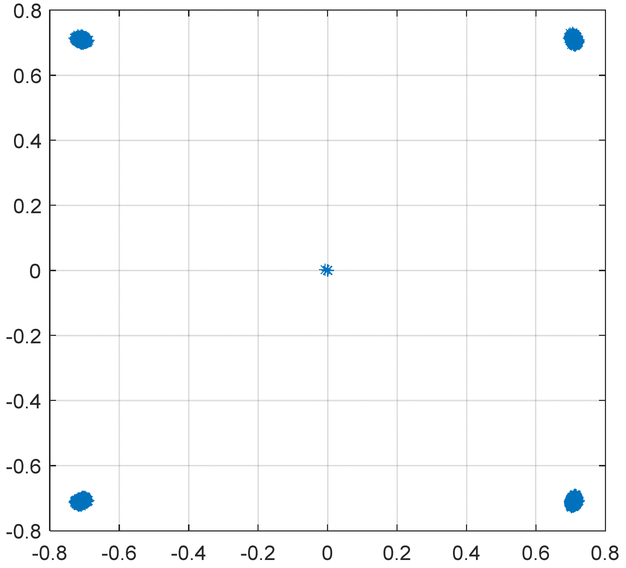


- Result:



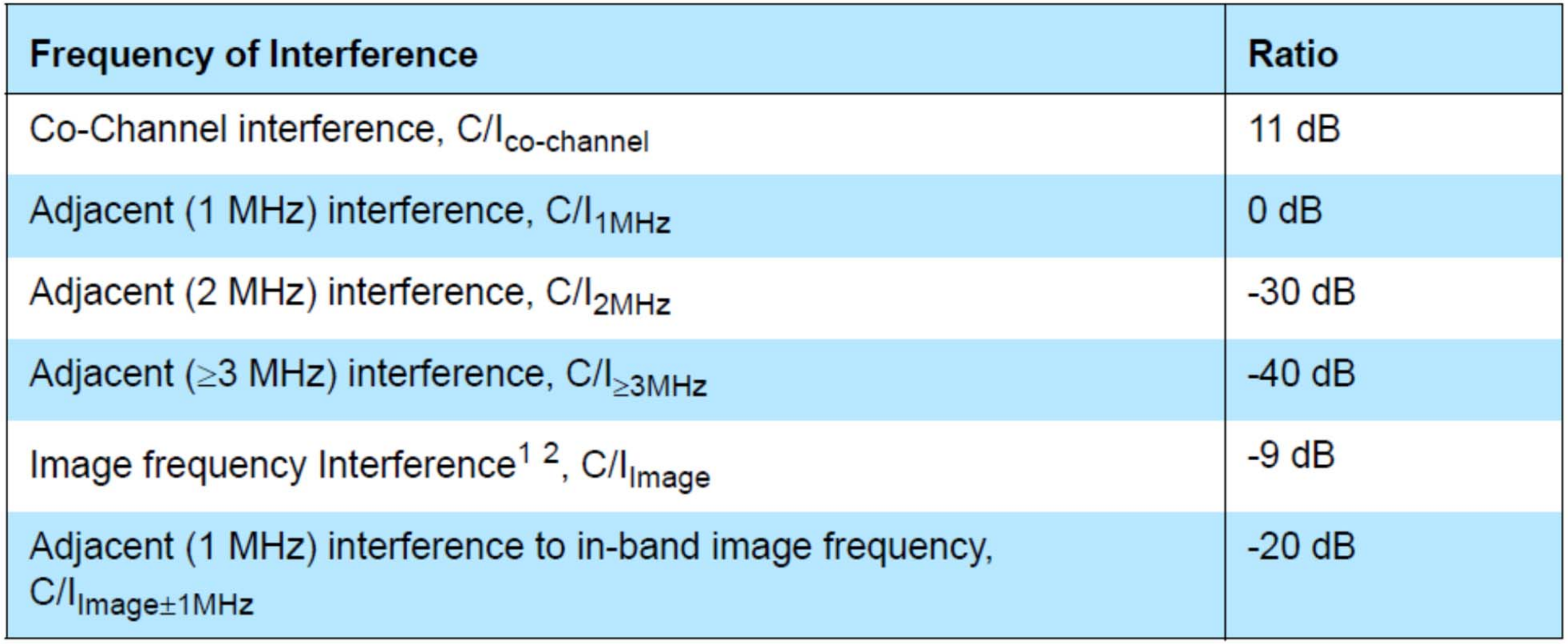
EVM=-41dB





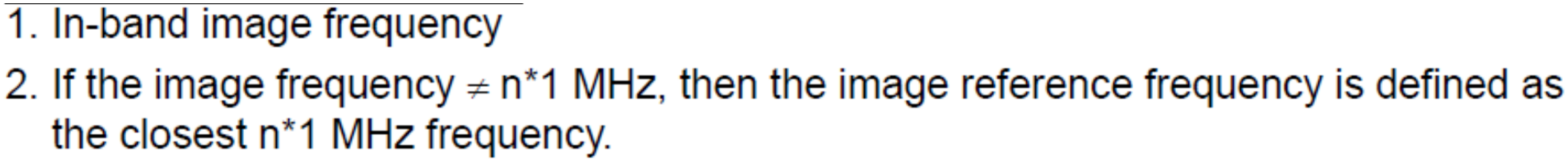
- Interference performance:

C: carrier/signal



Frequency of Interference	Ratio
Co-Channel interference, $C/I_{co\text{-}channel}$	11 dB
Adjacent (1 MHz) interference, C/I_{1MHz}	0 dB
Adjacent (2 MHz) interference, C/I_{2MHz}	-30 dB
Adjacent (≥3 MHz) interference, $C/I_{\geq 3MHz}$	-40 dB
Image frequency Interference[1 2], C/I_{Image}	-9 dB
Adjacent (1 MHz) interference to in-band image frequency, $C/I_{Image\pm 1MHz}$	-20 dB

* The BER should be ≤0.1% under the specified interference condition



1. In-band image frequency
2. If the image frequency ≠ n*1 MHz, then the image reference frequency is defined as the closest n*1 MHz frequency.

- Sensitivity: the lowest receive signal power level such that the designed target BER can be met.

$$\text{dBm} = 10 \times \log 10\left(Watts \times 1000\right)$$

- Requirements for BT systems:
  - GFSK: BER=10^{-3}, sensitivity=-70dBm
  - 4DPSK: BER=10^{-4}, sensitivity=-70dBm
  - 8DPSK: BER=$10^{-4,}$ sensitivity=-70dBm

- Practice 3:
  - Add AWGN to the system in Practice 1 yielding an SNR of 5 dB (after DAC). Calculate the BER.

- Homework:
  - Add AWGN to the system in Practice 2 yielding an SNR of -2dB (after DAC). Calculate the SER (symbols before differential encoding).