ECM 9043:

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?

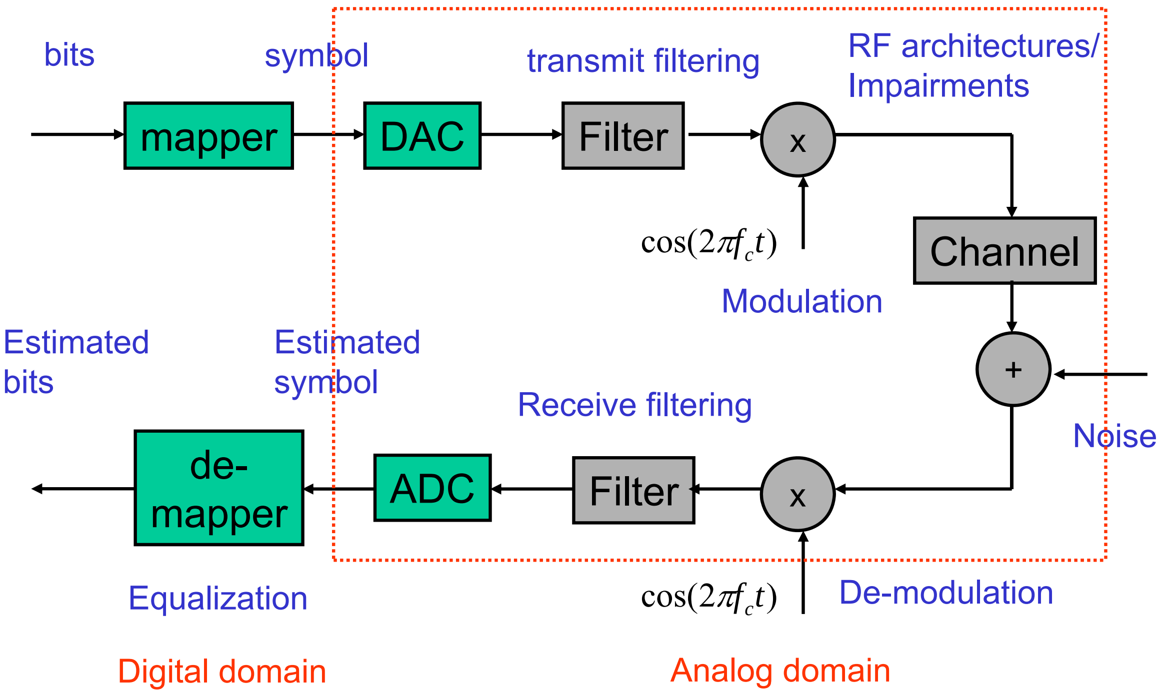
<一知半解>
<支離破碎>

- Do you know?
  - What is difference between “digital” and “analog” signal?
  - 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 communications
  - Provide you simulation skills

- Digital communication system:



- Three domains: digital, analog, and RF

- In the course, you are supposed to implement some typical communication systems with computer programs.
- Programming language – Matlab
- How to achieve this?
  - Quizs
  - In-class practice
  - Homework
  - Midterm/final
- **Instructor**: 吳文榕, ED 830, Phone ext. 31647, e-mail: wrwu@faculty.nctu.edu.tw
- 葉冠華/吳軒毅: ED 821, Phone ext.: 54566, e-mail: jackywu8605@gmail.com, zzzkkzzz9879@yahoo.com.tw

- Grading policy:
  - In-class practice: 20%
  - Homework: 20%
  - Quizs: 10%
  - Midterm/final: 50% (online test)
- Text book: None
- Lecture notes can be bought from the shop in the ED basement.

- 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 test

Format of Report

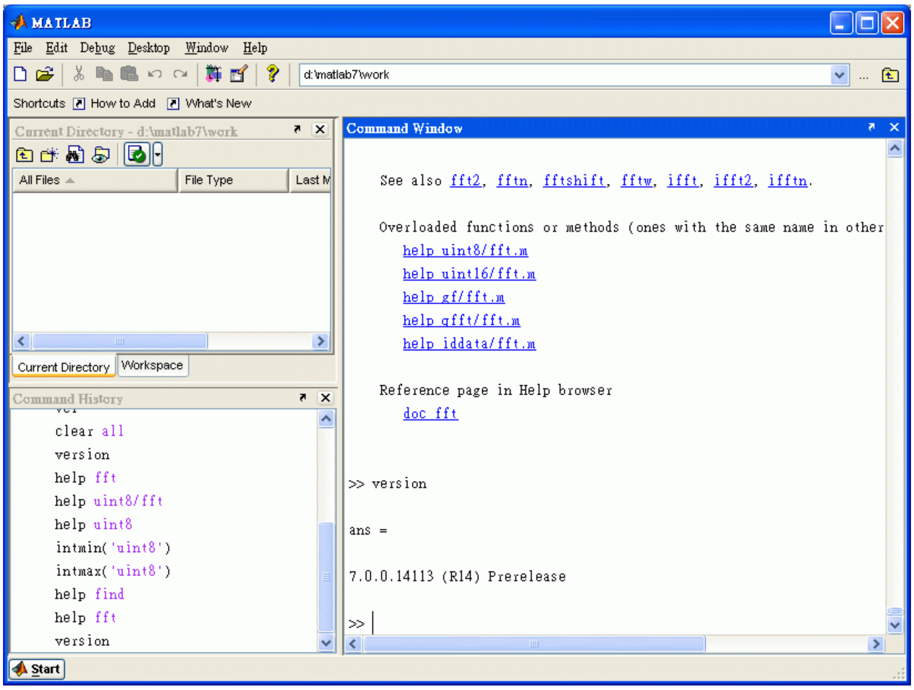
- Problem
  - Problem
  - Background, theories, or ideas
- Simulation results and discussions
  - Simulations results to explain the behaviors of the algorithm or method investigated (table or figures)
  - Discussions
- References
- Appendix

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 (1)

- Initiation:



- 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
- Continuous signal:
  - e.g. sinusoidal $\cos(2\pi ft)$

Lowest frequency : $f = 0$
Highest freqeuncy: $f = \infty$

- Sampled signal:

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

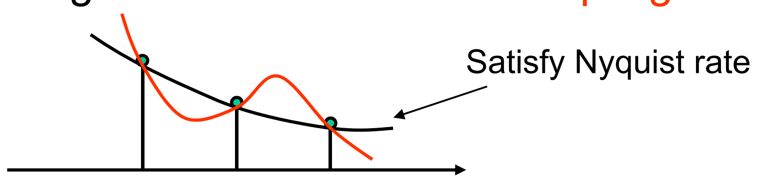
T : samping period

- Discrete sequence:

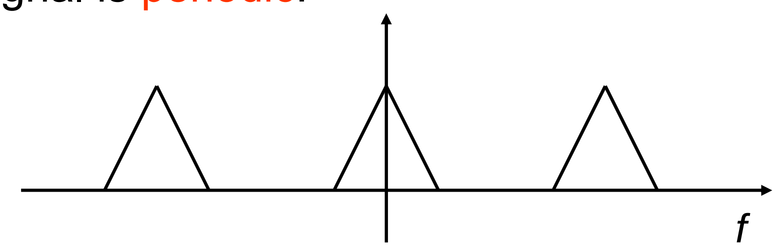
$$\cos(2\pi fn)$$

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.



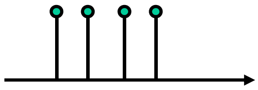
- This is also the reason why the spectrum of a discrete-time signal is periodic.



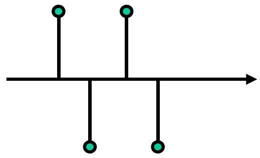
- Continuous and discrete frequency:

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

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



$f=0$



The maximum frequency:

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

- Note that $f=\frac{1}{2}\rightarrow\omega=\pi$

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

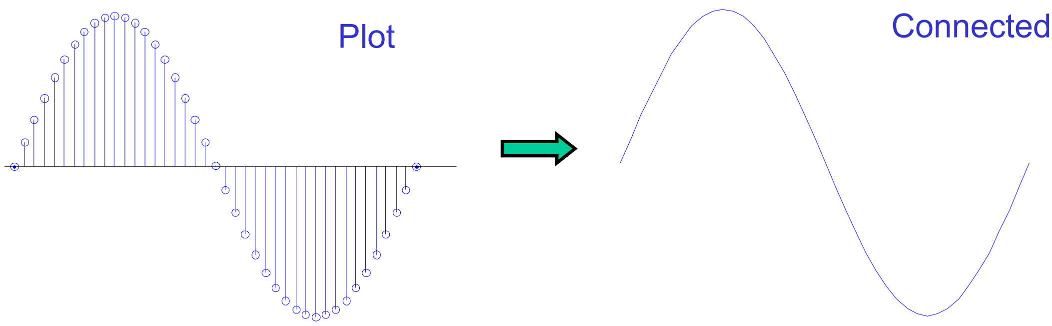
  * Also $\cos(2\pi fnT)=\cos(2\pi\frac{f}{f_s}n)$

- Also, $\cos(2\pi fn)=\cos(2\pi(f\pm1)n)$. Thus

$$f>0,\quad f=1-f$$
$$f<0,\quad f=1+f$$

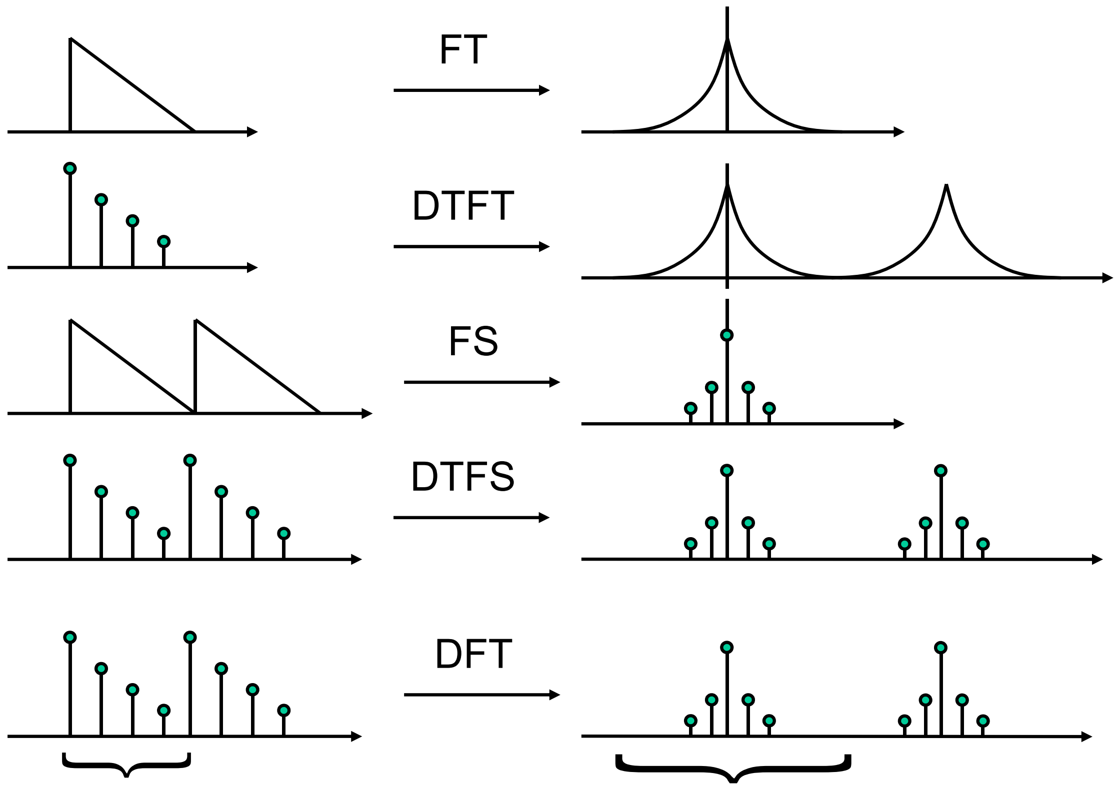
* Effective frequency range [-1/2,1/2]

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

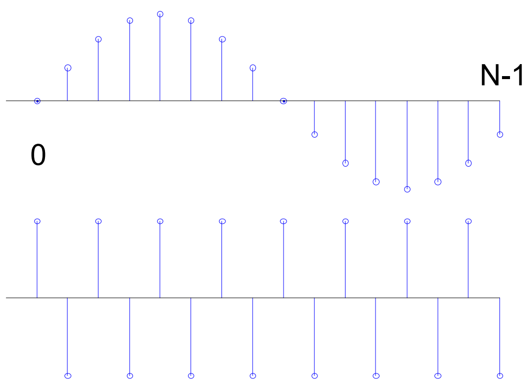


- Practice 1:
  - Generate a sinusoidal signal.
  - Gradually increase/decrease its frequency.
  - Check the maximum frequency, the negative frequency, and the equivalence of f, 1-f, and 1+f.

- Frequency domain representation:



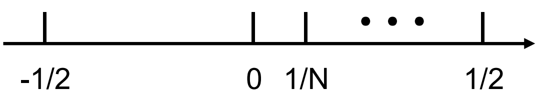
- Discrete-time sinusoidal signal:
  - $f = \frac{i}{N}, \quad i = 1, 2, \cdots, N-1$
  - $f \neq \frac{i}{N}$
- Fundamental frequency for the signal with period N.



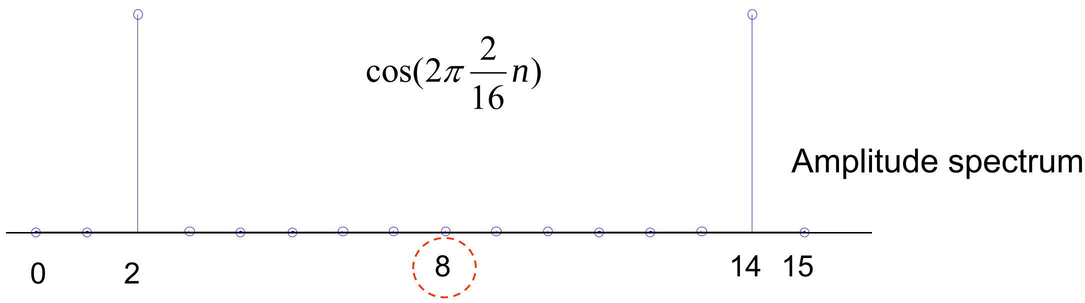
$$f_0 = \frac{1}{N}$$

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

- Frequencies:



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



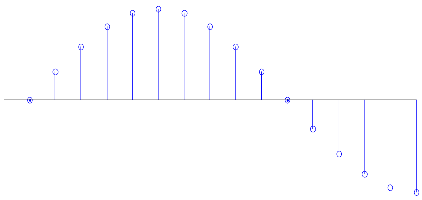
- For real signals, we have two impulses for a sinusoidal since

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

- Practice 2:
  – Generate sinusoidal signals with frequency i/N where N is a power of 2.
  – Observe its DFT spectrum, and the relationship to its time domain signal.

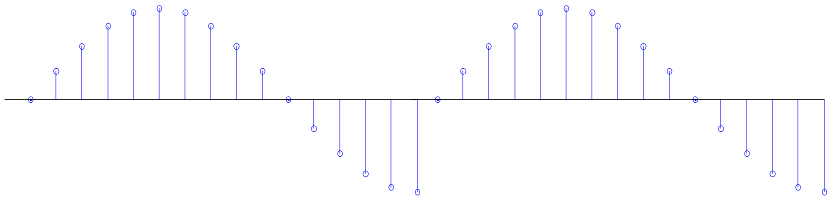
* Fast Fourier transform: fft(x)
* Plot(abs(fft(x)))
* Discrete signal plot: stem(x)

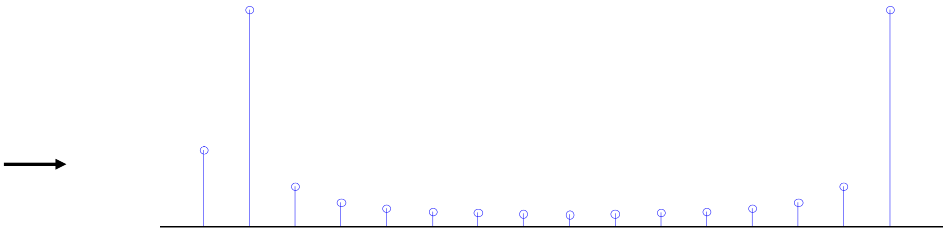
- What about the DFT size is N and $f \neq \frac{i}{N}$?



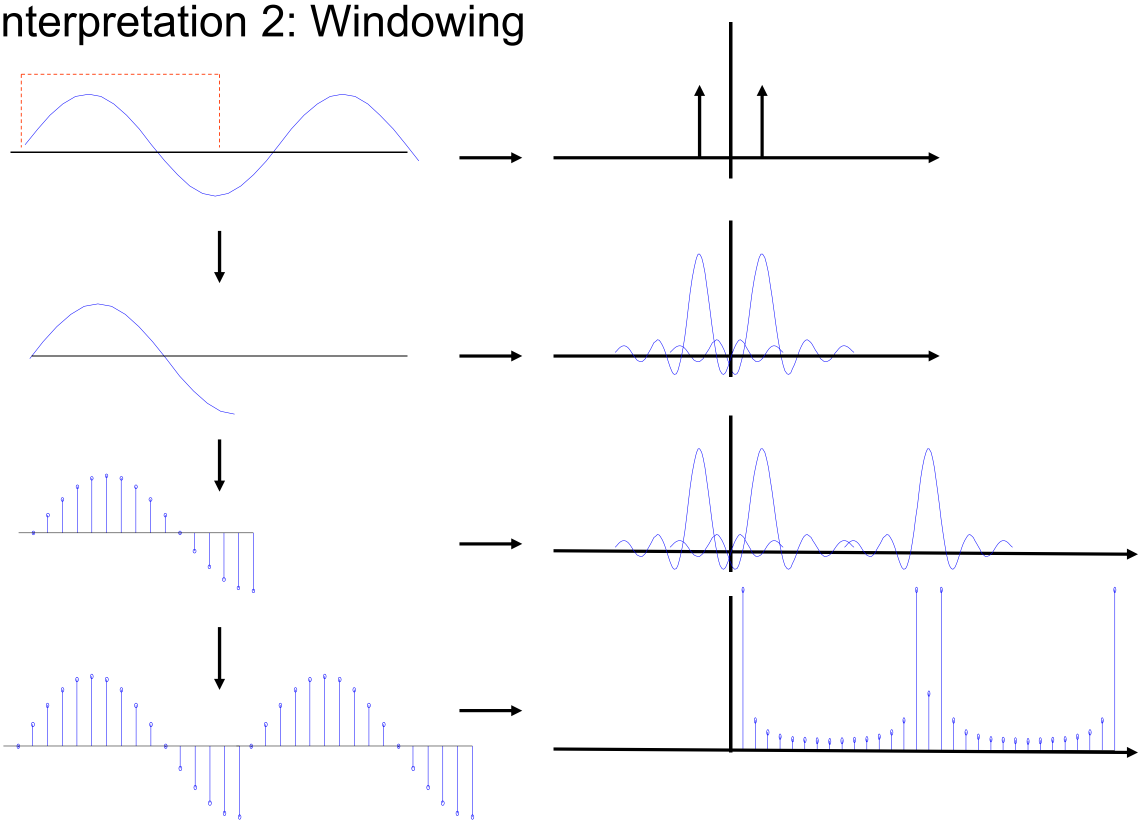
$$f = \frac{0.8}{16}$$

- Interpretation 1: not a sinusoidal signal anymore.

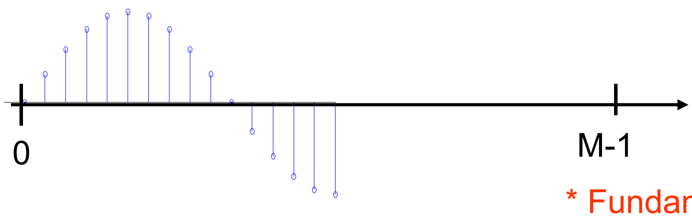




- Interpretation 2: Windowing



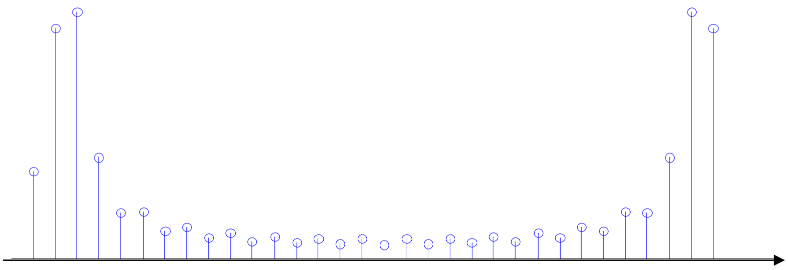
- What about add more points in DFT?



$M > N$

* Fundamental frequency=1/M

- DFT can be seen as a sampling of the spectrum of DTFT. Thus, the spectrum remains the same; only does the density becomes higher.

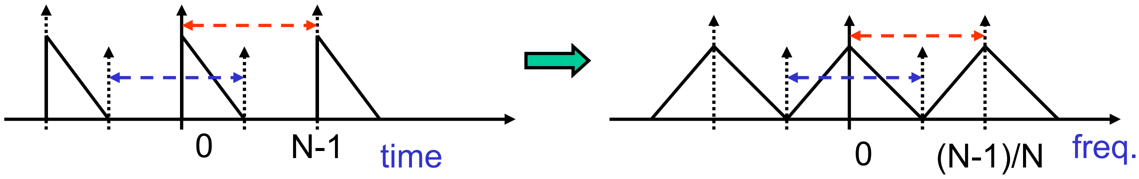


$M = 2 \times 16$

- Practice 3:
  - Generate sinusoidal signals with frequencies not equal to i/N.
  - Observe its DFT spectrum.
  - Pad zeros and observe its DFT spectrum again.
  - Generate sinusoidal signals with frequencies equal to i/N, 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.

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

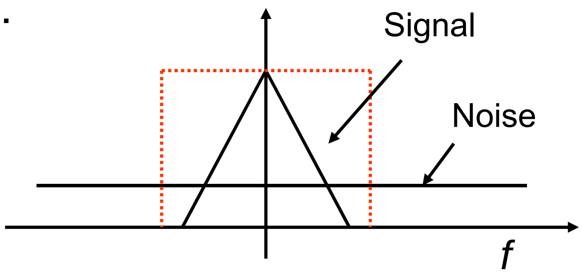


- The command “fft” shows the spectrum period of [0,(N-1)/N] while “ifft” shows the time-domain period of [0,N-1].
- If we want to see the spectrum period of [-1/2, (N-1)/2N] and the time-domain period of [-N/2,N/2-1], we can use the command of “fftshift” to rotate the result of “fft” or “ifft”.

- 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 rectangular pulse and find the sinusoidal components of the waveform.
  - Compress the waveform and observe the relationship of its time domain signal and the number of the sinusoidal reserved.

- 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 signal quality: signal-to-noise-ratio (SNR):

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

$$\mathrm{SNR(dB)}=10\log 10\frac{E\left\{\left|\bar{s}(n)\right|^2\right\}}{E\left\{\left|e(n)\right|^2\right\}}$$

- Note that we may have to adjust signal output delay/gain to obtain the maximum value.
- 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:
  – Generate a noise sequence and add it to a triangular waveform.
  – Recover the waveform with a frequency domain windowing operation.
  – Calculate the output SNR.

* Add Gaussian noise: randn(n,m)

- 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. :

$$s = 1000;$$
$$x = \mathrm{sgn}(randn(1,s));$$
$$v = \sqrt{\sigma_v^2} \times randn(1,s);$$
$$y = x + v;$$

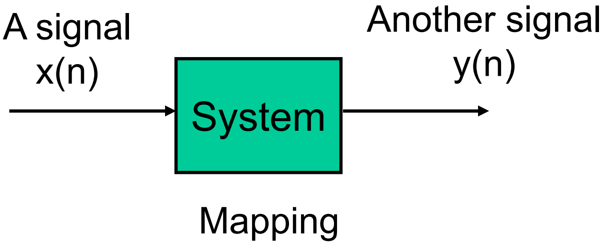
- For complex signal, the noise is also complex. As a result, the calculated variance has to be divided by two for the generation of real or imaginary noise.

- 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 smaller than 128) and add Gaussian noise to yield an average SNR of 15dB.
  - Using the frequency windowing technique to conduct a filtering operation (conducting 128-point DFT).
  - Find an optimum windowing function such that the mean-square-error 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.

$$\begin{aligned} x_1(n) &\to y_1(n) \\ x_2(n) &\to y_2(n) \end{aligned} \quad \Longrightarrow \quad ax_1(n) + bx_2(n) \to ay_1(n) + bx_2(n)$$

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

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

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

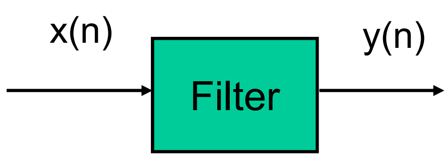
- Causal/noncausal:
  - Causal

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

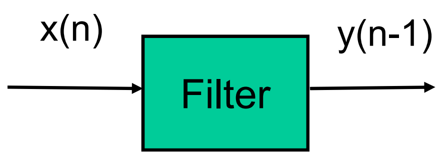
  - Noncausal

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

- 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

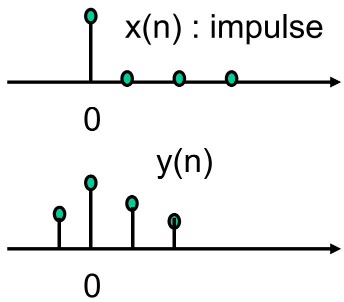
- LTI systems: linear and time-invariant
  - The output is a linear combination of the input values
  - Can be causal of noncausal

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

  - e.g.

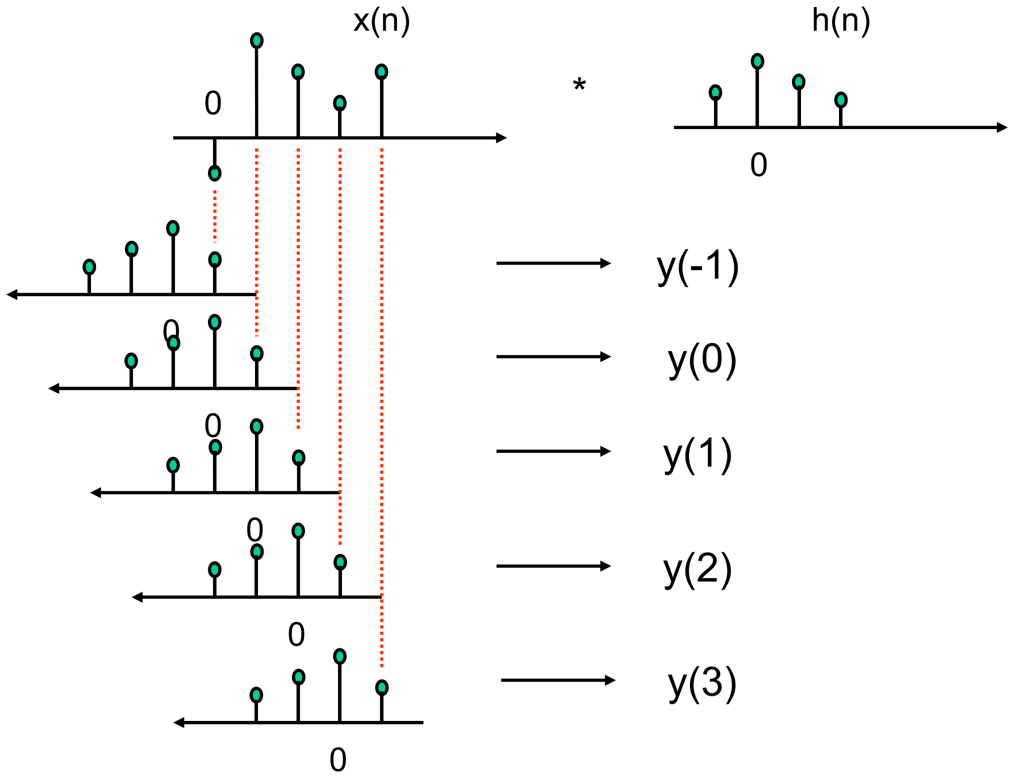
$$y(n) = h(-1)x(n+1) + h(0)x(n) + h(1)x(n-1) + h(2)x(n-2)$$

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



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

- 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:

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

- 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

$$\begin{aligned} H(z) &= \frac{a(0) + a(1)z^{-1} + a(2)z^{-2} + \cdots a(N)z^{-N}}{1 + b(1)z^{-1} + b(2)z^{-2} + \cdots b(M)z^{-M}} \\ &= \frac{(1 - a_1 z^{-1})(1 - a_2 z^{-1})\cdots(1 - a_N z^{-1})}{(1 - b_1 z^{-1})(1 - b_2 z^{-1})\cdots(1 - b_M z^{-1})} \end{aligned}$$

zero

Pole

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

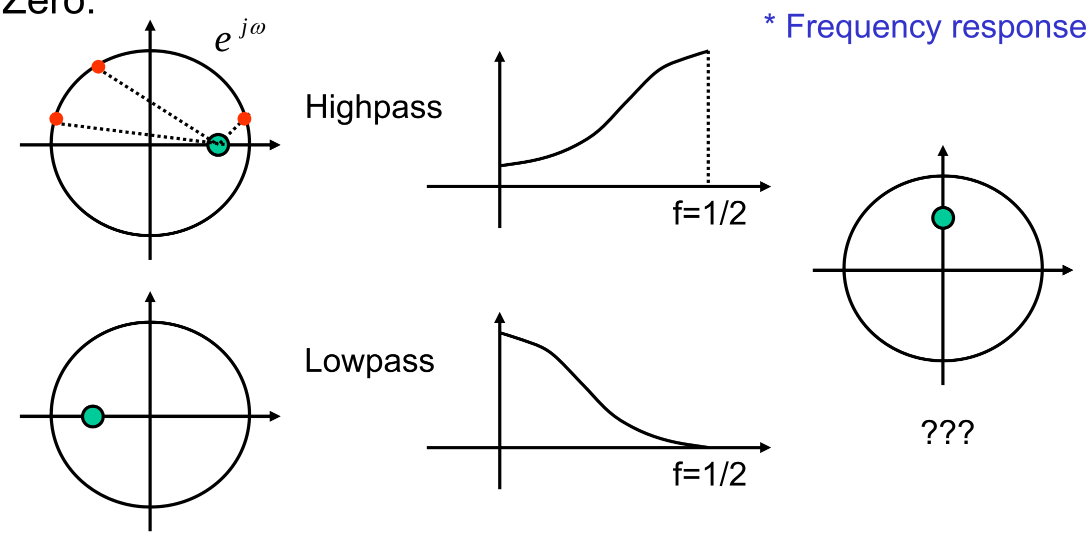
- Practice 1:
  - Generate a signal and the impulse response of an FIR system, and conduct the convolution operation (cannot use conv(.) command).
  - Generate a signal and a difference equation of an IIR system, and find the output of the system (cannot use filter(.) command).

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

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

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

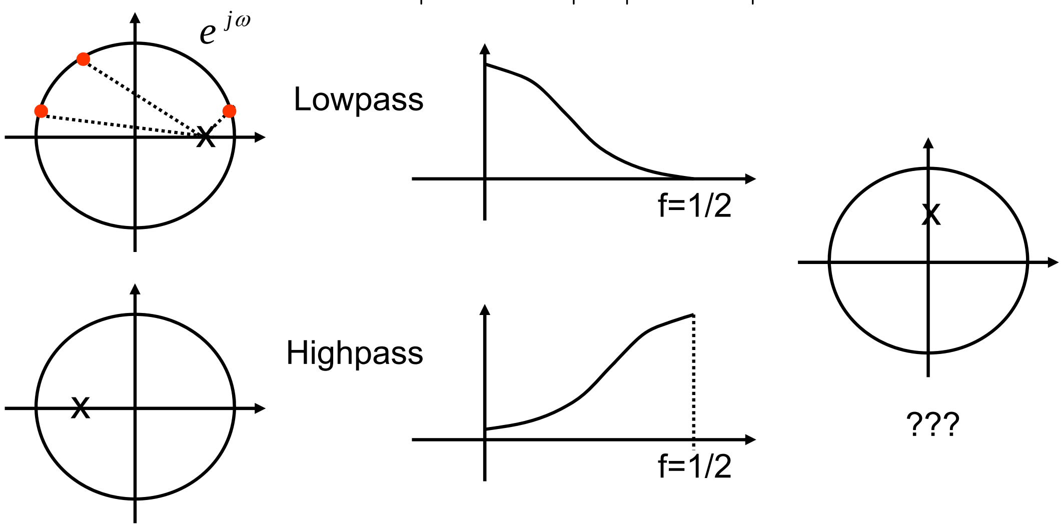
Zero:



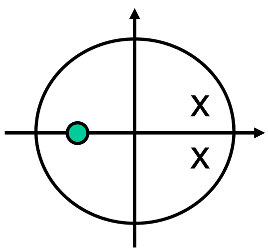
- For a pole:

$$H(z)=\frac{1}{1-b_1z^{-1}}\rightarrow H(e^{j\omega})=\frac{1}{1-b_1e^{-j\omega}}$$

$$\left|H(e^{j\omega})\right|=\frac{1}{\left|1-b_1e^{-j\omega}\right|}=\frac{1}{\left|e^{j\omega}-b_1\right|}$$



- So, we can design a filter 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

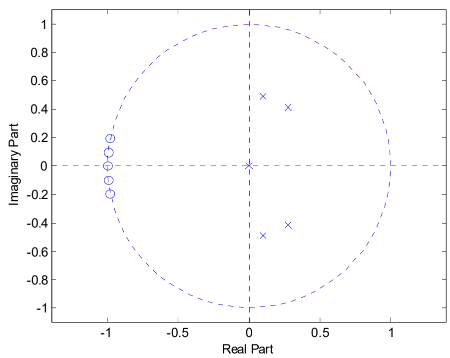
$$\frac{Y(z)}{X(z)}=\frac{1+0.9z^{-1}}{1-1.6z^{-1}+0.73z^{-2}}\Leftrightarrow$$

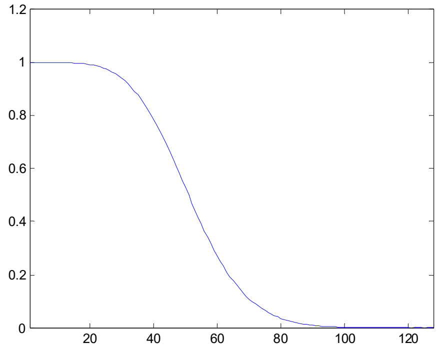
$$Y(z)-1.6Y(z)z^{-1}+0.73Y(z)z^{-2}=X(z)+0.9X(z)z^{-1}\Leftrightarrow$$

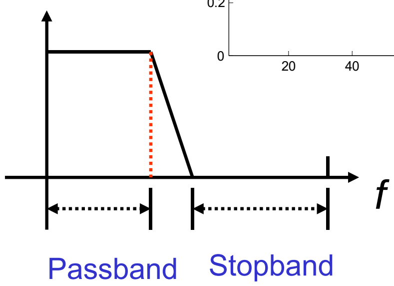
$$y(n)-1.6y(n-1)+0.73y(n-2)=x(n)+0.9x(n-1)\Leftrightarrow$$

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

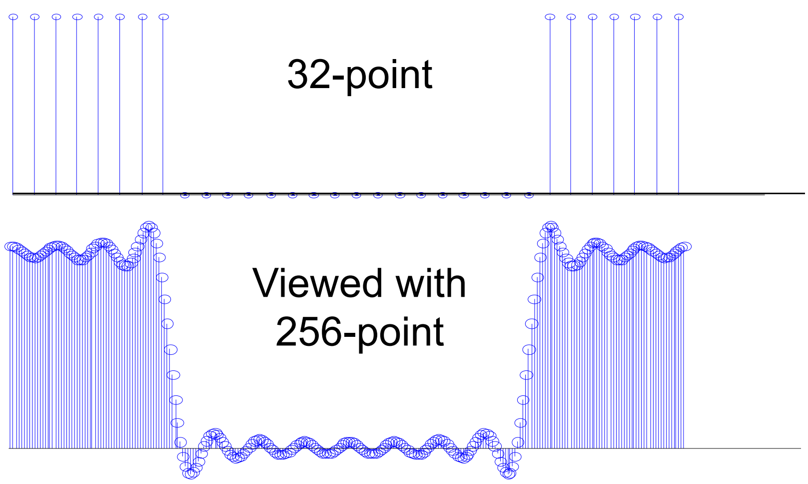
- Design guidelines:
  - Since zeros are simpler to control, place zeros on the unit circle around the stopband.
  - If the passband is not flat enough, put poles on a circle (with a radius smaller than one) around the passband.



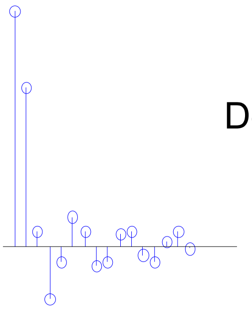




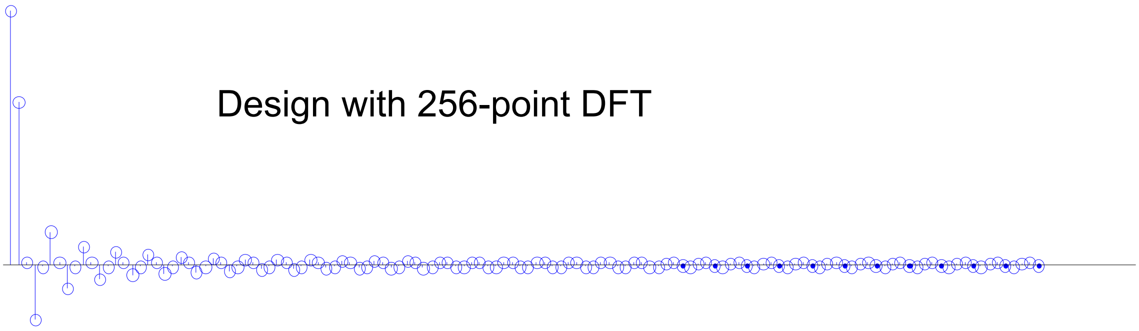
- You can also use DFT to design an FIR filter
  - Specify the frequency response of the desired filter.
  - Transform the response to the time domain.
- How does the size of the DFT influence the filter response?
  - The larger the size, the longer the time-domain response.
  - The spectrum is more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



- Time domain response (right half):



Design with 32-point DFT



Design with 256-point DFT

- 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 calculate the output SNR 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 calculate SNR, we need to conduct gain normalization and delay adjustment (group delay).
  - Sometime, it may require some trial-and-errors to obtain the maximum
  - For a long sequence, you can only calculate the signal and noise power in the stationary region.
  - You can also go to the frequency domain and calculate the power of desired signals and that of undesired ones.

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

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

$$\mathrm{SNR(dB)}=10\log 10\frac{E\left\{\left|\bar{s}(n)\right|^2\right\}}{E\left\{\left|e(n)\right|^2\right\}}$$

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

- Practice 2:
  - Generate a lowpass and highpass FIR systems by zero-placing (roots to polynomials, frequency response).
  - Generate a lowpass and highpass IIR systems by pole-placing .
  - Generate a bandpass system by zero-pole-placing.
  - Generate a lowpass filter by the DFT method.
  - Generate two sinusoidal signals, one in passband and one in stopband, pass the signal to a designed lowpass filter (in the time domain), and observe the output (time-domain signal and spectrum).

* 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)

- Homework:
  - Generate two sinusoidals with frequencies of 1/4 and 1/8.
  - Using the methods mentioned in this lab., design an FIR filter that can filter out the sinusoidal with the frequency of 1/8. The required output SNR is 20dB.
  - Design an IIR filter to do the same work.

- 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(2πfct)

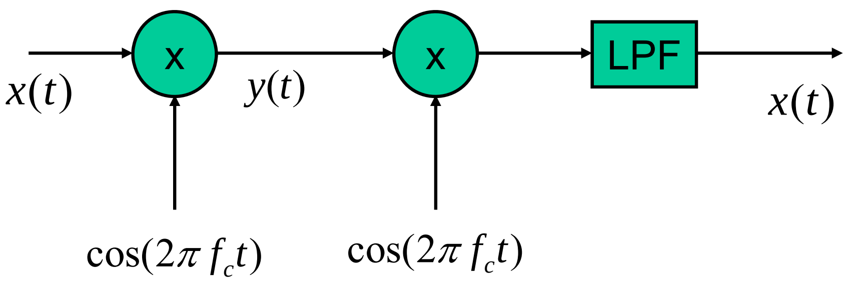
- Why modulation?
  – For transmission
  – For multiple access

- Demodulation (real):

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

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

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



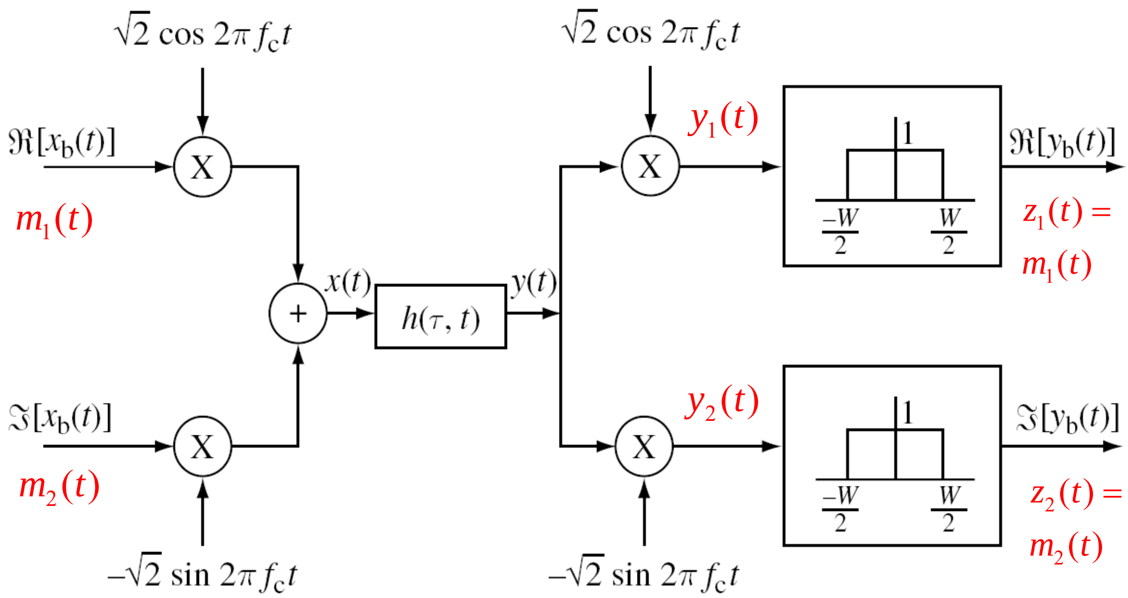
* What will happen if the carrier in the receiver has a phase offset?

- A discrete signal with a high sampling frequency can approximate a continuous signal.

$$y(t) = x(t)\cos(2\pi f_c t) \Leftrightarrow y(nT) = x(nT)\cos(2\pi f_c nT)$$
$$\Leftrightarrow x(n)\cos(2\pi \frac{f_c}{f_s} n) \Leftrightarrow x(n)\cos(2\pi \hat{f}_c n)$$

- Practice 1:
  – Generate a sinusoidal signal with frequency of 1/64 and a carrier with frequency of 1/4.
  – Conduct the modulation.
  – Design a LPF.
  – Conduct the demodulation.
  – Generate a triangular pulse with the duration of 64 and conduct the modulation and demodulation again.

- Modulation with two carries (complex):



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

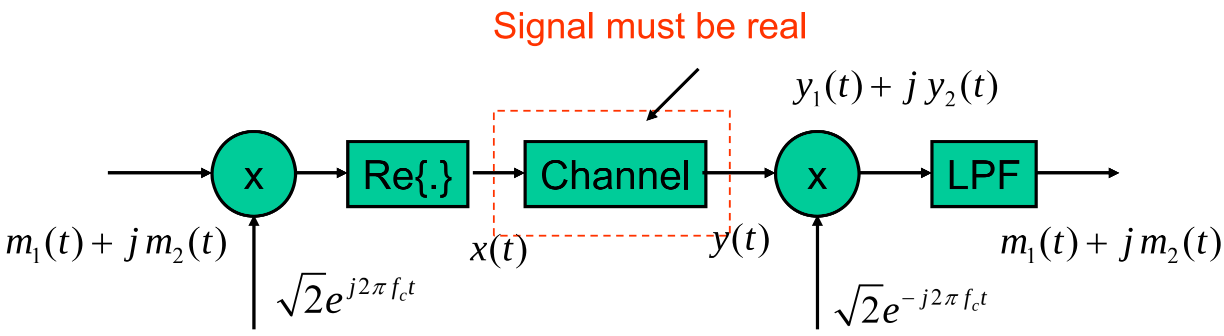
- Let y(t)=x(t), then

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

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

- If we treat $m_1(t)$ and $m_2(t)$ as the real and the imaginary part of a complex signal m(t), i.e., $m(t)= m_1(t)+j\, m_2(t)$, the whole system can be represented by an equivalent system with complex signal representation.

- The equivalent system:



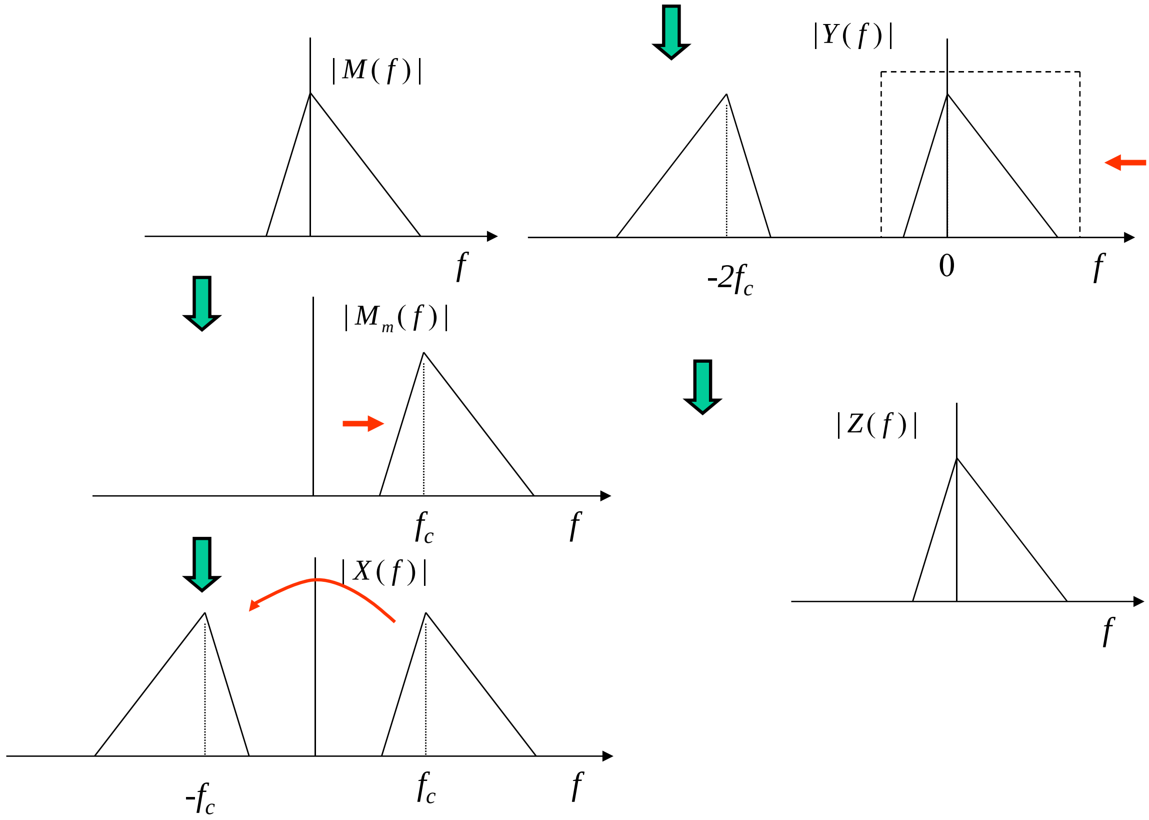
$$
x(t) = \sqrt{2}\left[m_1(t)\cos(2\pi f_c t) - m_2(t)\sin(2\pi f_c t)\right]
$$

$$
\begin{aligned}
y(t)\sqrt{2}e^{-j2\pi f_c t} &= y(t)\sqrt{2}\left(\cos(2\pi f_c t) - j\sin(2\pi f_c t)\right) \\
&= y_1(t) + jy_2(t)
\end{aligned}
$$

* y(n)=x(n)

$$
\begin{cases}
y_1(t) = \sqrt{2}y(n)\cos(2\pi f_c t) = \sqrt{2}x(n)\cos(2\pi f_c t) \\
y_2(t) = -\sqrt{2}y(n)\sin(2\pi f_c t) = -\sqrt{2}x(n)\sin(2\pi f_c t)
\end{cases}
$$

- 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

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

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

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

- Then,

$$y_1(t) = 2\left[m_1(t)\cos(2\pi f_c t+\theta) - m_2(t)\sin(2\pi f_c t+\theta)\right]\cos(2\pi f_c t)$$
$$= m_1(t)\left[\cos(4\pi f_c t+\theta) + \cos(\theta)\right] - m_2(t)\left[\sin(4\pi f_c t+\theta) + \sin(\theta)\right]$$
$$y_2(t) = -2\left[m_1(t)\cos(2\pi f_c t+\theta) - m_2(t)\sin(2\pi f_c t+\theta)\right]\sin(2\pi f_c t)$$
$$= -m_1(t)\left[\sin(4\pi f_c t+\theta) - \sin(\theta)\right] + m_2(t)\left[-\cos(4\pi f_c t+\theta) + \cos(\theta)\right]$$

- 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

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

- The complex output is then $z(t)=m(t)h(t)$.

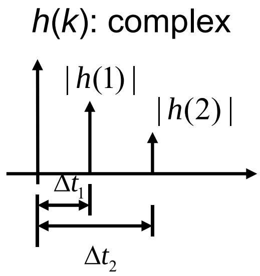
$$z(t) = (m_1(t) + jm_2(t))(\cos(\theta) + j\sin(\theta))$$

- In general, the channel has a multi-path response which can be modeled as

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

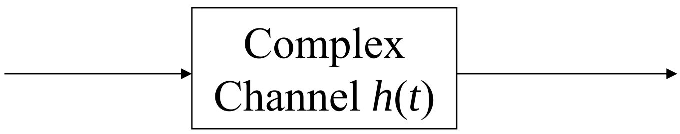
- Then,

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

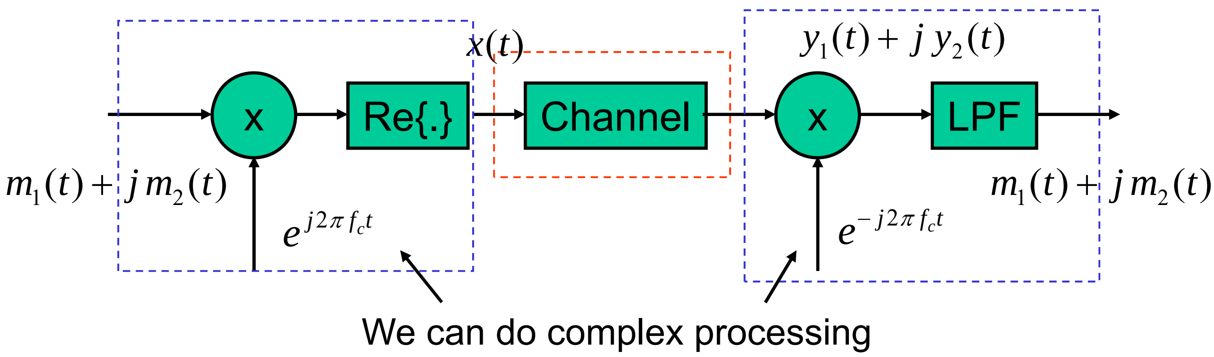


- Equivalent complex baseband representation (without involving modulation/demodulation):

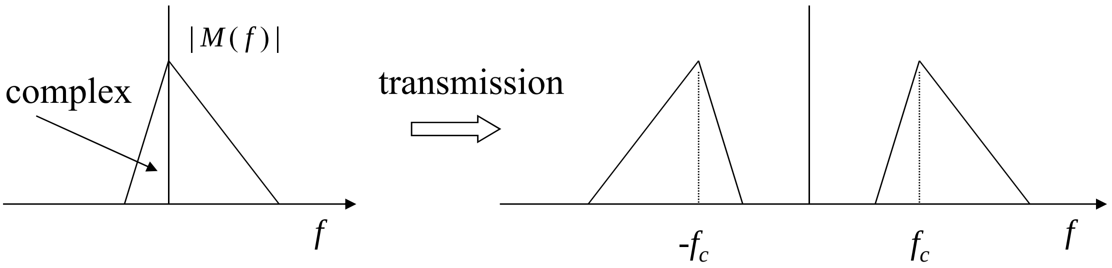
$$m(t) = m_1(t) + j\,m_2(t) \qquad z(t) = z_1(t) + j\,z_2(t)$$

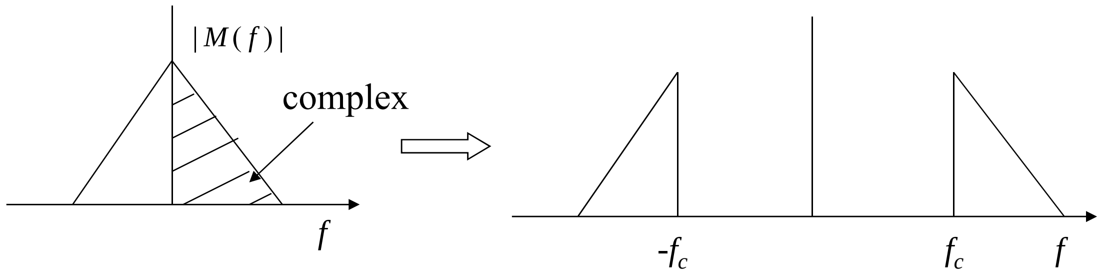


$$z(t) = m(t) * h(t)$$

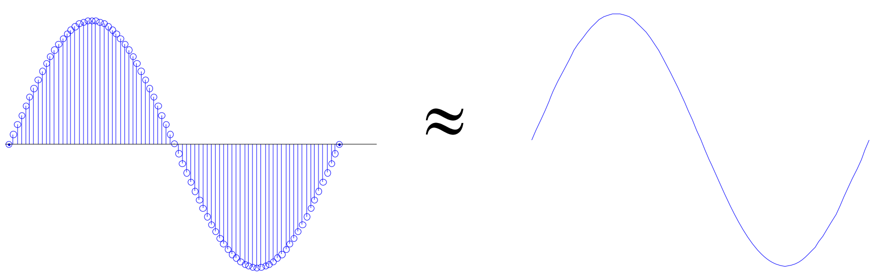


- Types of complex baseband signals:
  - Formed by two real signals
  - Extracted from a real signal





- As mentioned, we can only general discrete-time signals. To simulate analog modulation, we use sampled continuous-time signals. And, the sampling rate is high (let t=nT and T is small).



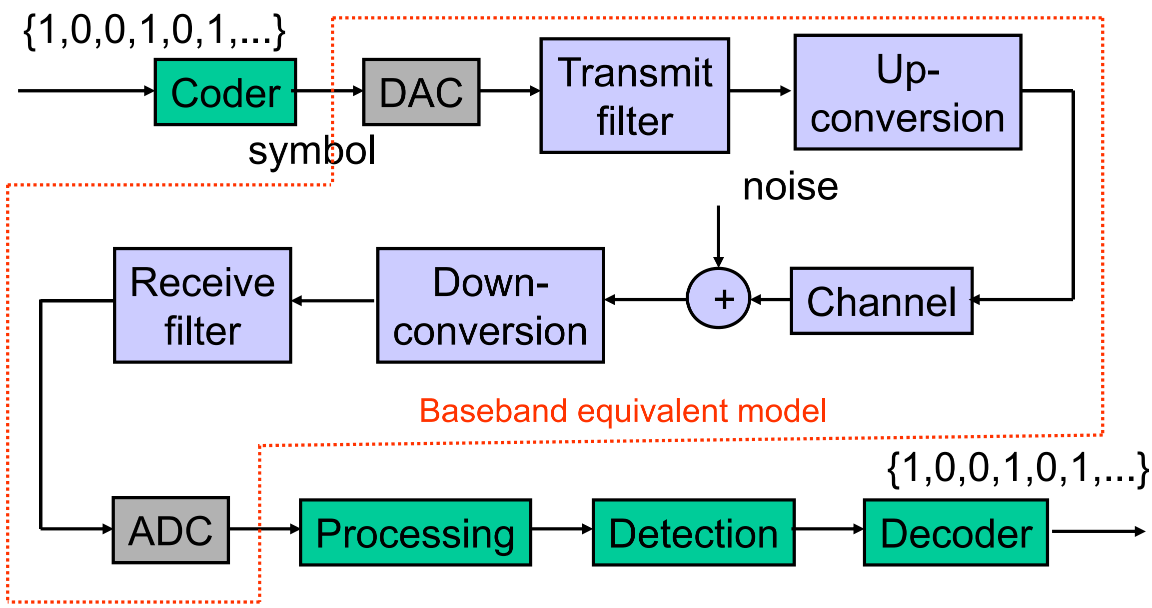
- Practice 2:
  - Implement a complex modulation system with real operations (Let I-branch transmit a rectangular pulse and Q-branch a triangular pulse).
  - Implement a complex modulation system with complex operations.

- Homework:
  - Create two real signals for transmission.
  - Implement a complex modulation system with the complex baseband signal.
  - Assume a channel delay and observe the output signals.
  - Implement the equivalent baseband representation and check if it matches the actual system.

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

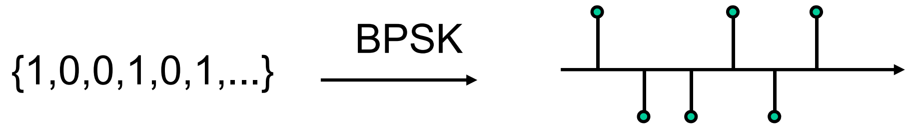
Lab. 5 Digital Modulation

- Digital modulation:

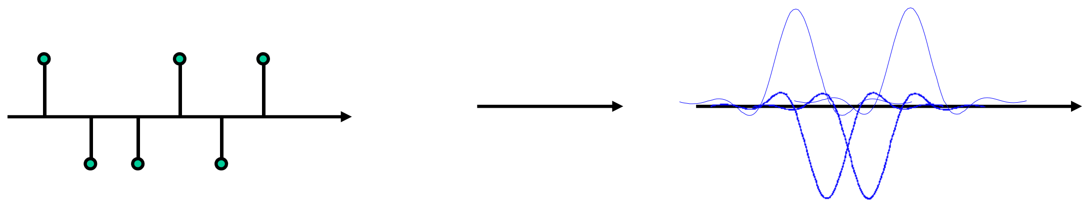


- Coder (symbol mapper):
  - Maps the input bits to a symbol (number).
- Digital-to-analog converter (DAC)
  - Convert digital sequence to analogy waveform
- Transmit filtering (pulse shaping):
  - Maps the symbol to an analog waveform.
- Receiver filtering (matched filtering)
  - Match the transmit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

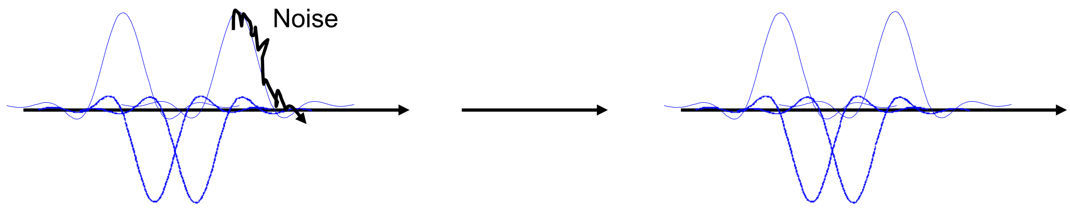
- Symbol mapping:



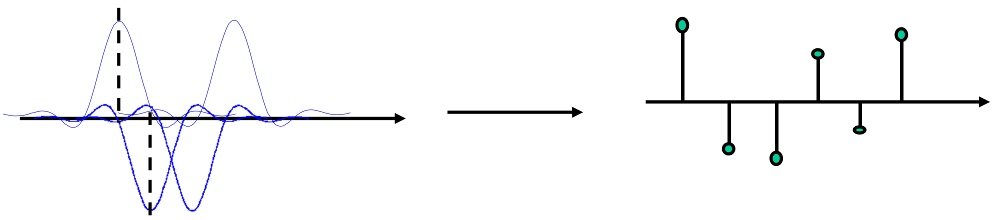
- Transmit filtering:



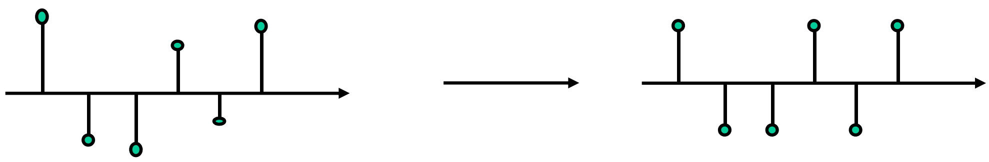
- Receive filtering:



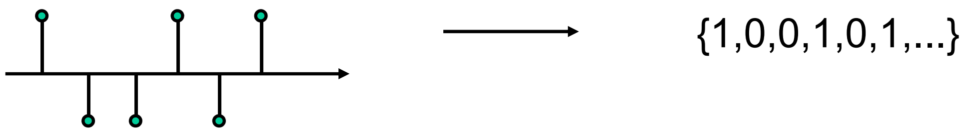
- Sampling:



- Detection:

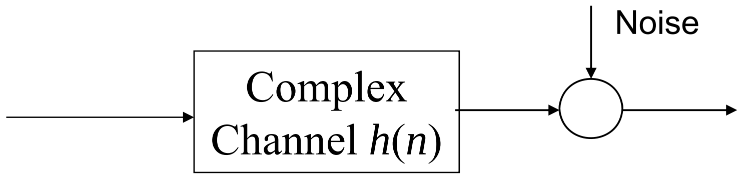


- Demapping:



- Digital equivalent baseband model:

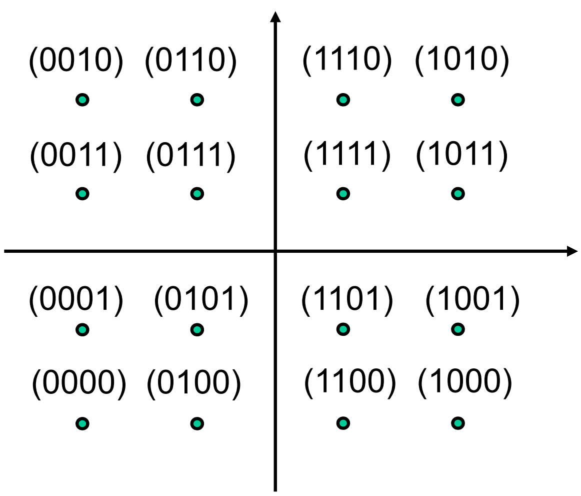
$$a(n) = a_I(n) + j\,a_Q(n) \qquad y(n) = y_I(n) + j\,y_Q(n)$$



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

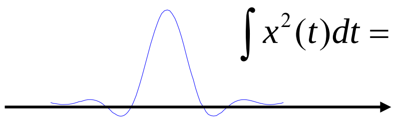
- Coder (symbol mapping):
  - PAM (±1,±3,±5,±7,..)
  - QAM (±1/±3/±5/±7,..+j ±1/±3/±5/±7)

- Practice 1:
  - Generate a 8-PAM mapper.
  - Generate a 16-QAM mapper (Gray mapping).
  - Map a bit sequence with the mappers.

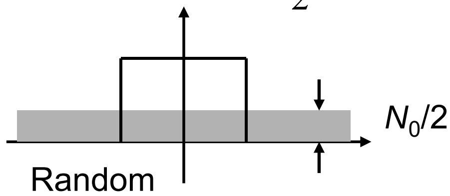


- E_s/N_0 : symbol energy to noise spectral density ratio

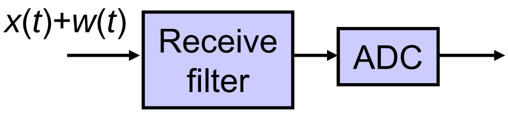
$$x(t) = \sqrt{E_s}\phi(t)$$
$$\int x^2(t)dt = E_s$$

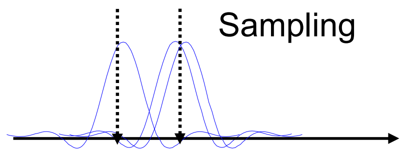


$$E\{w(t)w(s)\} = \frac{N_0}{2}\delta(t-s)$$



- At the ADC output:





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

Signal power: E_s

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

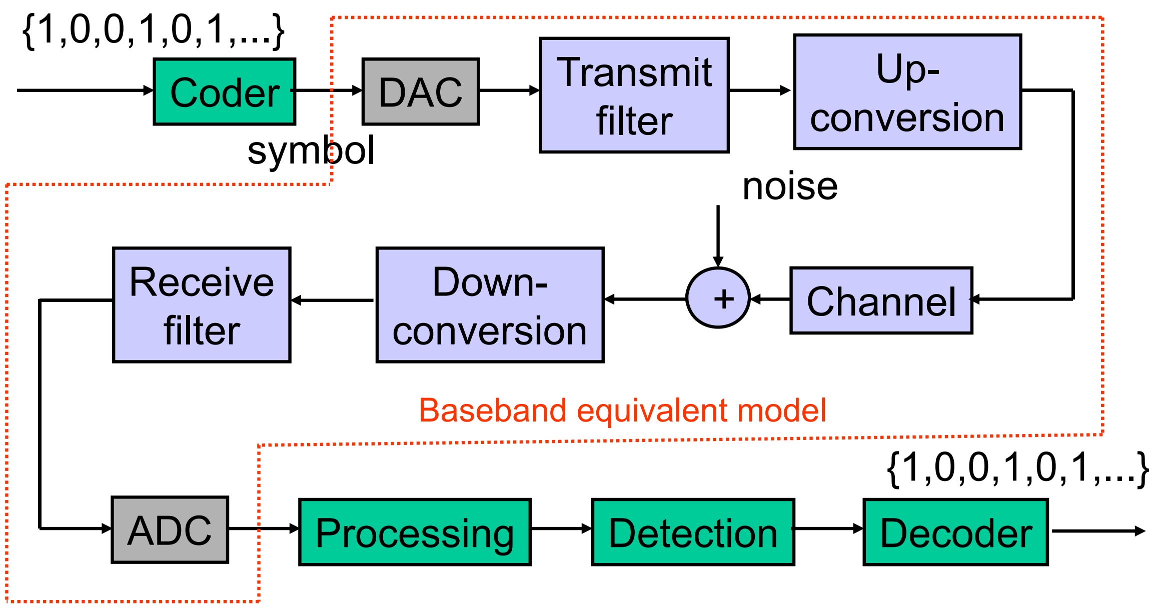
- Thus, the SNR (evaluated in the digital domain) is then $2E_s/N_0$. For complex signals, we then have SNR=E_s/N_0.
- Now, if a symbol carry M bits information, then the energy consumed for one bit transmission is then $E_s = ME_b$. Then,

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

* For BPSK modulation, SNR=E_b/N_0

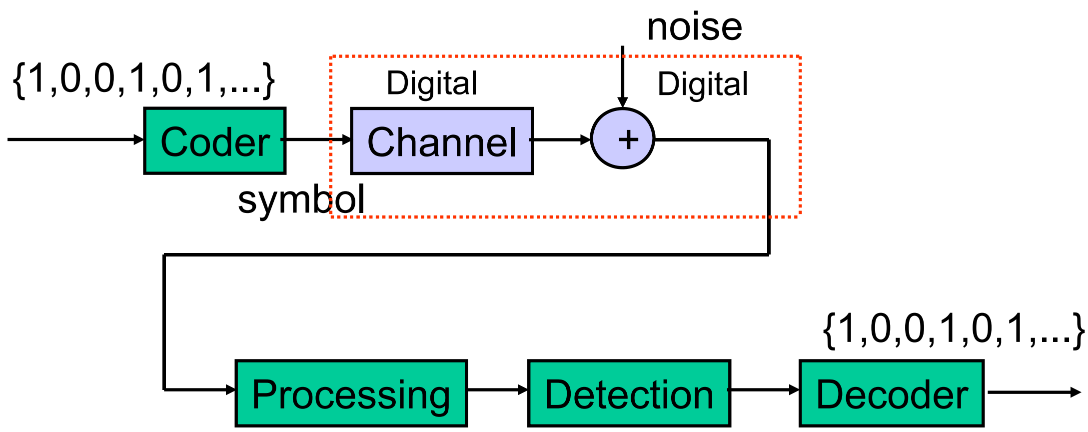
- E_b/N_0 is then a normalized measure indicating “SNR per bit”.

- We first consider the baseband equivalent system.



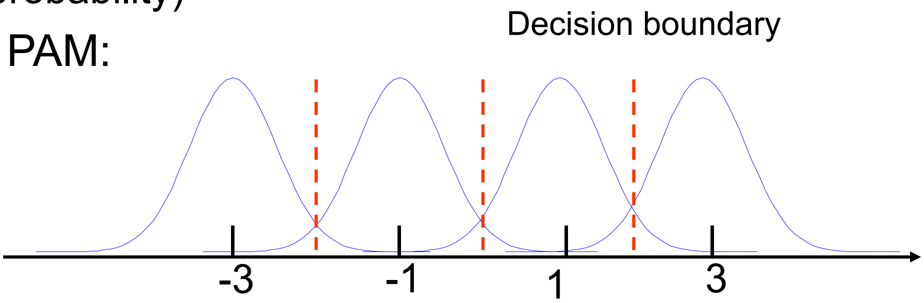
- Note that the operations of the digital systems can be exactly modeled. However, those of analog systems can only be approximately modeled.

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

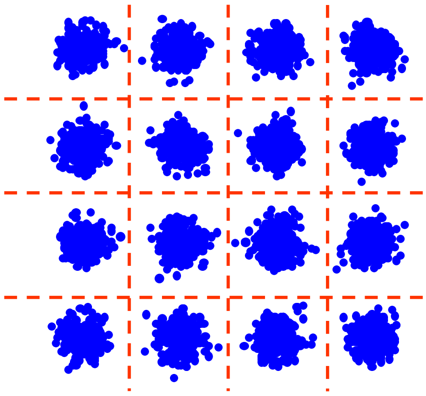


- This assumes that all the processing in analog domain is either perfect or can be absorbed into the channel and noise effects.

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



- For QAM:



- Practice 2:
  - Generate a 16-QAM sequence with SNR of 10 dB.
  - Check if your generation is right.
- Practice 3:
  - Add Gaussian noise to the sequences
  - Conduct detection for the PAM/QAM symbols.
  - Calculate the probability of symbol error.

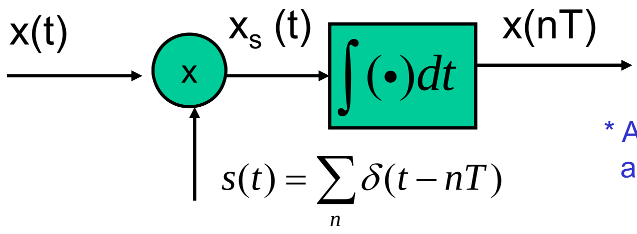
- Homework:
  - Simulate the symbol error rates (SERs) of the 16-QAM scheme with SNRs of 5dB, 10dB, 15dB, etc such that you can plot a SER curve.
  - Calculate the theoretical SERs and also plot a curve.
  - Put these two curves in the same figure to see if your simulation results are OK.

  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:



* An impulse is an analog signal.

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

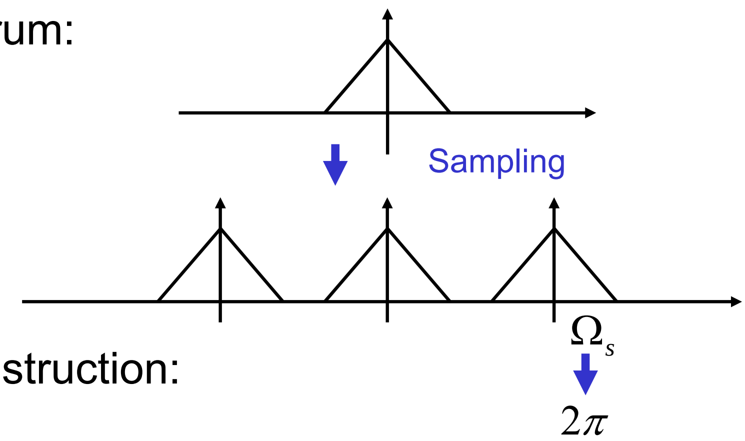
- The Fourier transform of an impulse train is still an impulse train.

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

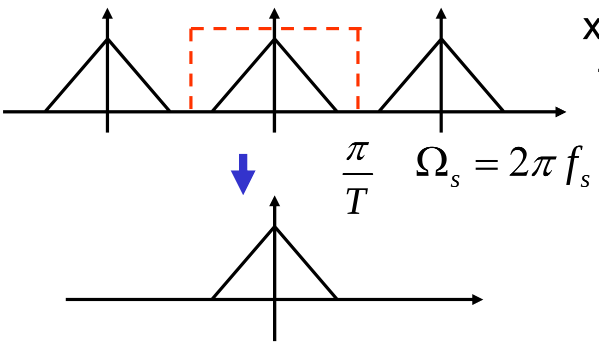
- Then,

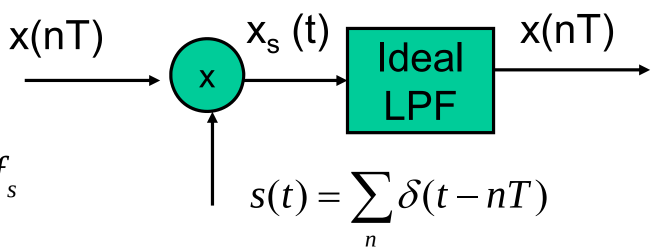
$$X_s(j\Omega) = \frac{1}{2\pi} X(j\Omega) * S(j\Omega) \Rightarrow X_s(j\Omega) = \frac{1}{T}\sum_k X(j\Omega - kj\Omega_s)$$

- Spectrum:



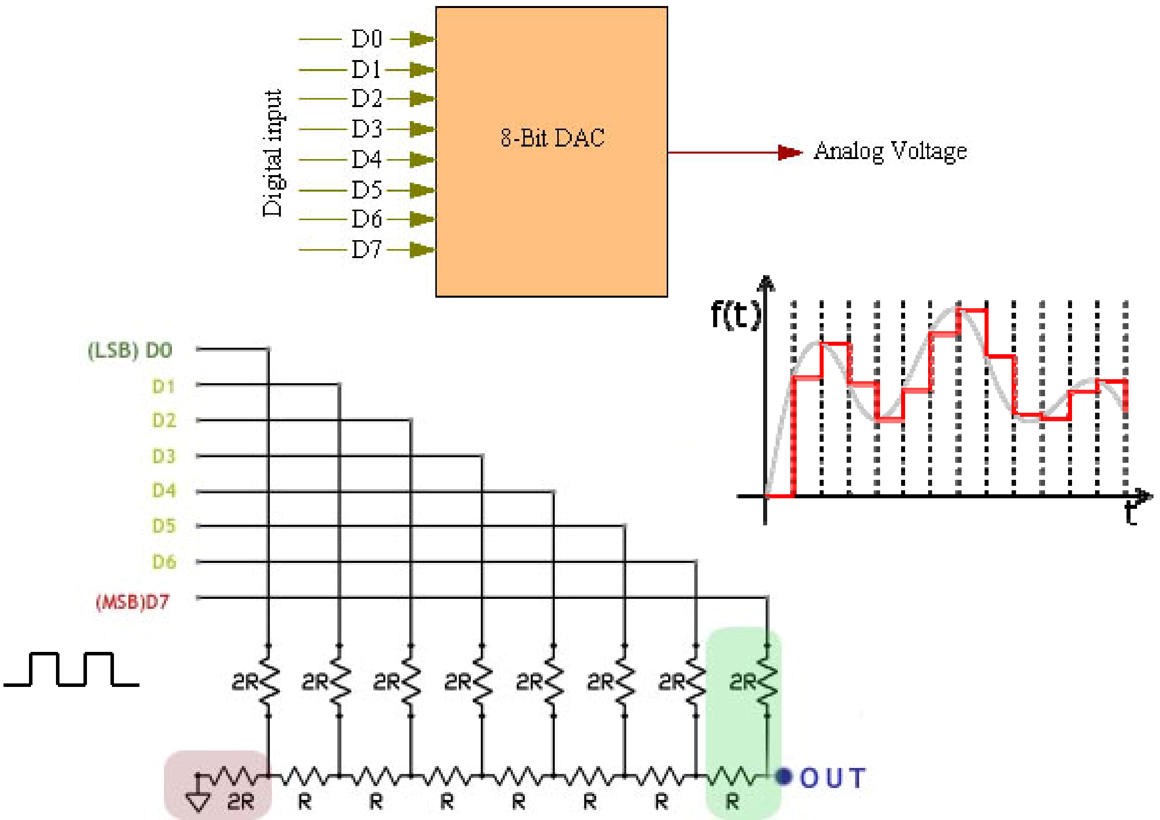
- Reconstruction:



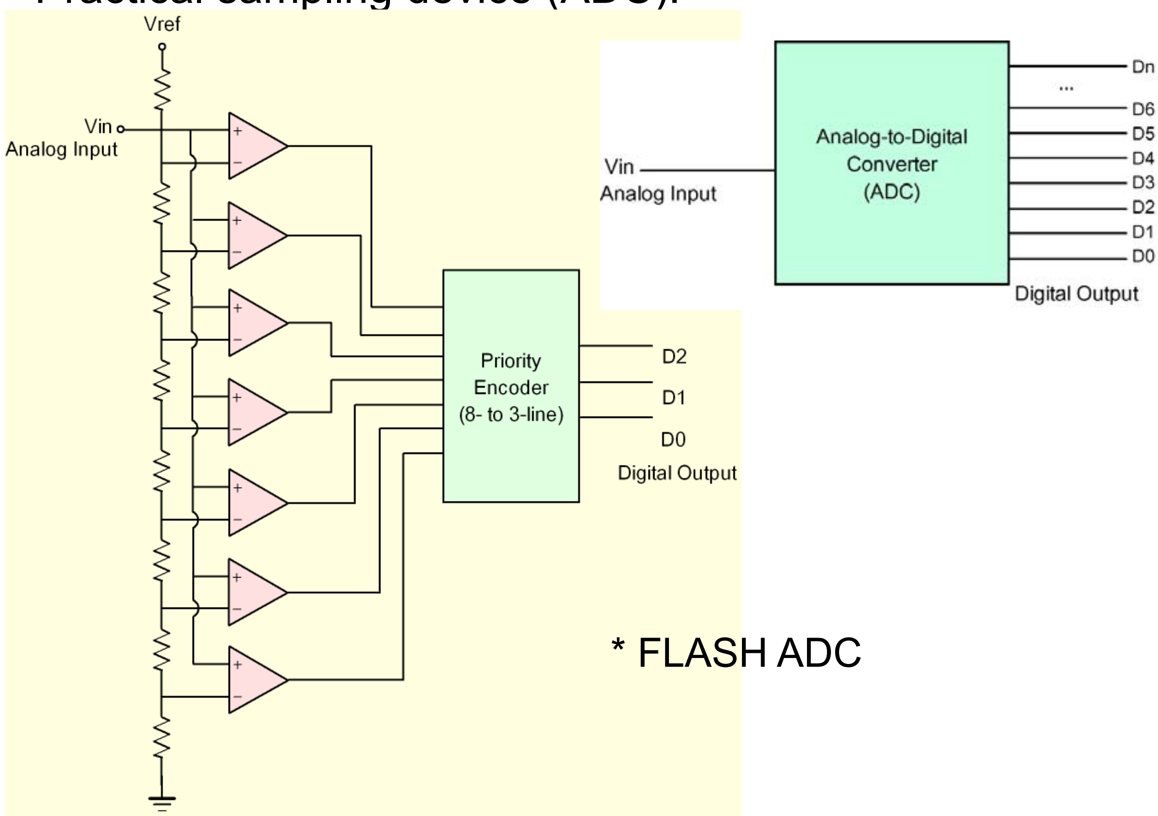


$$h_r(t) = \frac{\sin \pi t / T}{\pi t / T}$$

- Practical reconstruction device (DAC):

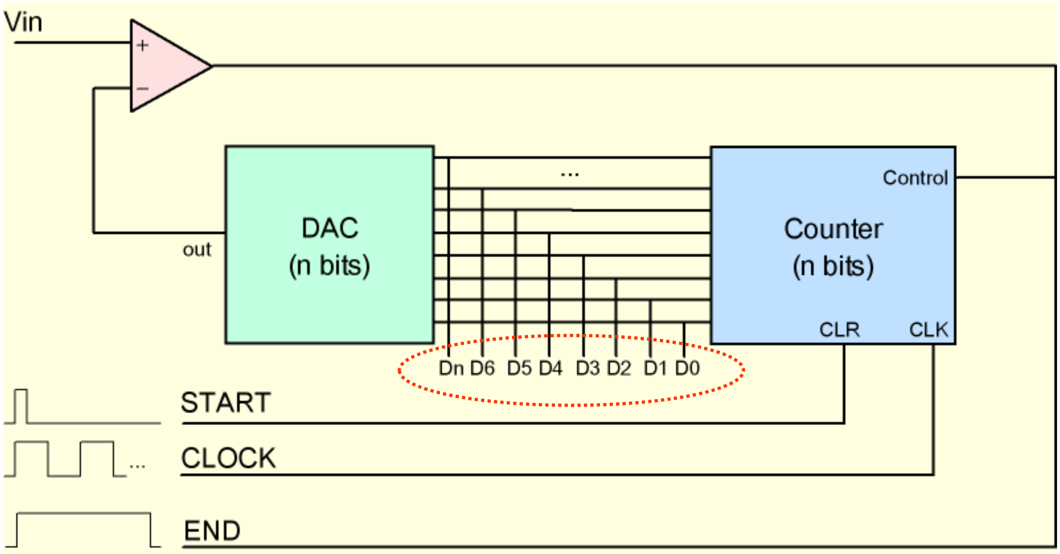


- Practical sampling device (ADC):

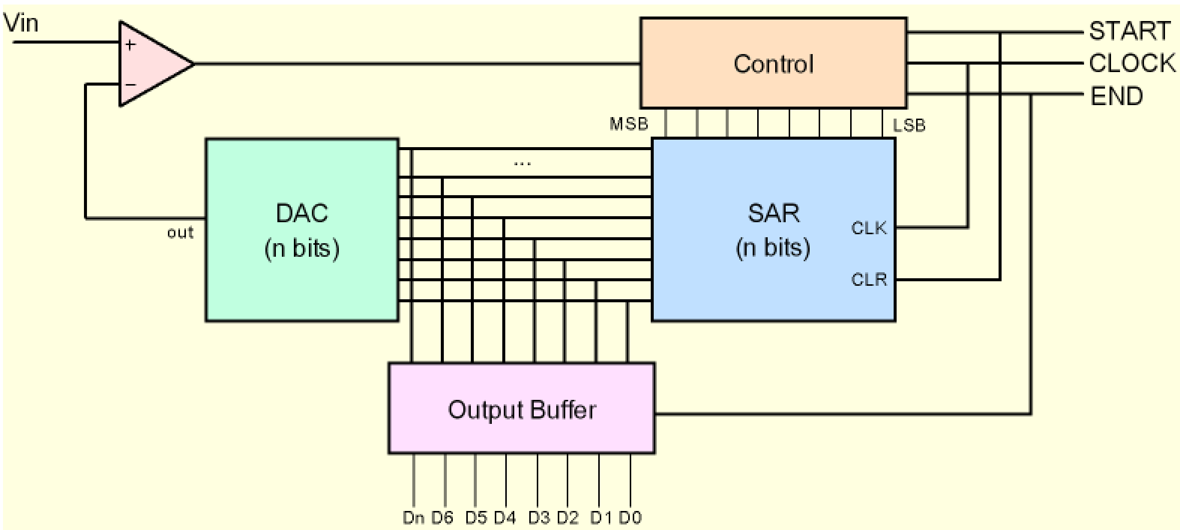


* FLASH ADC

- Ramp counter ADC:

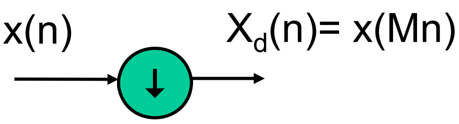


- Successive approximation ADC:



* Tree search

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



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

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

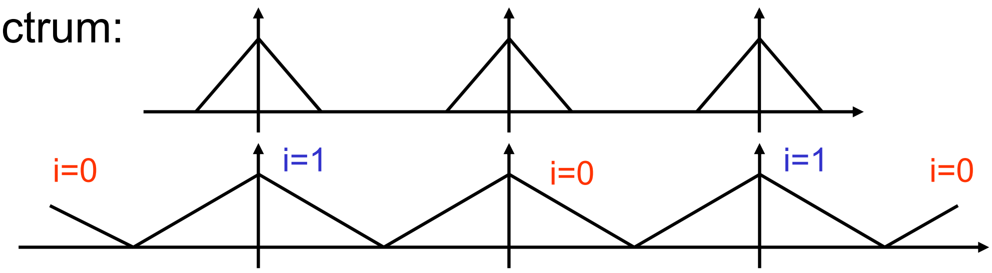
$$\boxed{* \, \omega = \Omega T = \frac{\Omega}{f_s}}$$

- Let m=i+kM and we have

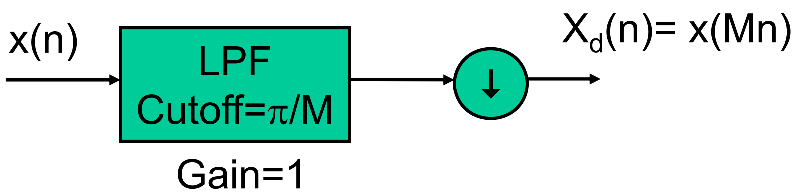
$$X_d(e^{j\omega}) = \frac{1}{M}\sum_{i=0}^{M-1}\left[\frac{1}{T}\sum_k \underbrace{X\left(j\frac{\omega}{MT} - j\frac{2\pi k}{T} - j\frac{2\pi i}{MT}\right)}_{X(e^{j(\omega-2\pi i)/M})}\right] \Rightarrow$$

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

- Spectrum:

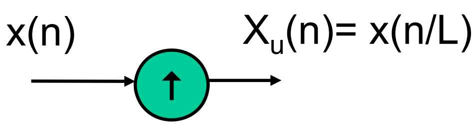


- To avoid aliasing, a filter is generally applied before the downsampling operation.



- Upsampling:

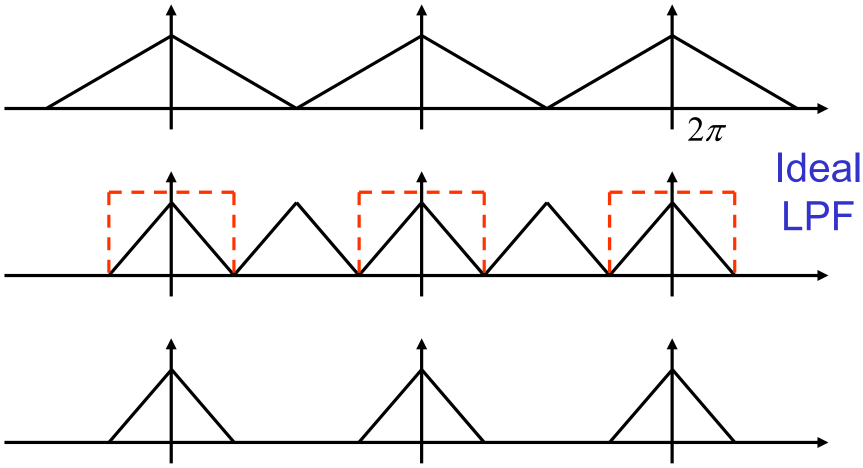
$$x_u(n) = \begin{cases} x(n/L), & n = 0, \pm L, \pm 2L, \cdots \\ 0, & \text{otherwise} \end{cases}$$



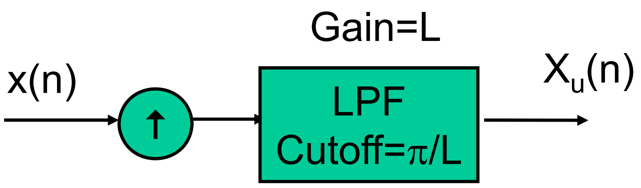
- The spectrum:

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

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



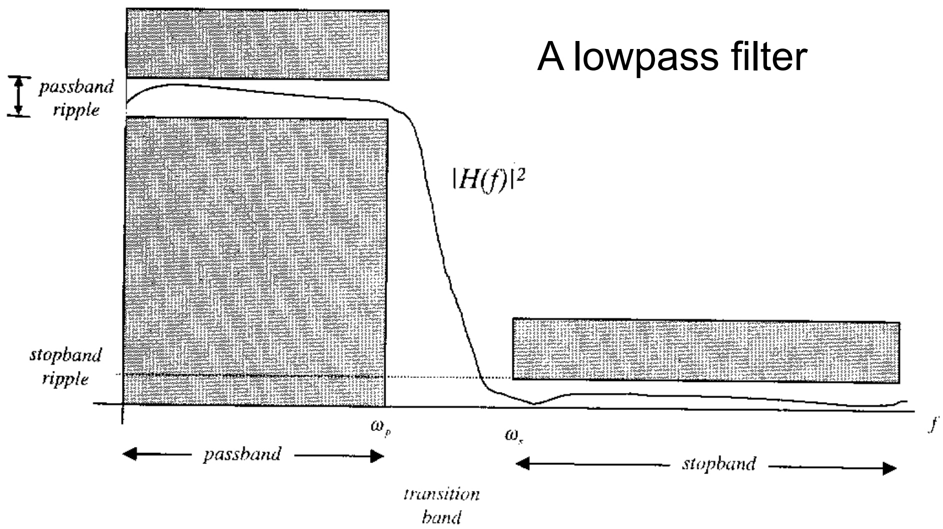
- The upsampling process is then equivalent to increase the sampling rate by a factor of L.



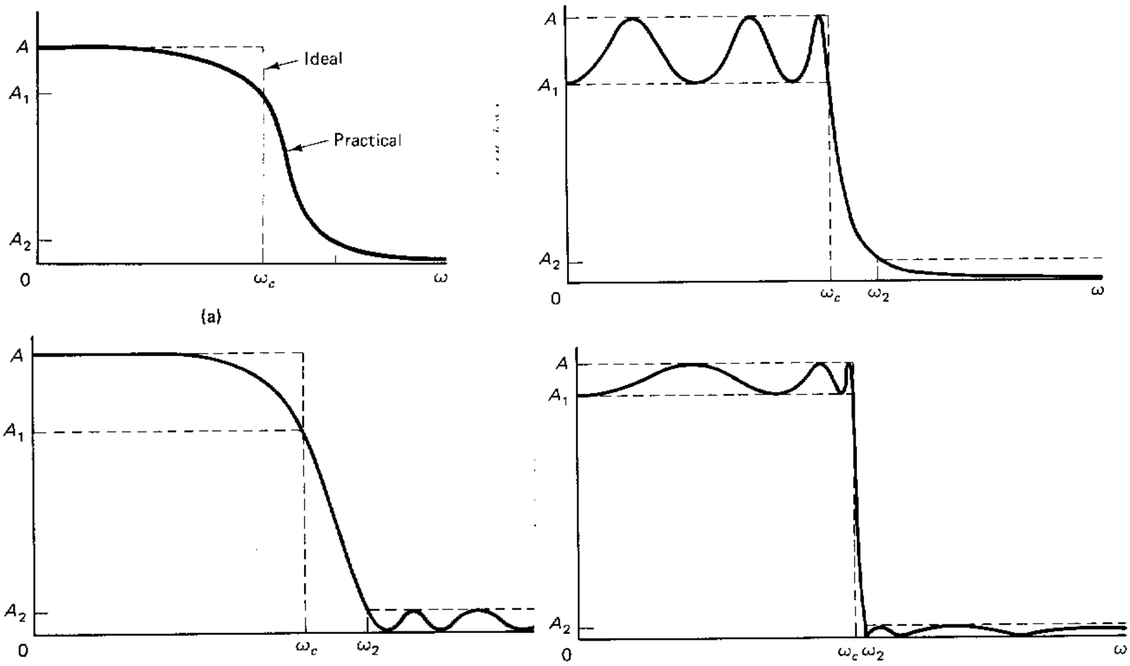
- The filtering operation is also known as interpolation.

- Practice 1:
  - Generate a narrow sinusoidal signal, downsample the signal, and observe the its spectrum.
  - Determine the maximum downsampling rate such that the aliasing will not occur.
  - Then upsample the downsampled signal, and observe its spectrum.

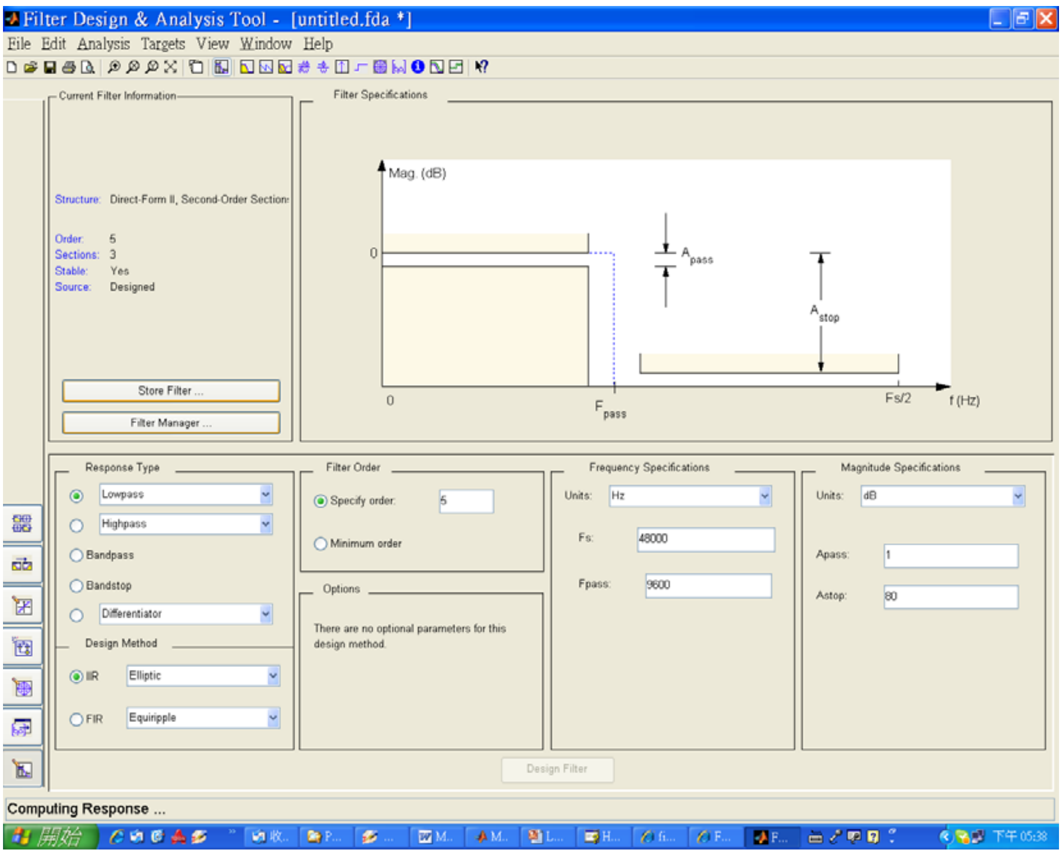
- General filter design:
  - Pass band
  - Stop band
  - Transition band
  - Passband ripple/stopband ripple



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



- filterDesinger in Matlab:



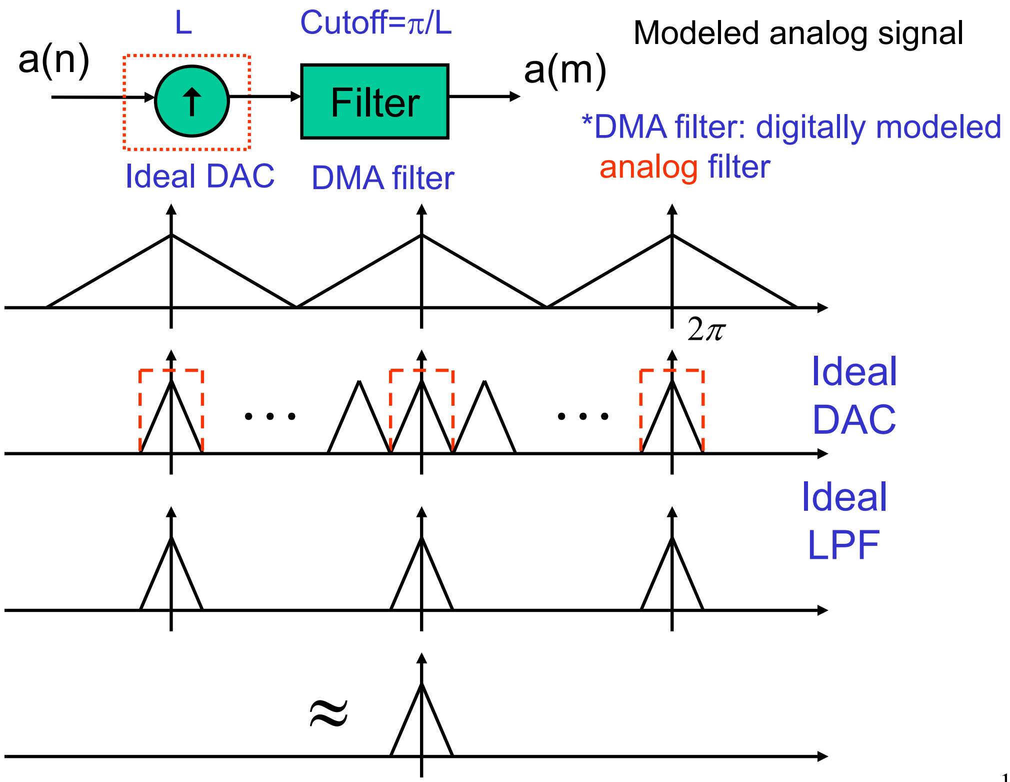
- Practice 2:
  - Generate a sinusoidal signal, downsample the signal (no aliasing), and then upsample the downsampled signal.
  - Design an FIR LPF and let the upsampled signal pass the filter such that the upsampled signal is similar to the original signal.
  - Design an IIR LPF and redo the experiment.
  - Calculate the MSE of these two interpolation schemes (adjusting delay and gain).

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

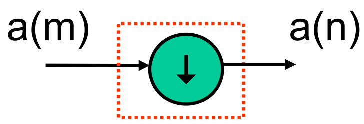
$$e(n)=\overline{s}(n)-s(n)+\overline{v}(n)=y(n)-s(n)$$

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

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



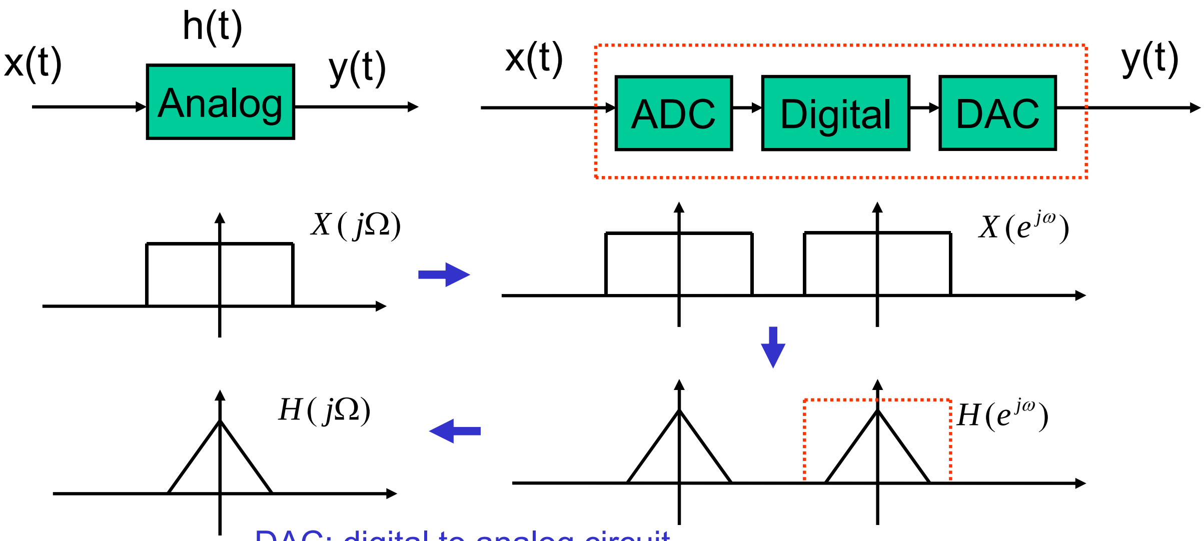
- The filter, which is digital, used to model the analog filter is called digitally modeled analog (DMA) filter.
- Thus, an ideal DAC can be modeled as an device to increase the sampling rate.
- Note that a DMA filter, modeling an analog filter, always has an IIR responses.
- In our simulations, we then 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:
  - Create an arbitrary digital signal
  - Design a DAM filter and let it pass through an ideal DAC with the upsampling rate of 32.
  - Let the modeled analog signal pass with an ADC with the rate of 32.
  - Compare the ADC-sampled signal with the original signal
- Homework:
  - For a given signal, try to conduct downsamping with a largest factor without causing distortion.
  - Design a filter and conduct upsampling to recover the signal.
- Reading assignment:
  - Pulse shaping
  - RC, SRRC

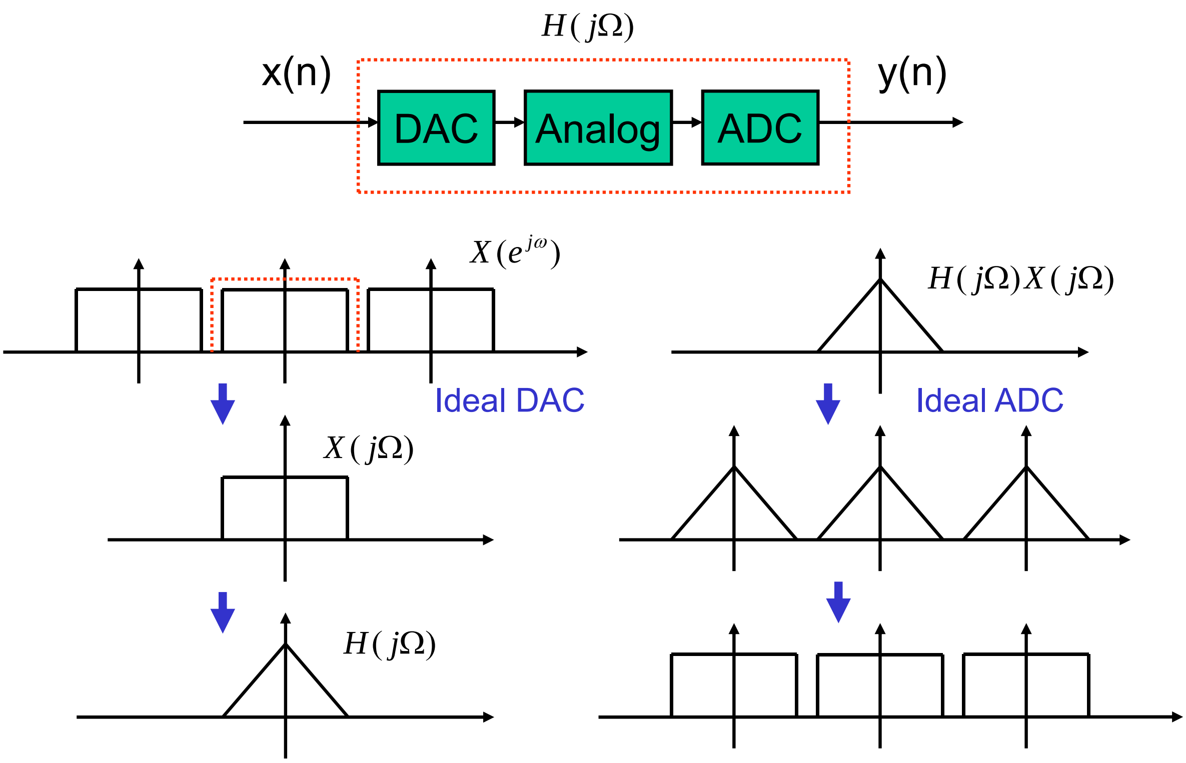
Lab. 7 Transmit Filtering/Up Conversion I

- Digital processing of analog systems:

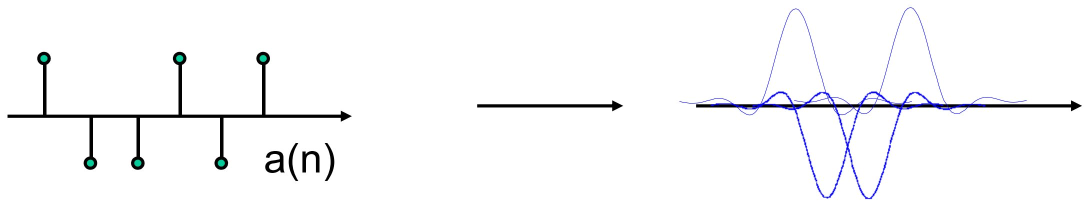


DAC: digital to analog circuit
ADC: analog to digital circuit

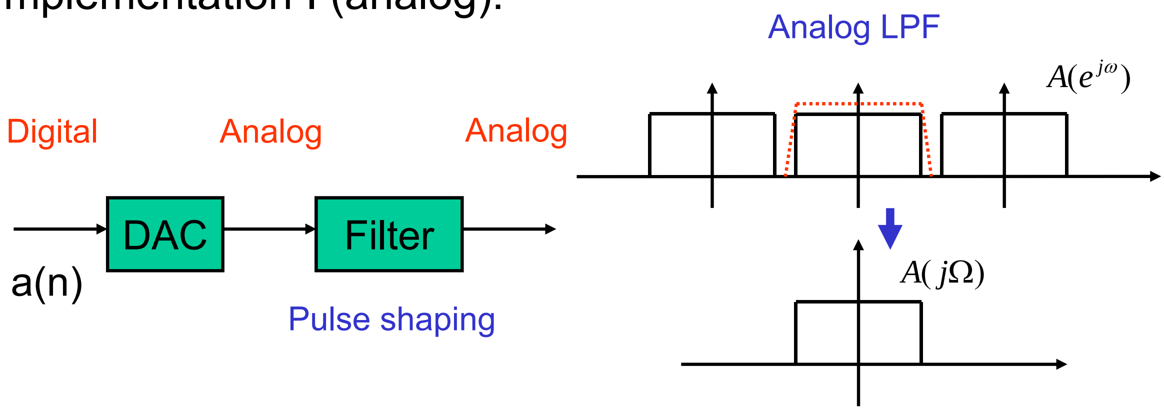
- Digital communication system:



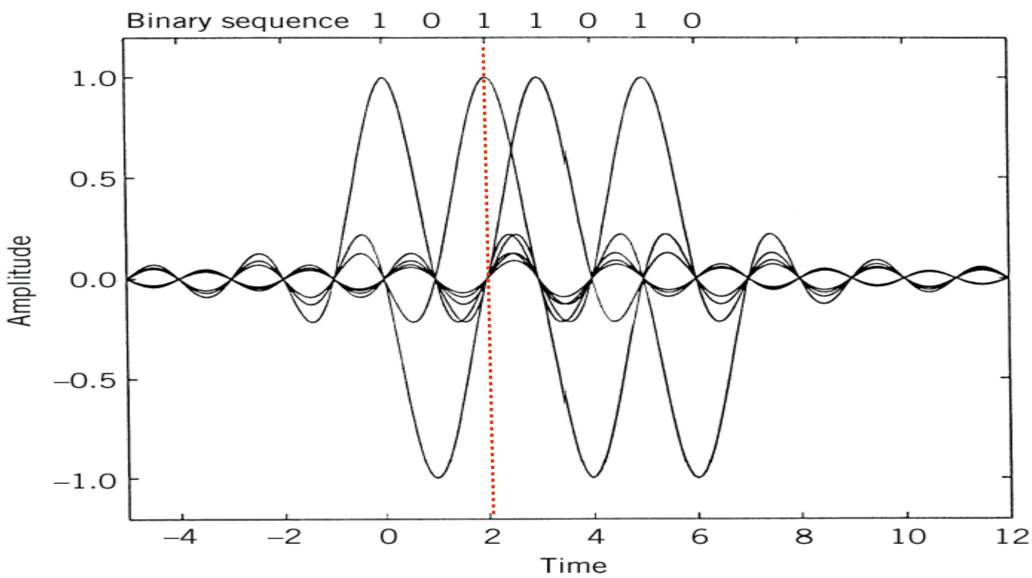
- Transmit filtering (pulse shaping):



- Implementation I (analog):

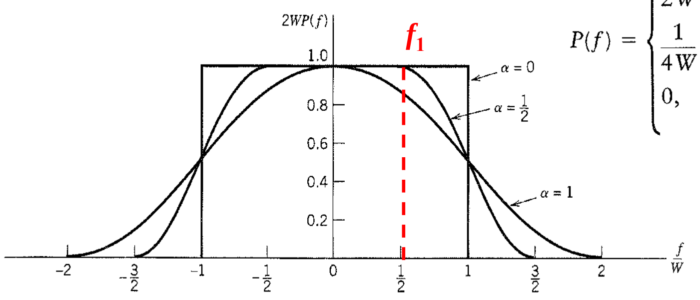


- Nyquist pulse shaping:



ISI-free pulses

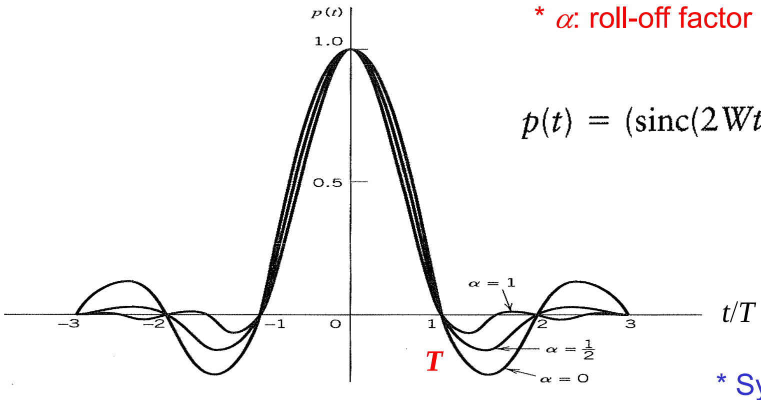
- Raised cosine (RC) pulse:



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

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

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



* α: roll-off factor

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

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

* Symbol rate=bandwidth

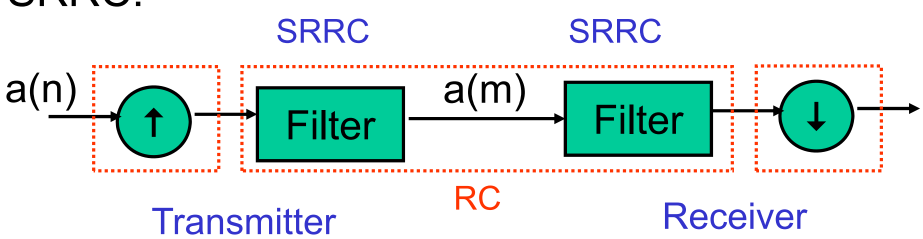
- Squared root raised cosine pulse (SRRC):

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

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

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

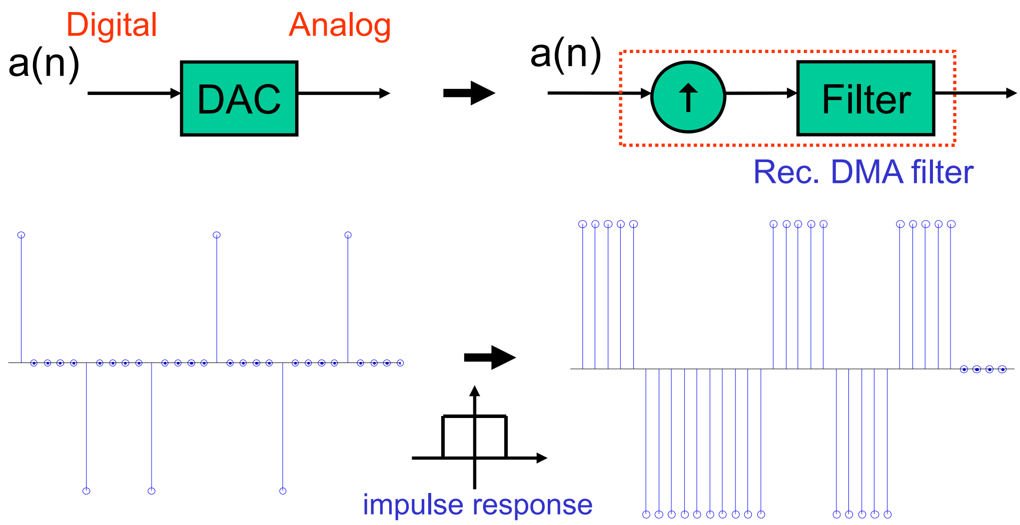
- Why SRRC:



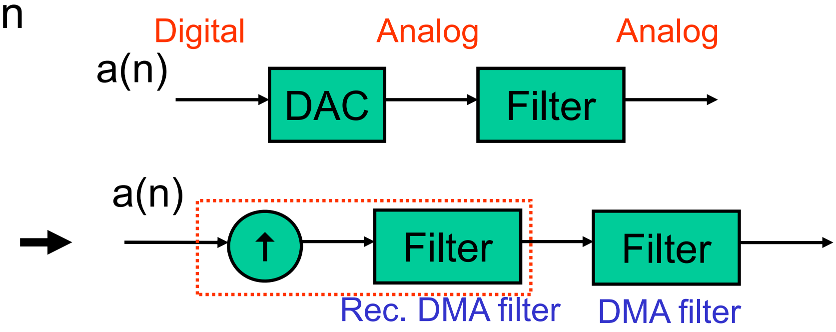
- Note that the pulse shaping filter here is an analog filter and it should have an IIR response.
- In reality, the RC and SRRC pulses cannot be generated with analog filters.
- For the time being, we just assume that there exist filters that can generate RC and SRRC pulse.

- Practice 1:
  - Plot a RC pulse and see its spectrum
  - Plot a SRRC pulse and see its spectrum
  - Check if the convolution of a SRRC pulse and another SRRC pulse will give you a RC pulse.

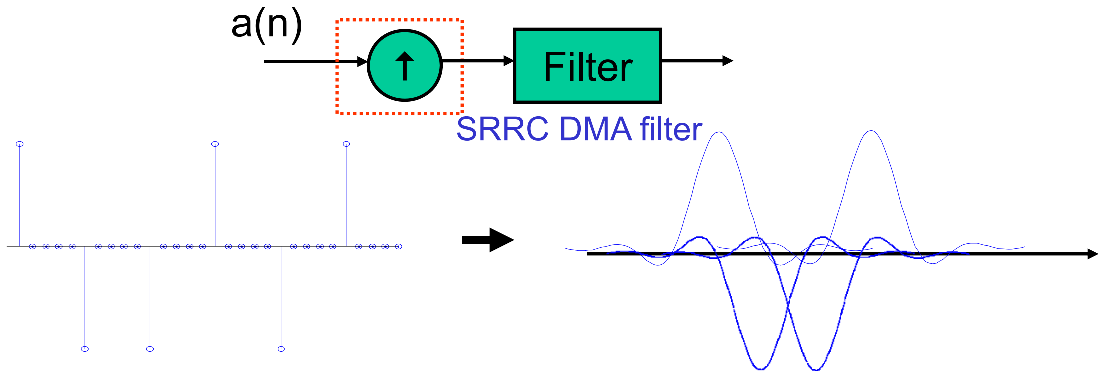
- Note that we use a digital pulse with a high sampling rate to model the analog pulse. As mentioned, we call this a digital modeled analog filter (DMA filter).
- The operation of a practical DAC can be modeled with a up-sampling operation followed by a rectangular filtering .



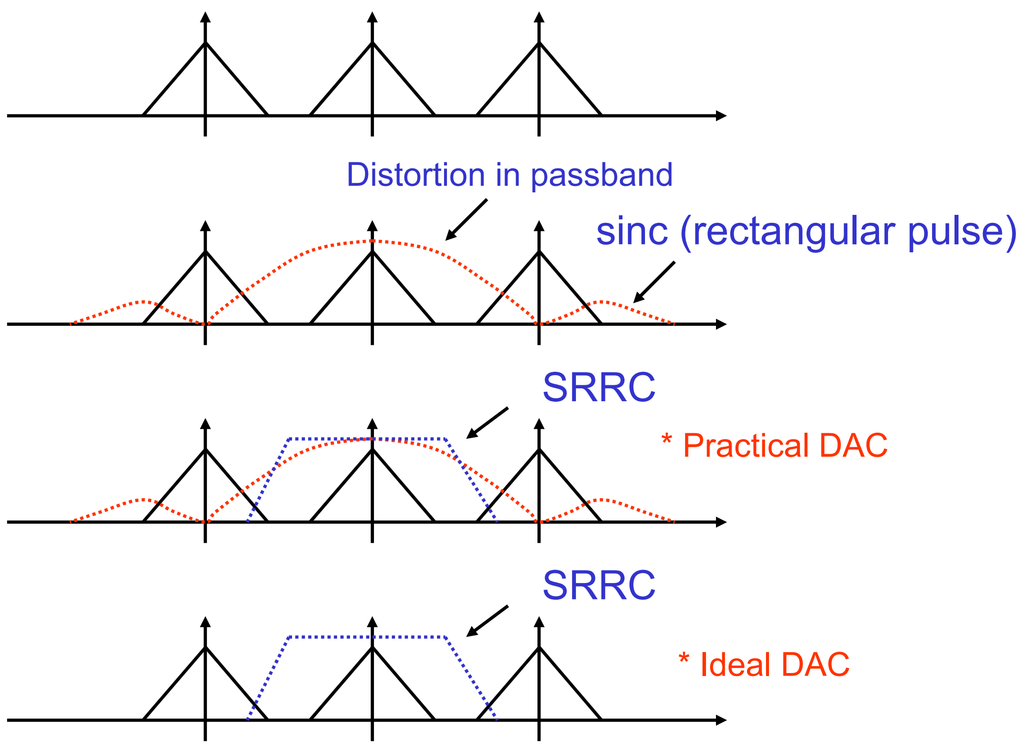
- Then



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

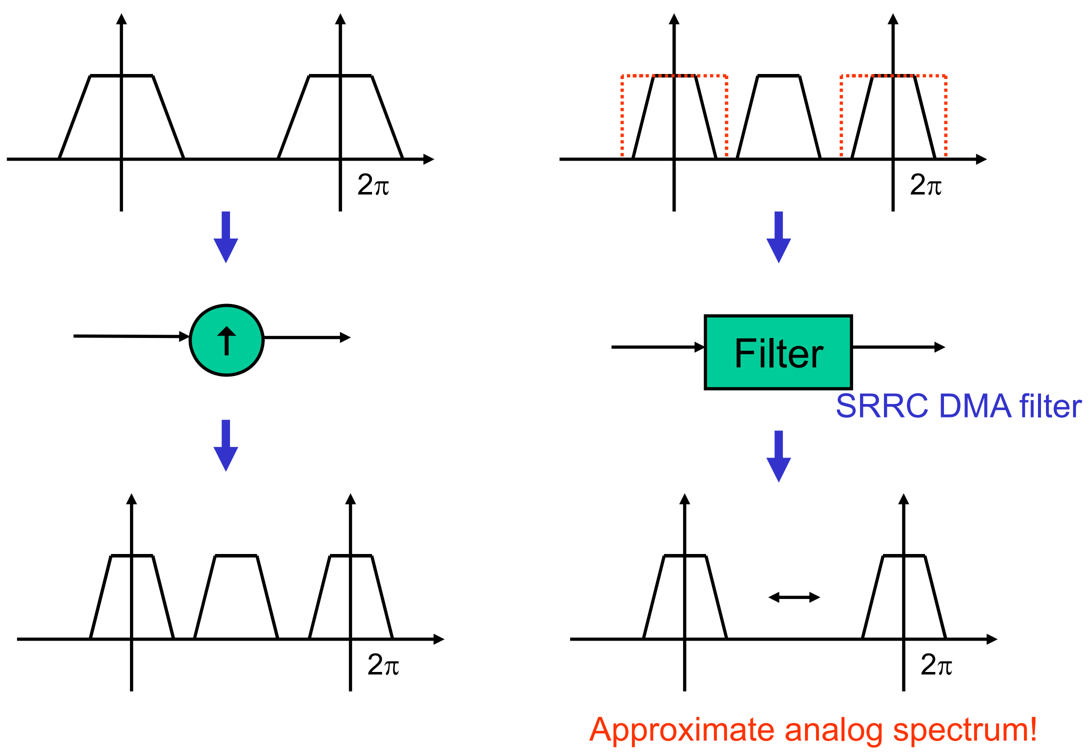


- Spectrum of practical DAC:

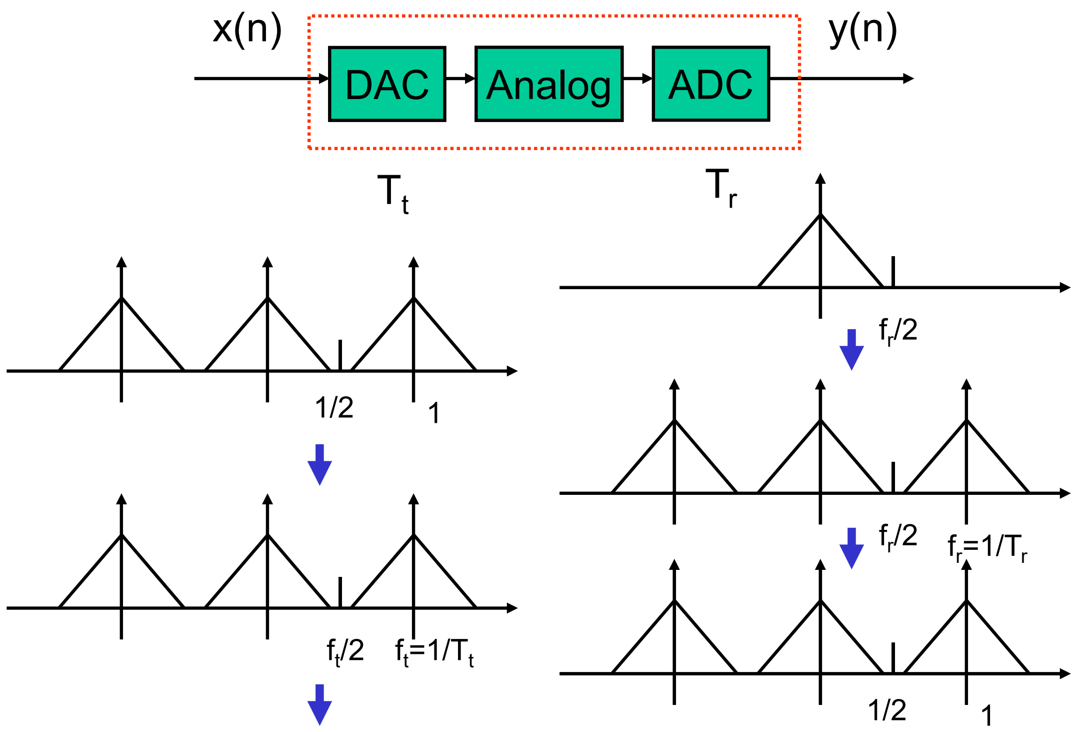


- Practice 2:
  - Generate a BPSK sequence.
  - Conduct the RC pulse shaping operation with an ideal DAC (upsampled by a factor of 32).
  - Conduct a dowsampling operation to obtain the original signal (by a factor of 32)
  - Conduct the SRRC pulse shaping operation with an practical DAC (upsampled by a factor of 32).

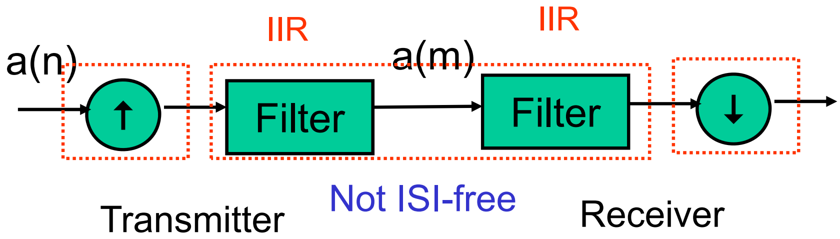
- The operation of DAC in the frequency domain:



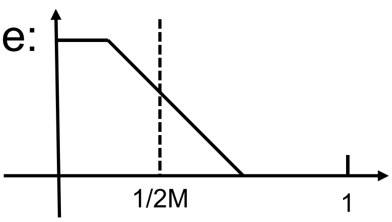
- Relationship of analog and digital frequency.



- The response of the analog LPF for the DAC can be used as the pulse shaping also.
- Unlike raised cosine pulses, the Nyquist criterion may not be perfectly met for IIR pulse shaping. As a result, intersymbol interference (ISI) may occur. To obtain an low-ISI IIR pulse may need some special design.



- Design guideline:

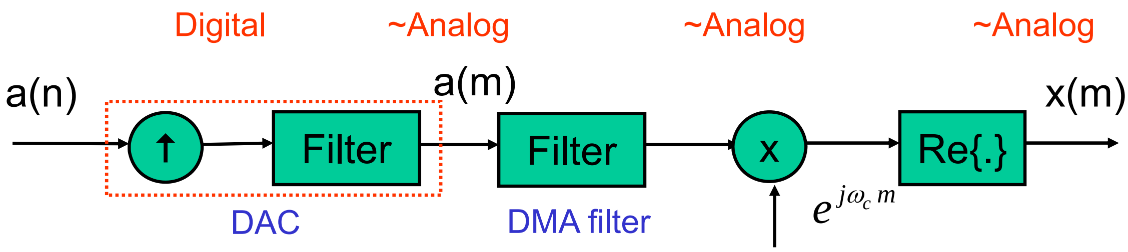


- Use Butterworth
- Mainlobe width should be around 2M wide
- Passband edge ~ 1/4M
- Stopband edge ~ 1/2M+1.5/4M
- Stopband ripple ~ -20dB

- Practice 3:
  - Design an IIR DMA filter for a upsampling process with L=32
  - Generate a BPSK sequence and use the filter as the pulse shaping filter.
  - Let the same filter be used at the receiver. Conduct the detection at the receiver. Can you have ISI free pulses?
  - Adjust the cutoff to obtain low-ISI pulses.

* Note that the delay of the IIR filter can be obtained from its group delay. Since there are transmit and receive filter, the delay has to be doubled.

- Up-conversion:

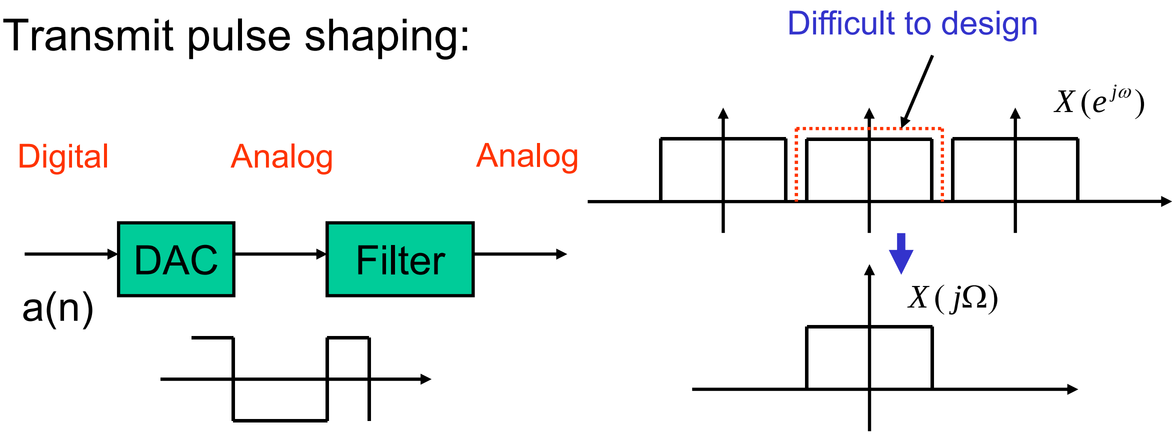


- Calculation of the carrier frequency:
  - The sampling rate is MR (R: symbol rate, MR→1).
  - If the carrier frequency is $f_{d,c}$ ($f_{d,c}$: digital frequency), then the corresponding analog frequency is $MRf_{d,c}$.
- Example:
  - Let the symbol rate be 1MHz, the upsampling factor be 100, and the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

- Homework:
  - Conduct SRRC pulse shaping for a QPSK sequence (the upsampling factor is 64).
  - Use the practical DAC.
  - 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.

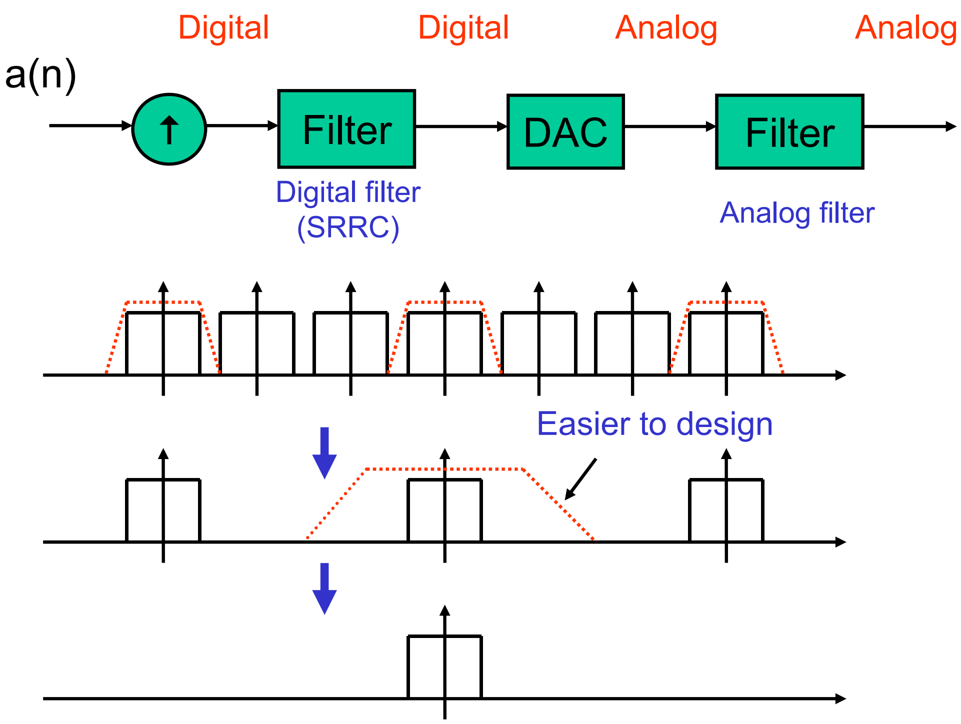
Lab. 8 Transmit Filtering/Up Conversion II

- Transmit pulse shaping:

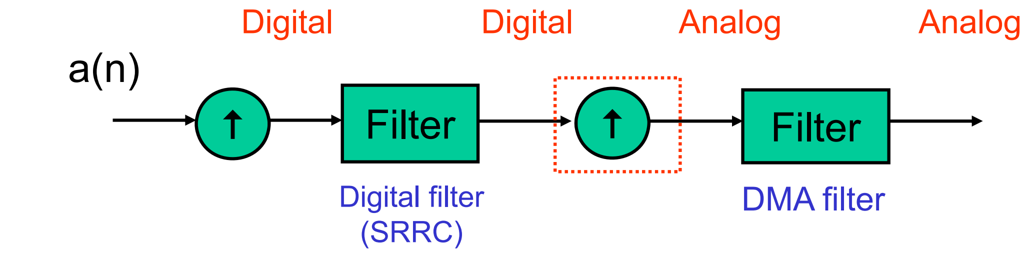


- The analog filter may be difficult to implement due to its stringent requirements.
- One way to solve the problem is to use a digital filter sharing the analog filtering operation (hybrid filtering).

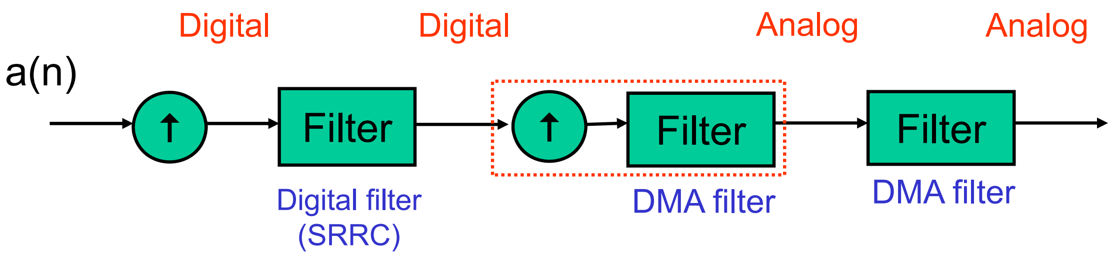
- Pulse shaping implementation II (hybrid):



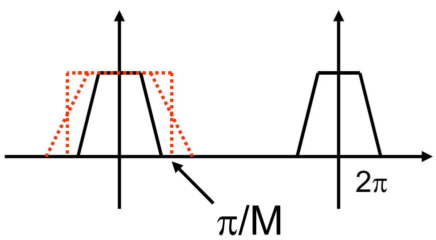
- For an ideal DAC, we then have



- 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)
- The SRRC filter can also be used as the lowpass DMA filter (analog).
  - Let M be the upsampling factor.
  - The cutoff frequency should set at π/M.



- Note that the analog SRRC filter is not realizable.

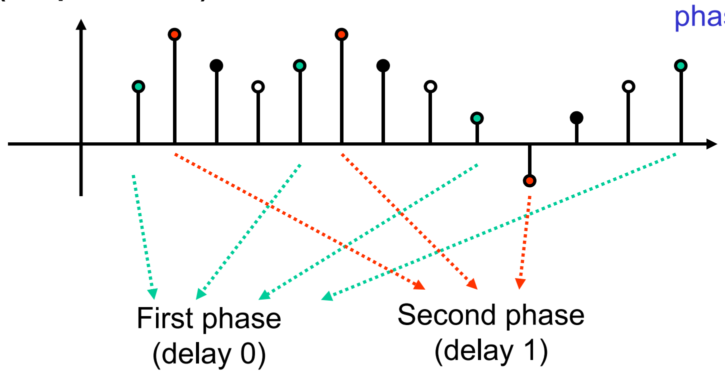
- Practice 1
  - Conduct SRRC pulse shaping (in the digital domain) for a BPSK sequence. The SRRC filter is also used as the DMA filter. The up-sampling factor for digital filtering is 4, and that for the DMA filter is 16 (use the ideal DAC).

Note:

1. Digital SRRC is upsampled by a factor of 4 and DMA SRRC is also upsampled by a factor of 4.
2. The delay in the receiver is half of the size of the RC filter.
3. The sampling phase must be adjusted.

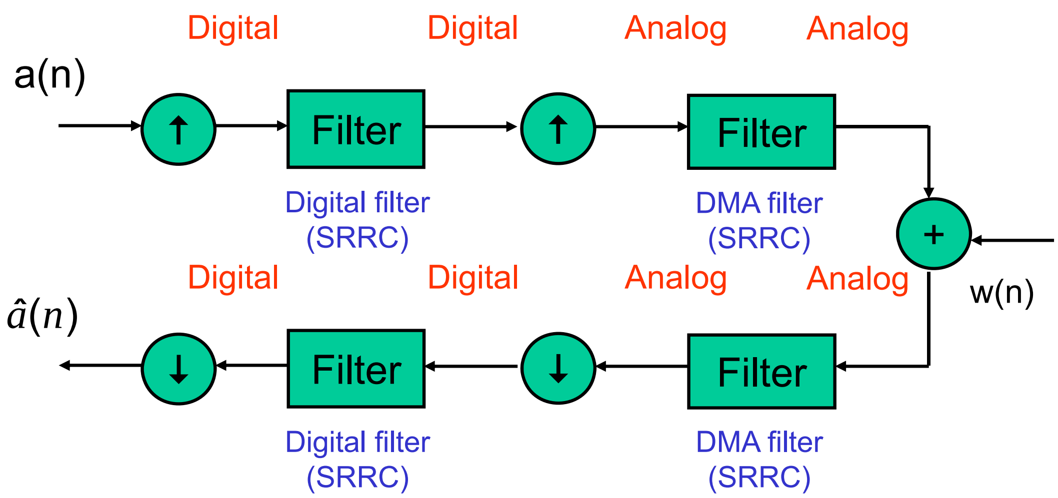
- A filter may have the non-causal property making a delay required in the output.
  - For linear-phase FIR filters, the delay is (L-1)/2.
  - For IIR filters, the delay is found by a derivative in the phase response.
- For a dowsampling with factor M, there are M possible results (M phases).

* Note that dowsampling phase is (L+1)/2.



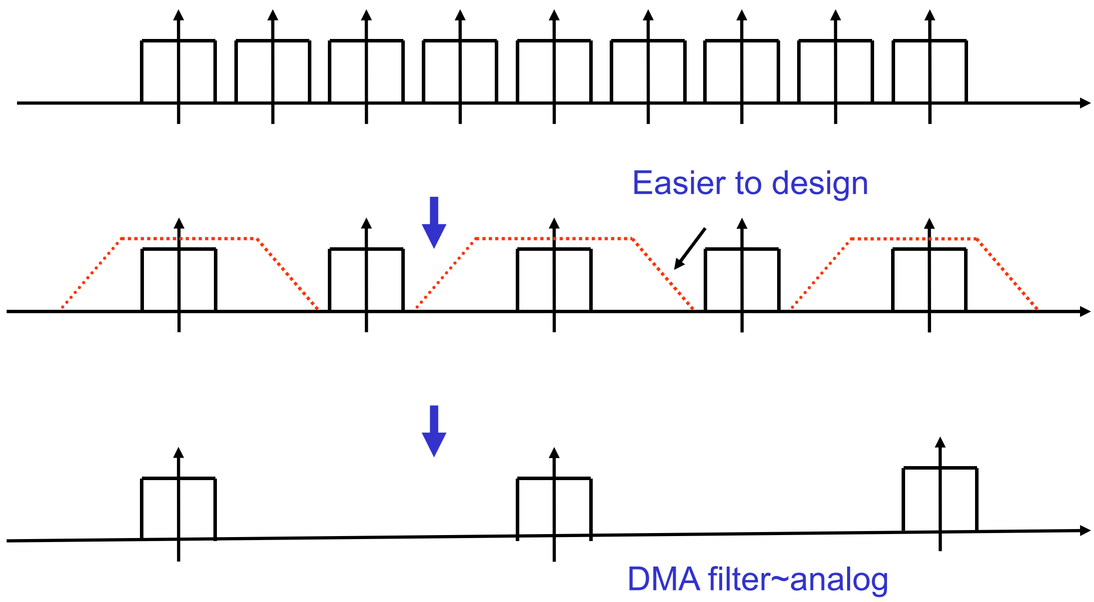
- There will be one phase giving the best result.

- Practice 2:
  - Using the problem in the previous practice, conduct the signal recovery in the receiver side.



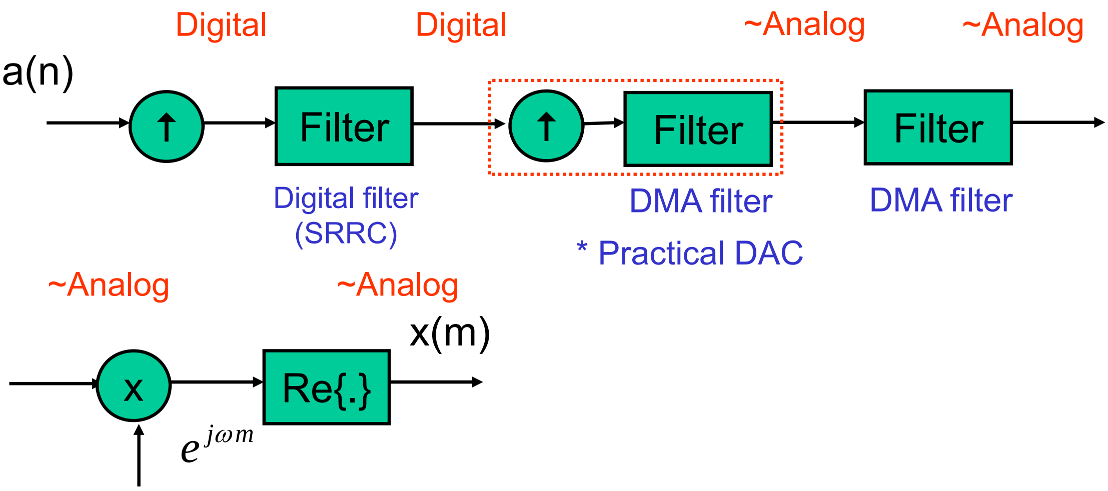
Note: You may first implement the analog (not hybrid) approach.

- 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 the previous practice.
  - Conduct the simulation again.

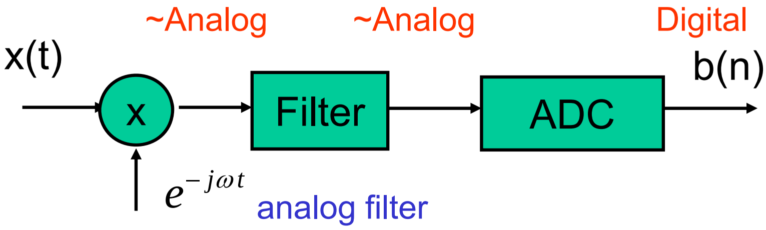
- Up-conversion:

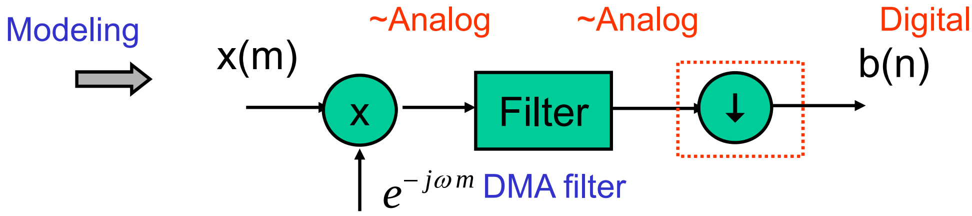


- Homework:
  - Let the symbol rate for a system be 1MHz, the sampling rate of the DAC be 4MHz, and sampling rate for DMA filter be 32MHz.
  - Let the modulation be BPSK, and the digital pulse shaping is SRRC.
  - Design an IIR DMA filter with 5 coefficients that maximizes stopband attenuation.
  - Conduct the transmit and receive operation for a sequence (without modulation).
  - Let the carrier frequency be 8MHz, and conduct the up-conversion operation for the pulse shaped sequence.

Lab. 9 Receive Filtering/Down Conversion

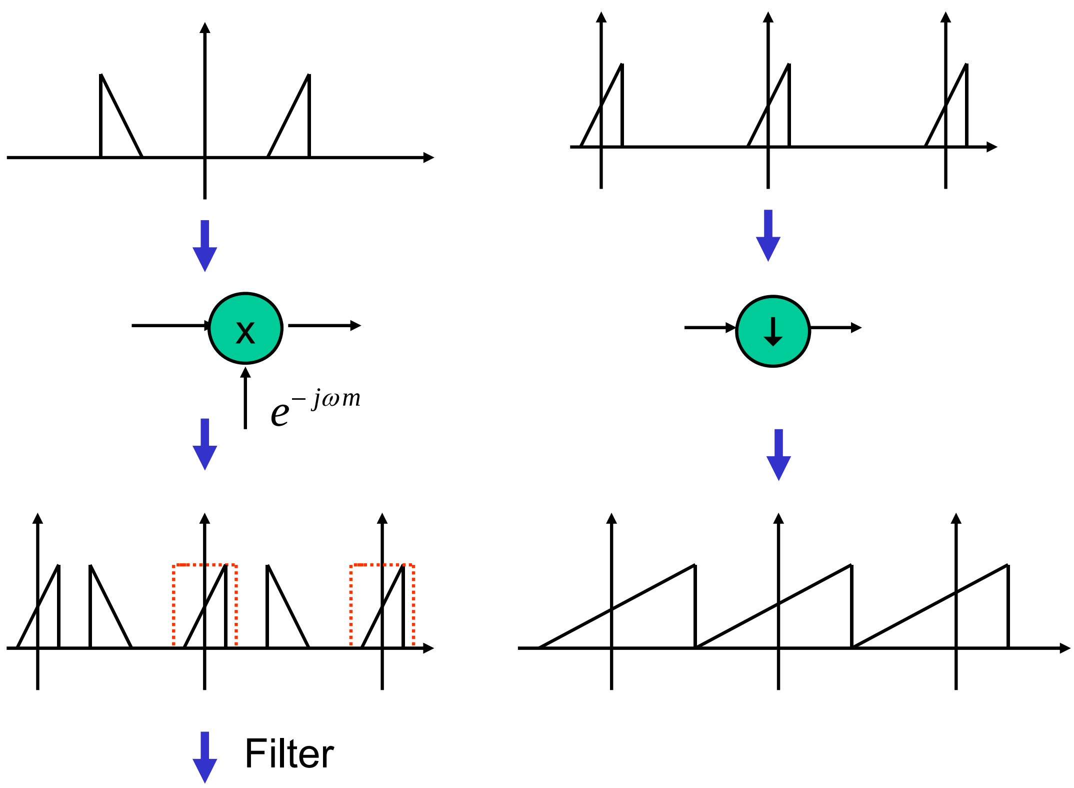
- Down-conversion (direct conversion):





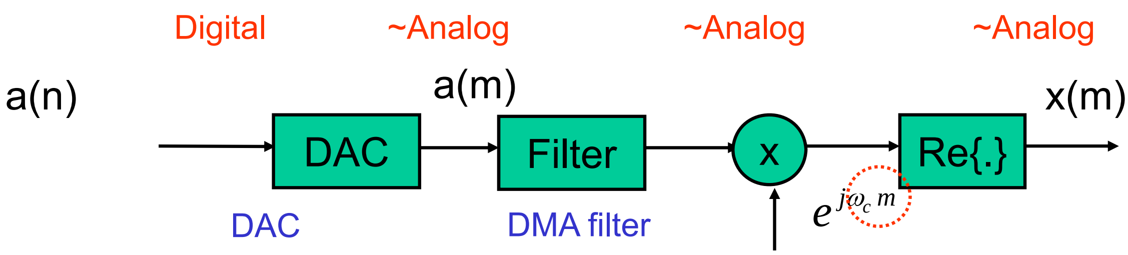
- The transition band of the DMA filter may be narrow.

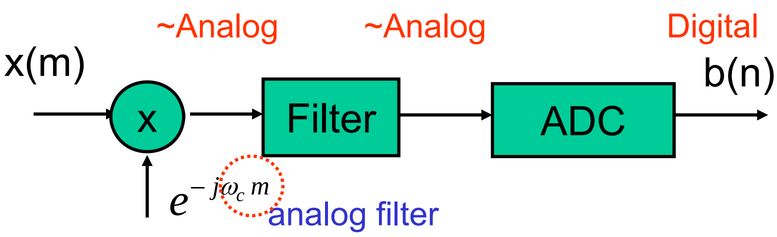
- Spectrum relationship:



- Practice 1:
  - Let the symbol rate for a system be 1MHz, the sampling rate of the DAC be 16MHz (hybrid shaping), the sampling rate for DMA filter be 32MHz, the modulation be BPSK, 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 conduct detection to recover the transmit symbols.

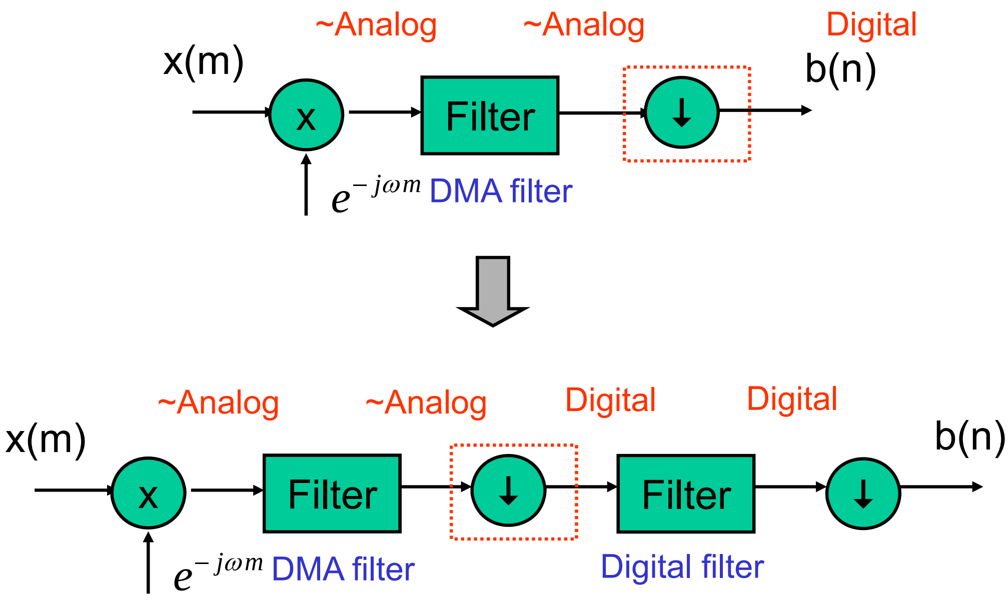
- The modulation system we consider is a coherent system meaning that the phase of the demodulating carrier must be known.





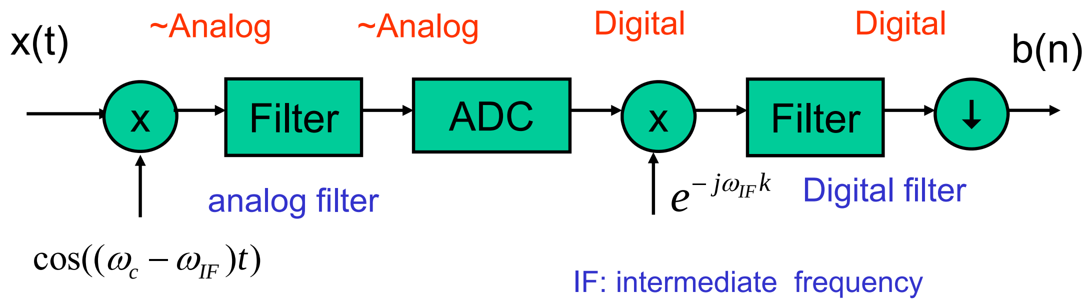
* What happens if $e^{-j(\omega_c m+\theta)}$?

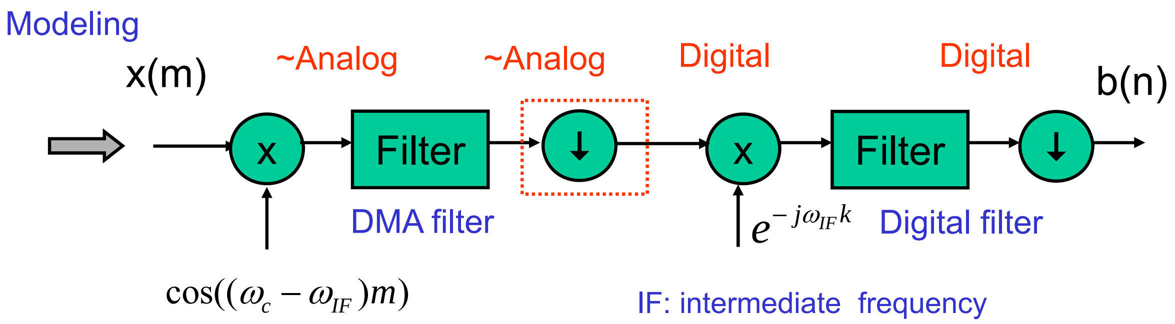
- Note that the LPF can also be implemented with a hybrid form.



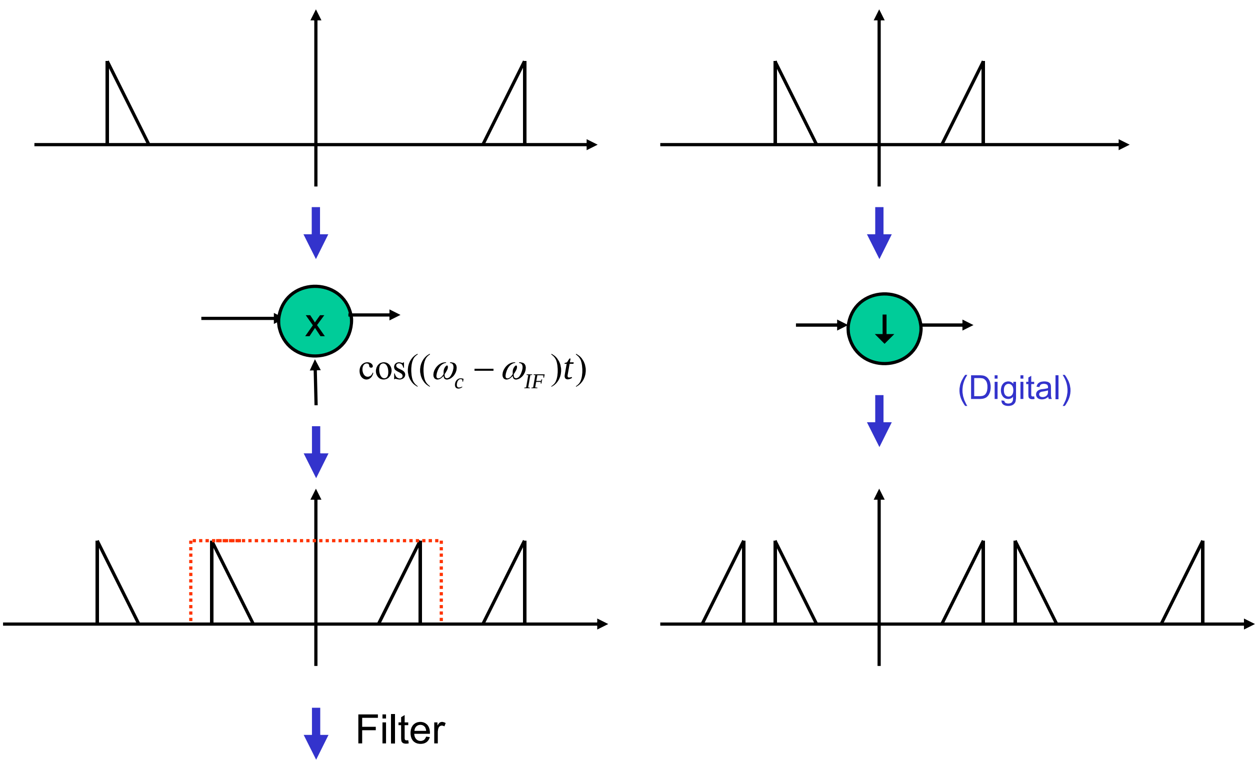
- Practice 2:
  - Using the transmitter in Practice 1.
  - Let the sampling rate of the ADC be 16MHz. Conduct direct conversion (using same filters) to down-convert the receive signal and conduct detection to recover the transmit symbols.
  - Assume there is a phase difference between the transmit and receive carrier and redo the simulation.

- Intermediate-frequency (IF) demodulation:

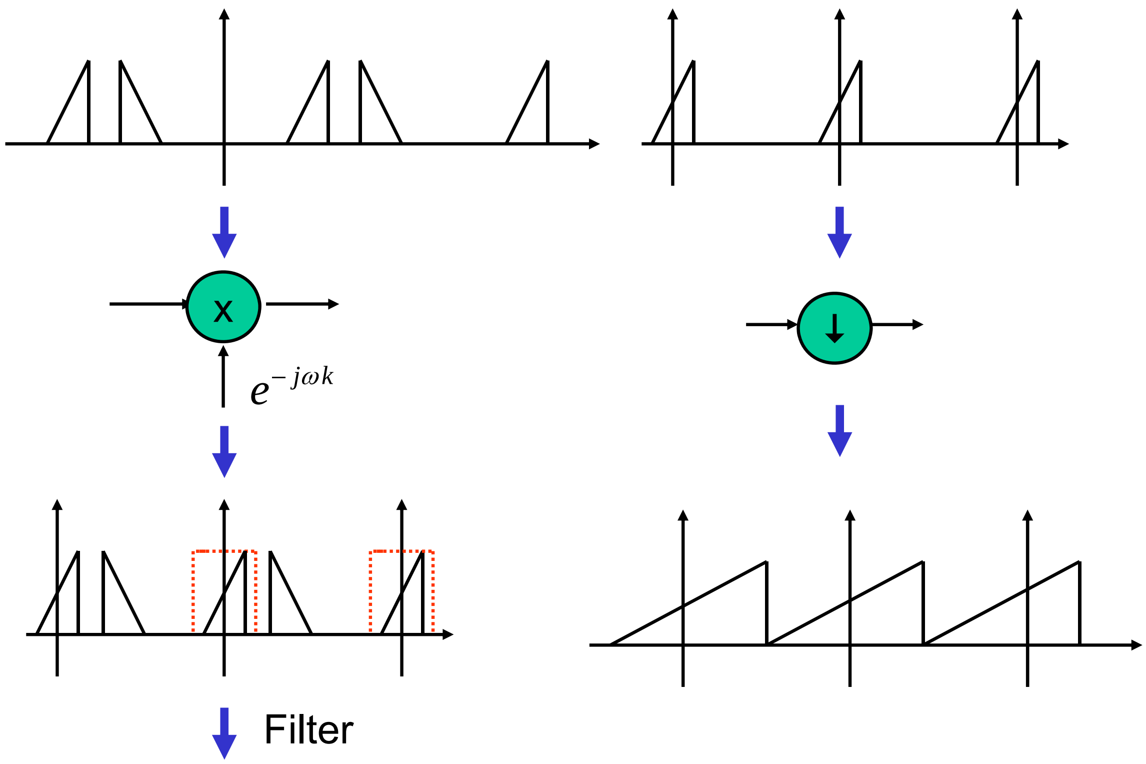




- Spectrum relationship:



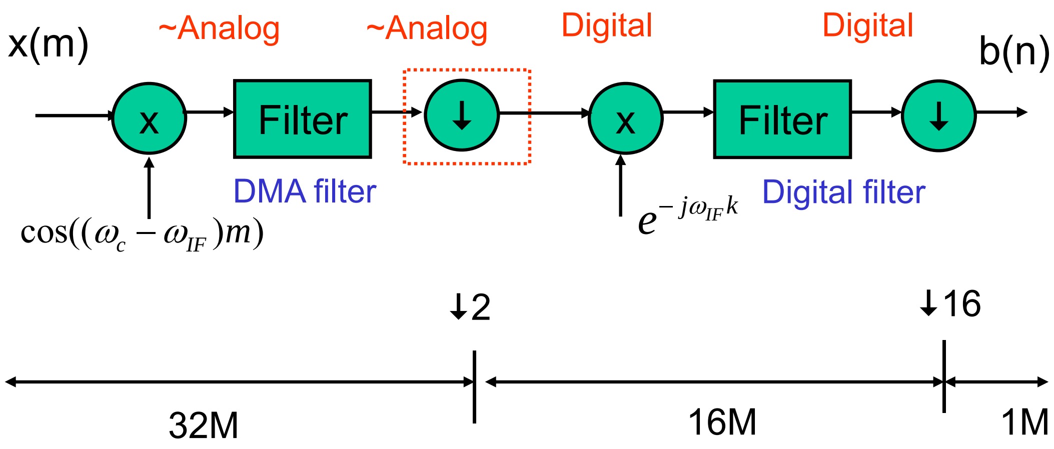
- Continue-



- What is the advantage of IF demodulation?
  - Only is one ADC required.
  - Avoid IQ mismatch.

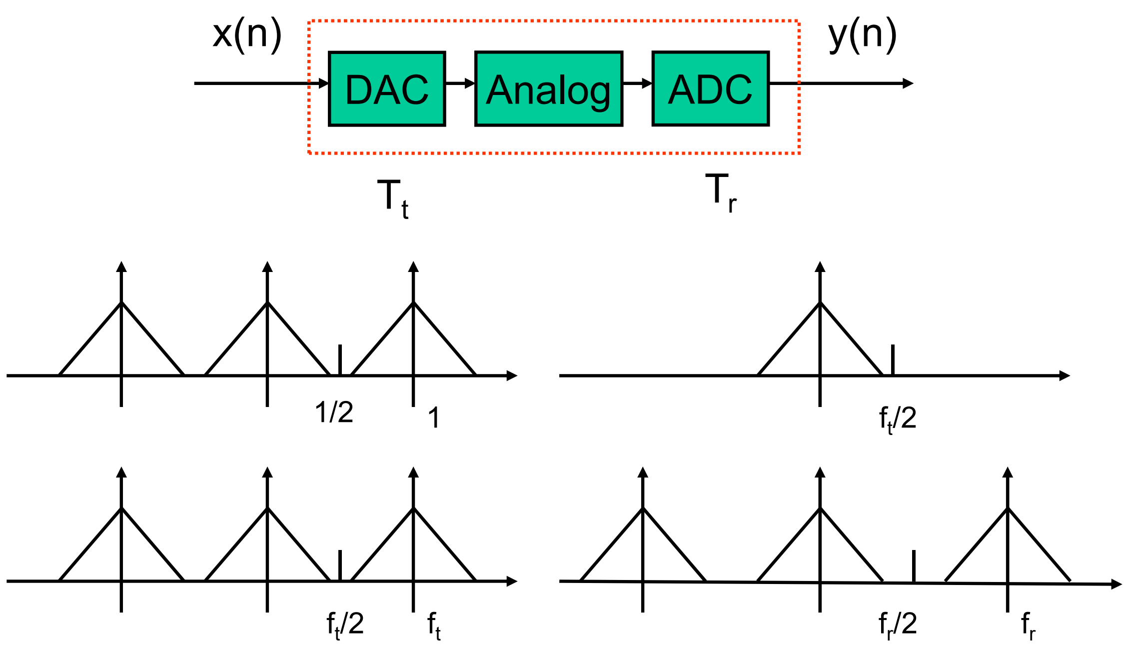
- Practice 3:
  - Let the symbol rate for a system be 1MHz, the IF frequency be 4M, the sampling rate of the ADC be 16MHz, the sampling rate for the DMA filter be 32MHz, and the carrier be 8MHz.
  - Let the modulation be BPSK, and the pulse shaping be SRRC.
  - Design a lowpass DMA filter to conduct the IF demodulation.
  - Check the spectrum of the IF band.

- Hint:

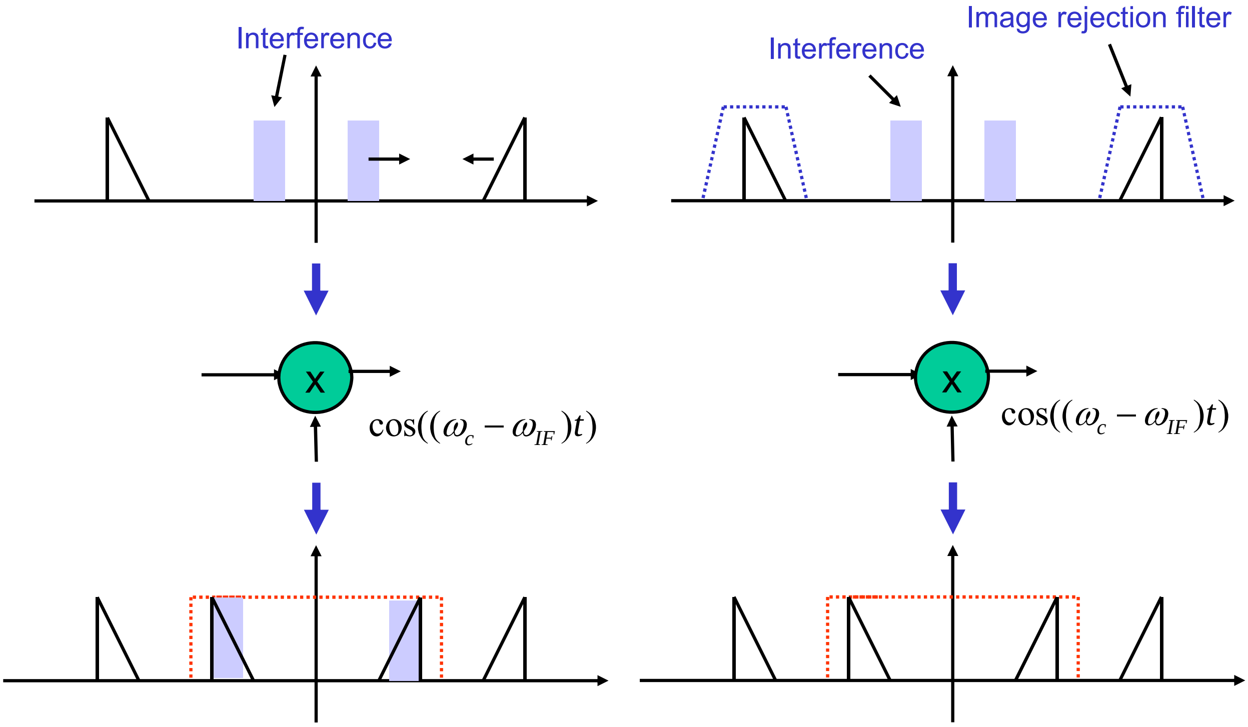


$$\cos(2\pi \frac{8\text{M-}4\text{M}}{32\text{M}} m) = \cos(\frac{2\pi m}{8}) \qquad e^{-j(2\pi \frac{4\text{M}}{16\text{M}} m)} = e^{-j(\frac{2\pi m}{4})}$$

- Relationship of analog and digital frequencies.



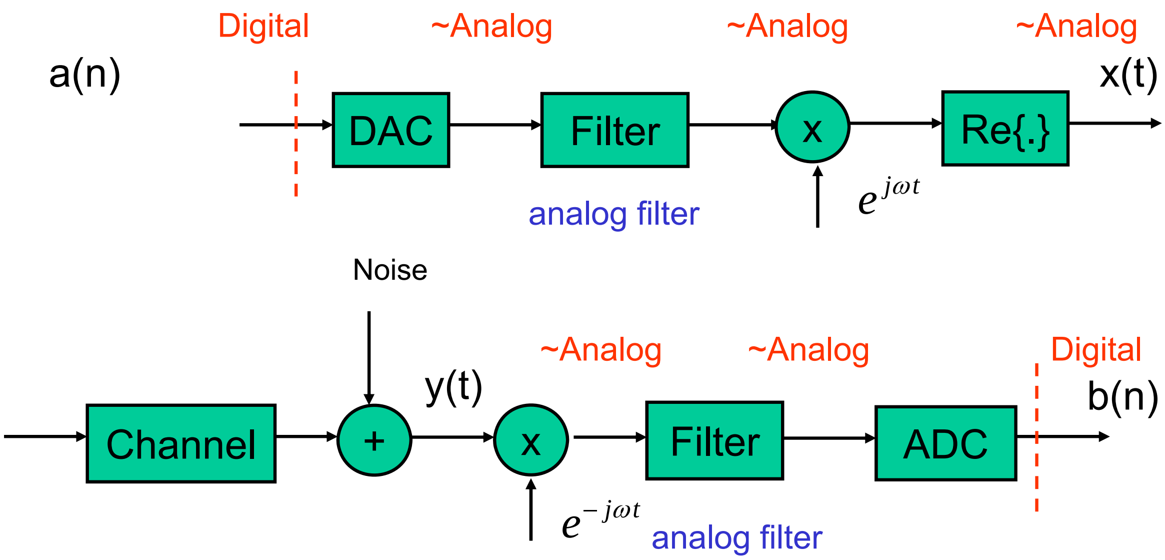
- Image problem for IF demodulation:



- Homework:
  - Let f_c be the carrier frequency and f_i be the IF frequency. Calculate the frequency of image signal.
  - Let f_c=16MHz, the symbol rate be 1MHz, and the f_i=4MHz. Design an image rejection filter.
  - Let the sampling rate of ADC be 16M and the sampling rate of the DMA filter be 64MHz.
  - Use IF demodulation to downconvert the receive signal and conduct detection to recover the transmit symbols.
  - Add noise in the receive signal and conduct the simulation again.

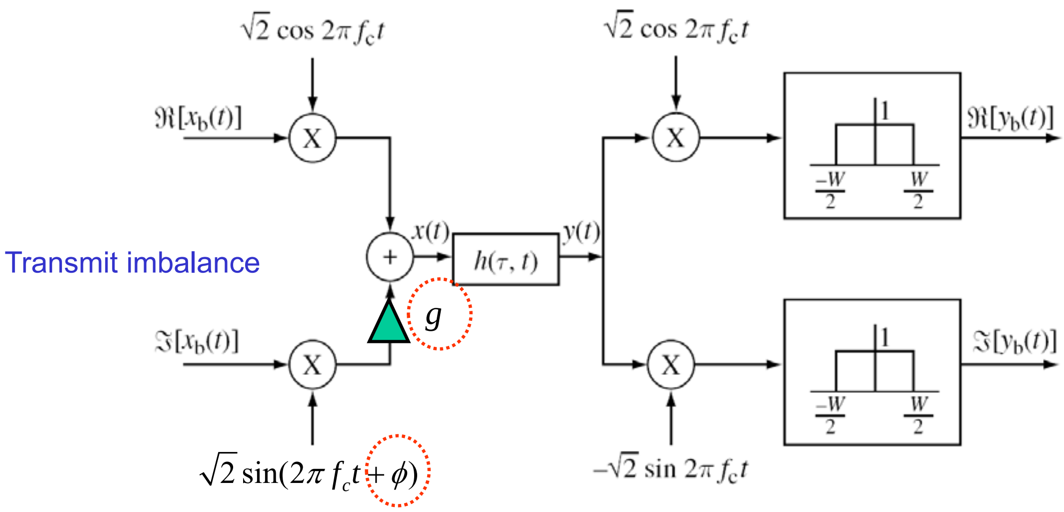
Lab. 10 RF Impairments

- 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

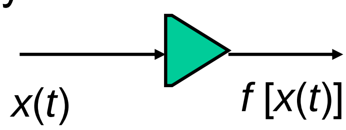


- Phase noise:

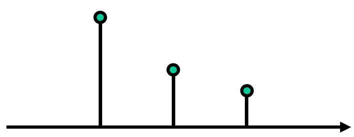
Lowpass random signal

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

- PA nonlinearity:

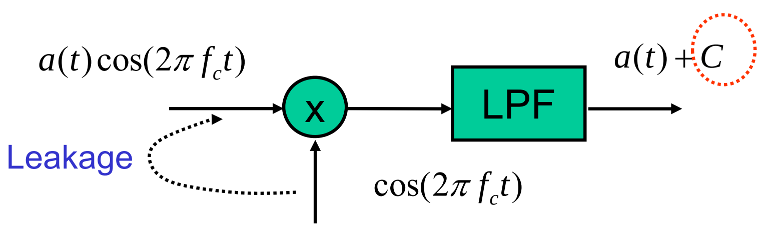


- Channel effect:

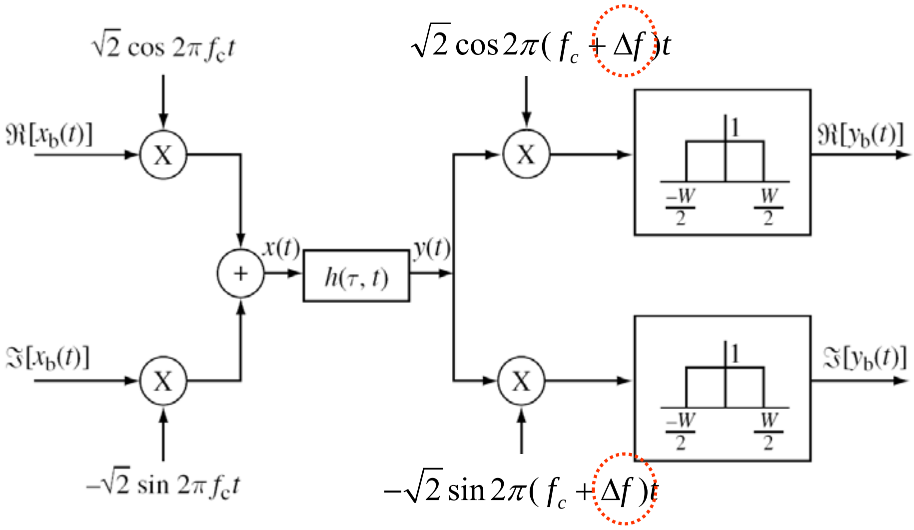


Multi-path

- DC offset:



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



- Transmit IQ-imbalance:

$$x(t) = a_I(t)\cos(\omega t) - a_Q(t) g \sin(\omega t + \phi)$$
$$= a_I(t)\frac{e^{j\omega t} + e^{-j\omega t}}{2} + jga_Q(t)\frac{e^{j(\omega t+\phi)} - e^{-j(\omega t+\phi)}}{2} \quad (\Delta)$$

- Demodulated signal (receiver does not have imbalance):

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

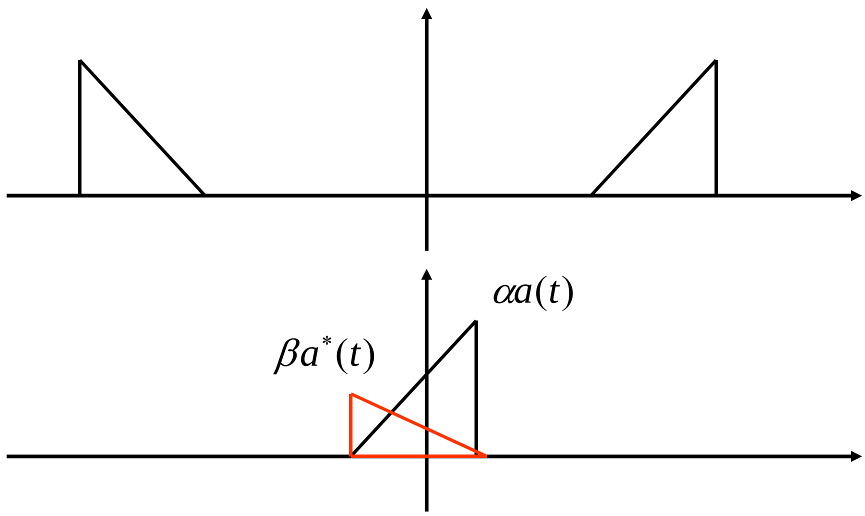
* From (Δ)

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

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

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

- 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:
  - Assume an arbitrary complex signal for transmission.
  - Use the DC architecture in Lab 9 and simulate a system with transmit IQ-imbalance and observe its effect.
  - Verify the equation on Page 6.
  - Change the transmit signal to a QPSK signal and redo the experiments.

- 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

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

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

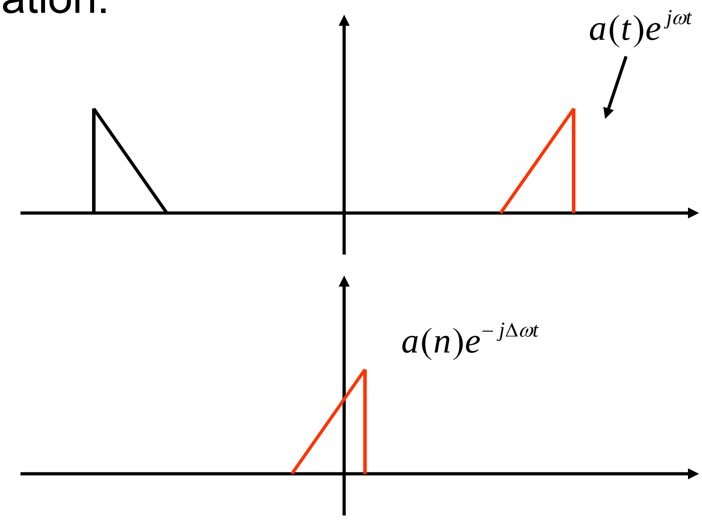
$$
\mathbf{b}(t) = \begin{bmatrix} b_I(t) \\ b_Q(t) \end{bmatrix} = \mathbf{H}^{-1}\mathbf{a}(t)
$$

- Practice 2:
  – Use the system in the previous practice.
  – Conduct the IQ-imbalance compensation in the transmitter and observe the result.

- 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.



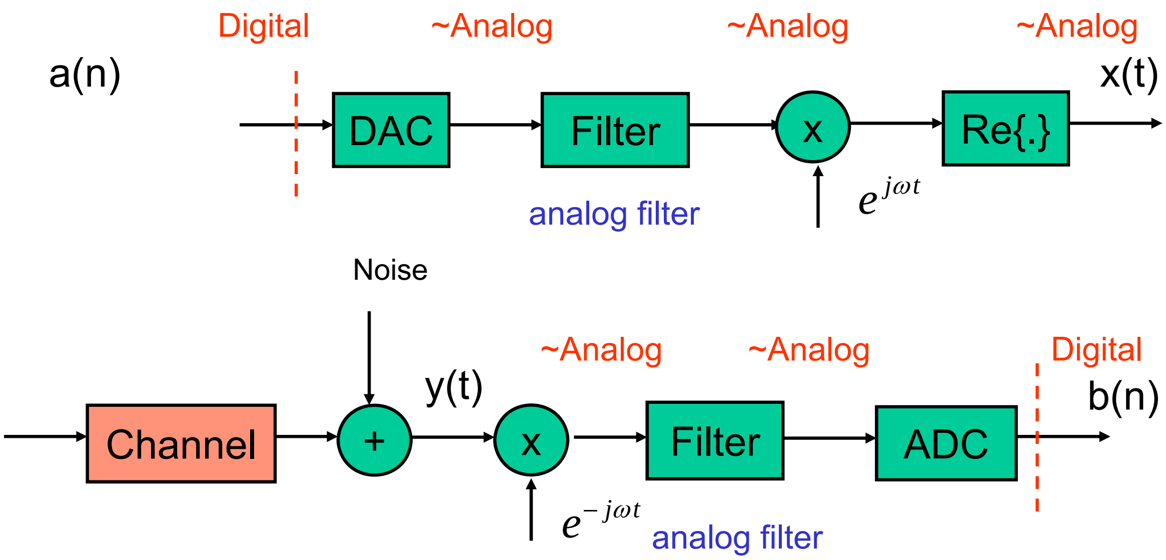
- Practice 3:
  - Use the result in Lab. 9 and simulate a receiver with CFO and observe its behavior (constellation rotation).

- Exercise:
  - Derive the effect of the transmit IQ-imbalance for the system with IF demodulation,
  - Use the system in Lab. 9, simulate the IF-demodulated system with transmit IQ-imbalance.

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

Lab 11. Equalization

- Channel effect:

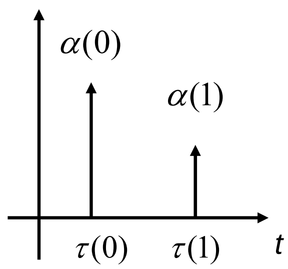


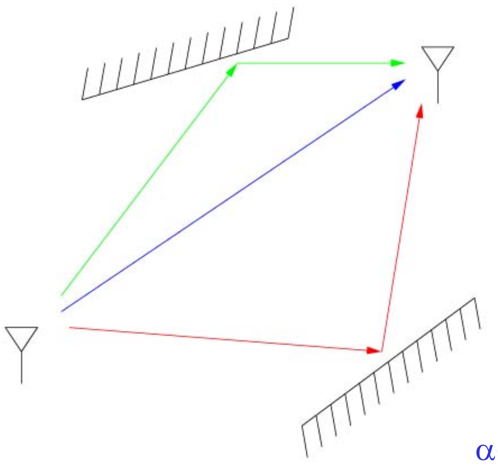
- Let

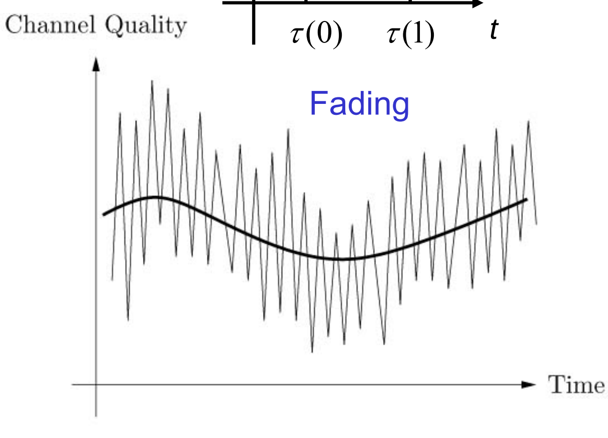
$$x(t) = \mathrm{Re}\{x_b(t)e^{j2\pi f_c t}\} \quad \text{(passband transmit)}$$
$$y(t) = \mathrm{Re}\{y_b(t)e^{2\pi f_c t}\} \quad \text{(passband receive)}$$

- Multi-path channel effect:

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







α(i): amplitude attenuation

- Then,

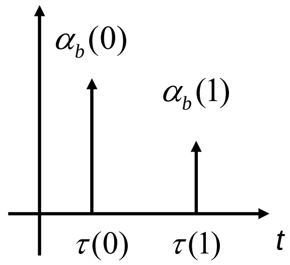
$$y(t) = \mathrm{Re}\left\{ y_b(t) e^{j2\pi f_c t} \right\} = \sum_i \alpha(i)\, \mathrm{Re}\left\{ x_b(t-\tau(i)) e^{j2\pi f_c (t-\tau(i))} \right\}$$

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

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

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 $\tau(i)=K_iT$. After the ADC, we have

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

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

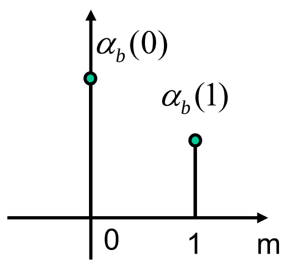
- The equivalent discrete-time signal is then

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

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

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

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

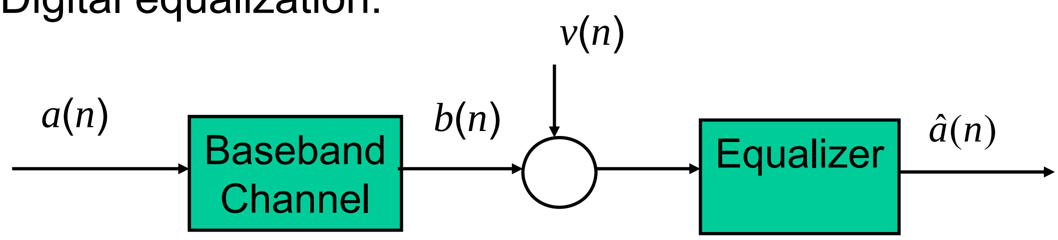


- Practice 1:
  - Use the built plateform to verify the equivalent baseband channel effect (Hint: you can transmit a digital impulse signal and observe the output signal).
  - You can let channel have two paths, and let the time delay of the second tap be a multiple of the sampling interval.
  - Try two cases. The first one is the magnitude of the first path is larger than that of the second. The other one is the magnitude of the first path is smaller than that of the second.
  - Let the time delay of the second tap is not a integer multiple of the sampling period. Observe the difference.
  - Observe that for different sampling phases (of ADC), you will obtain different equivalent channels.

$$\text{Note}: \ \alpha_b(i) = \alpha(i)e^{-j(2\pi f_c \tau(i)+\theta)} = \alpha(i)e^{-j(2\pi \frac{f_c}{f_s}\frac{\tau(i)}{T_s}+\theta)}$$

$$\text{e.g.: } f_c = 8\text{M},\ f_s = 32\text{M}, \tau(i) = 1 \text{ symbol } = 32/32\text{M} \Rightarrow \alpha_b(i) = \alpha(i)e^{-j(2\pi\times\frac{1}{4}\times 32+\theta)}$$

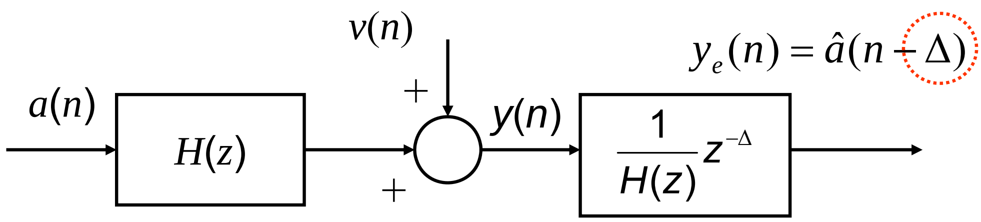
- Digital equalization:



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

$\alpha_s(i)$: sampled channel response

- Zero-forcing (ZF) equalizer:



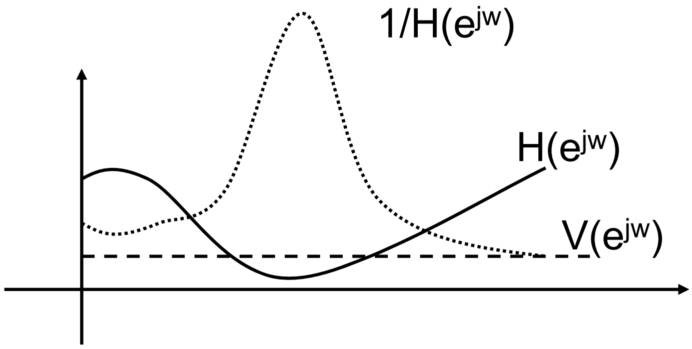
* The delay is for non-causal processing

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

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

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

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



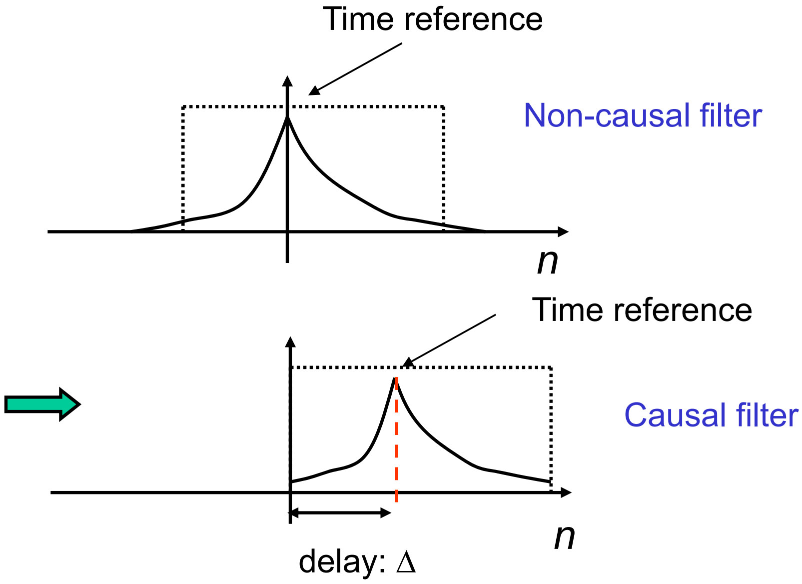
- Note that the *non-causal* processing capability is a distinct advantage of digital signal processing.
- For some H(z), the stable causal 1/H(z) does not exist. However, the stable non-causal 1/H(z) does exist.
- For example:

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

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

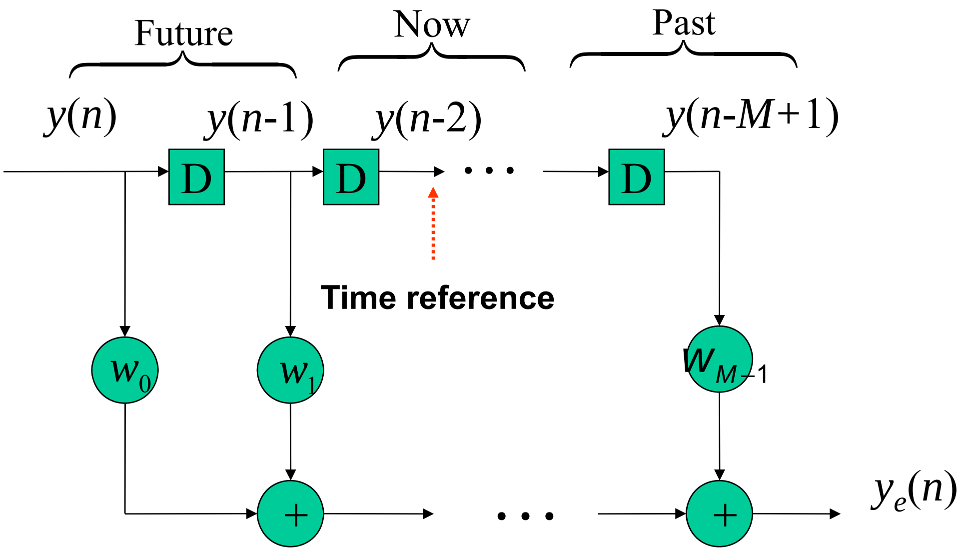
- We can realized the non-causal filter using truncation and delays.

- Non-causal FIR filtering:



- The filter output is then delayed.

- Non-causal FIR filtering:



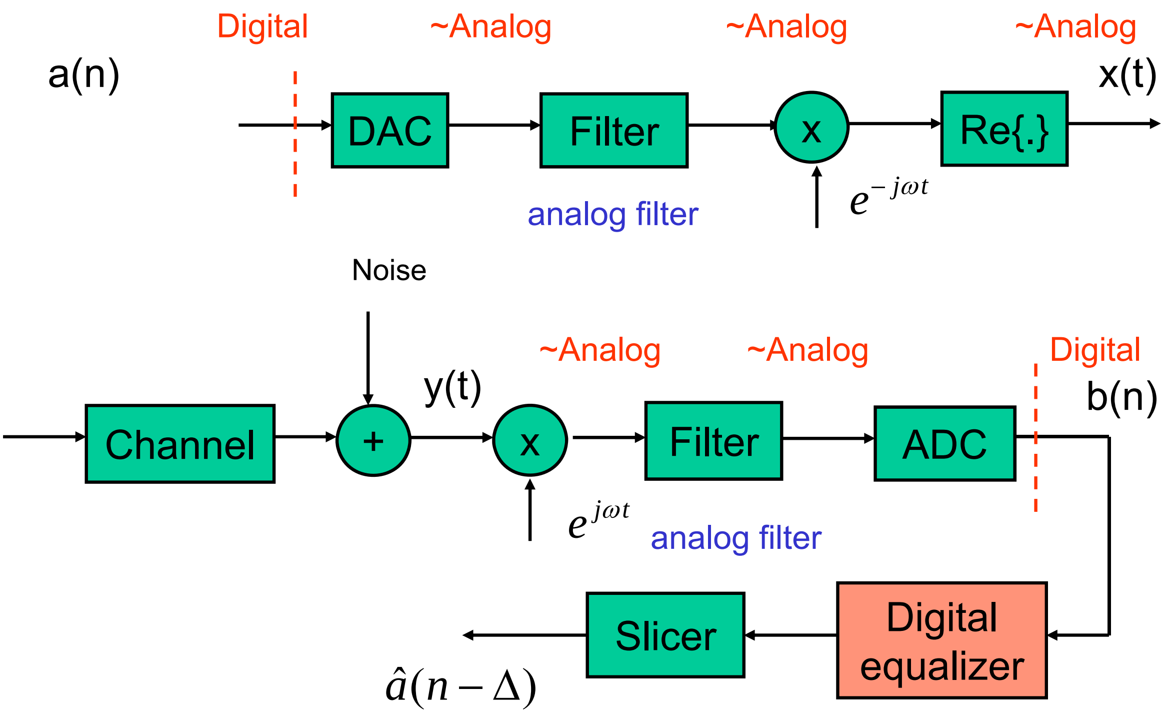
$$\mathbf{w} = [w_0 \; w_1 \; \dots \; w_{M-1}]^T,$$

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

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

- Practice 2:
  - Find the stable inverse responses of the sampled channels (with the two found equivalent channels).
  - Convolve the inverse with the equivalent channel to see if the channel is equalized.

- Digital equalizer:



- Practice 3:
  - With the built plateform and the assumed channels, conduct digital ZF equalization to recovery the transmit symbol signal.
  - If the equalizer response is non-causal, use delays and truncations to obtain a causal response.
  - Add different levels of noise and observe the equalized results.

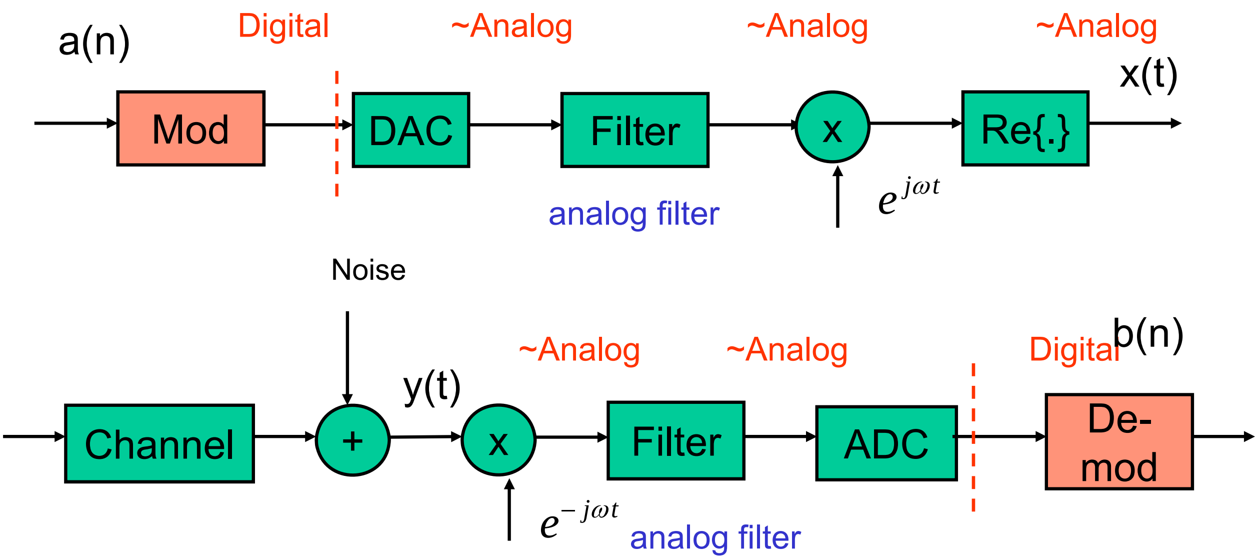
- Homework:
  - Assume that there are three paths ($\alpha(0)$=0.3, $\alpha(1)$=1, and $\alpha(2)$=0.3) in a wireless channel and the delay between any two paths is an integer multiple of the symbol period.
  - With the built plateform and the assumed channels, conduct digital ZF equalization to recovery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

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

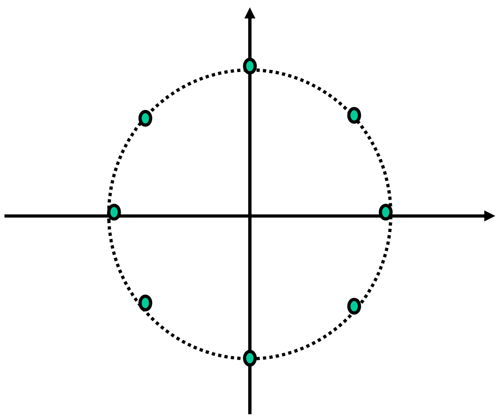
Lab 12. Constant Envelop Modulation

- Digital communication system:



- QAM:
  - QPSK (maps two bits)
  - 16QAM (maps four bits)

- PSK:
  - 8-PSK (maps three bits)



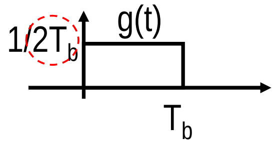
* Constant envelope

- Practice 1:
  - Use the built system to simulate a 8-PSK modulated system.
  - Compare the performance with the QPSK and 16QAM schemes.

Continuous-phase frequency shift keying (CPFSK)

- Let a PAM signal be denoted as

$$s(t) = \sum_n a_n g(t - nT_b), \quad a_n = \pm 1, \pm 3, \cdots$$



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

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

f_d: peak frequency deviation

- We can also express the signal as

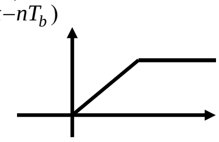
$$x(t) = \cos\left[2\pi f_c t + \theta(t)\right]$$

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

$$\boxed{h = 2 f_d T_b = 2\left(\frac{f_d}{f_b}\right)}$$

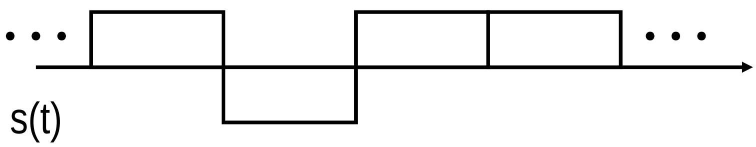
$$\theta(t) = 2\pi f_d T_b \sum_{n=-\infty}^{n-1} a_k + 2\pi(\underbrace{2 f_d T_b}_{h})\underbrace{\left(\frac{t - nT_b}{2T_b}\right)}_{q(t-nT_b)} a_n$$

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

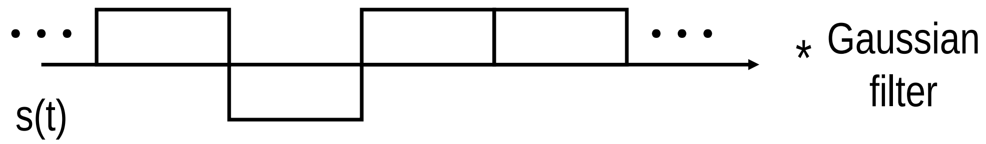


$$q(t) = \begin{cases} 0, & t < 0 \\ t/2T_b, & 0 \le t \le T_b \\ 1/2, & t > T_b \end{cases}$$

- 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.

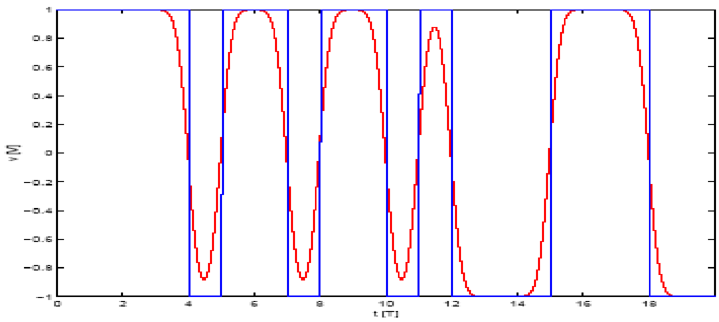


- If we apply a Gaussian filter to s(t) and then CPFSK modulation, we then have the GFSK.



- Example:

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



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

$S_f(t)$: Gaussian filtered

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

- 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 filtering
  - 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

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

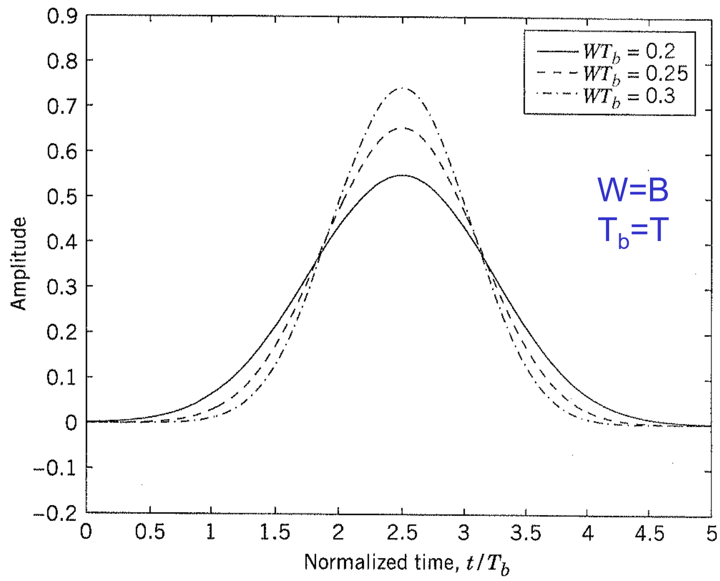
$$h(t) = \sqrt{\frac{2\pi}{\log 2}}B\exp\left(-\frac{2\pi^2}{\log 2}B^2t^2\right)$$

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

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

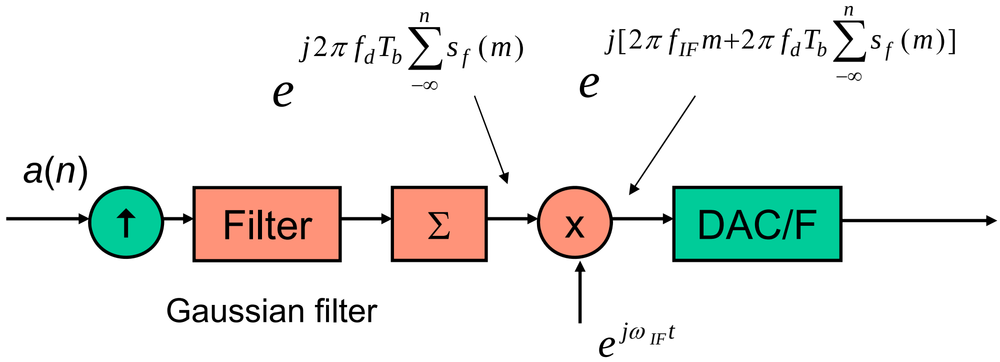
- The response can be expressed as the difference between two complementary error functions as

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

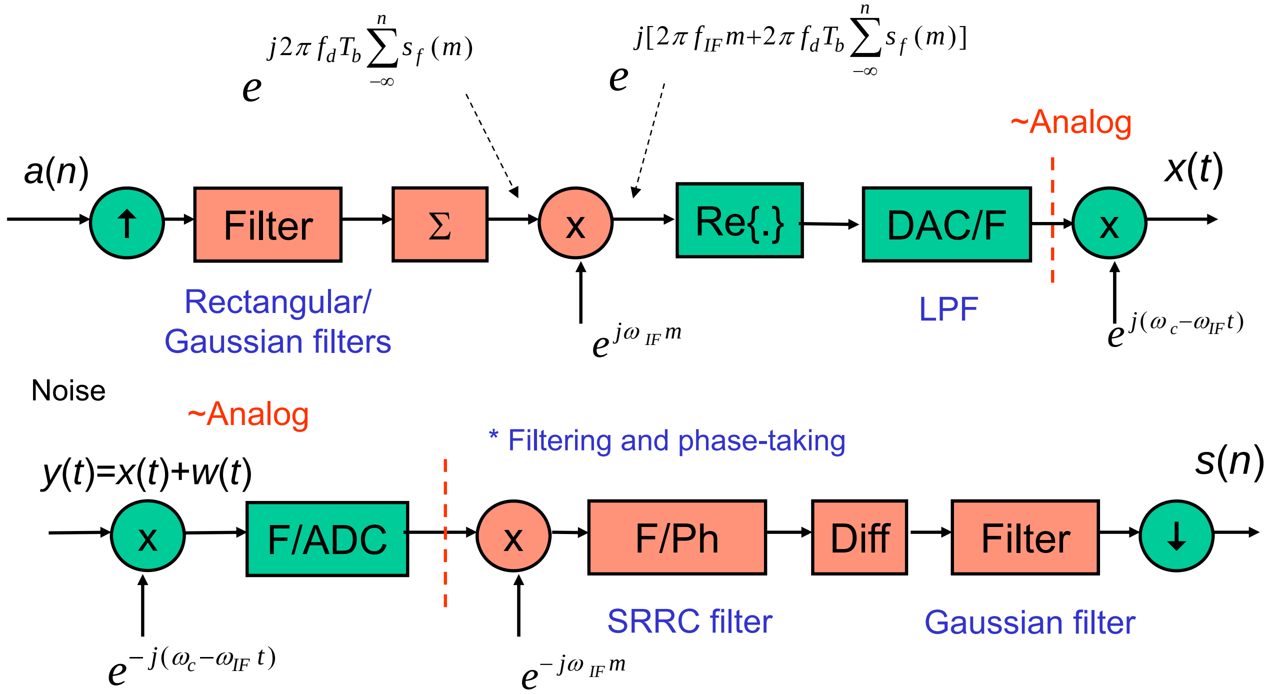


BT: design parameter

- 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



- The system:



- Practice 2:
  - Let the BT of a Gaussian filter be 0.5. Write a function to generate the sampled Gaussian filter (input: M).
  - Check the relationship of its impulse and frequency responses.

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

$$\Rightarrow h(n)=h(nT)=C\exp\left(-\frac{2\pi^2}{\log 2}\left(\frac{B}{f_s}\right)^2 n^2\right)$$

$$=C\exp\left(-\frac{2\pi^2}{\log 2}\left(\frac{B}{Mf_b}\right)^2 n^2\right)=C\exp\left(-\frac{2\pi^2}{\log 2}\left(\frac{1}{2M}\right)^2 n^2\right)$$

C: normalization constant, $\quad * \ \frac{B}{f_b}=BT$

M: oversampling factor

- Practice 3:
  - Let the symbol rate of a GFSK system be 1MHz, f_d be 150KHz, f_{IF} be 2MHz, BT=0.5, the sampling rate of the DAC/ADC be 16MHz, the sampling rate for analog signal be 64MHz. Also let the symbol be BPSK.
  - Design a proper DMA filter.
  - Construct the transmitter and the receiver (ignoring the analog carriers).
  - Observe the spectrums of modulated signals and recover signal.

$$\theta(nT_b+t)=\theta(nT_b)+2\pi f_d T_b\left(\frac{t}{T_b}\right)a_{n+1}$$

$$\Rightarrow \theta(nT_b+\frac{m}{M}T_b)=\theta(nT_b)+2\pi\left(\frac{f_d}{f_b}\right)\frac{m}{M}a_{n+1}=\theta(nT_b)+2\pi m\left(\frac{f_d}{Mf_b}\right)a_{n+1}$$

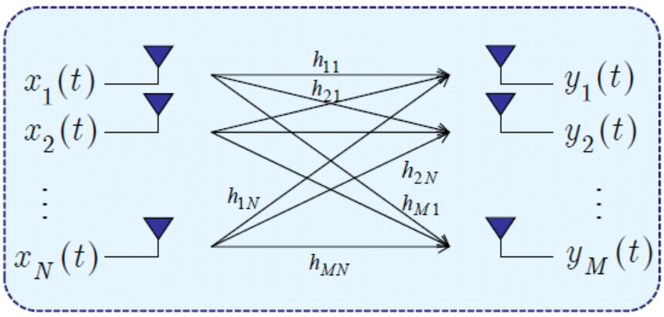
* M: oversampled factor

- Homework:
  - Add noise and observe the performance of the system constructed in Practice 3.
  - Calculate the modulation index of the system.
  - Change the modulation index and observe the performance of the system (with respect to the modulation index).

Lab. 13 MIMO Transmission

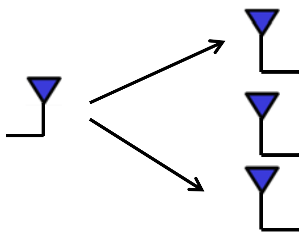
- Conventional communication systems only have one transmit/receive antenna.
- With multiple transmit/receive antennas, a multiple-input-multiple-output (MIMO) system can be formed, and the system capacity can be effectively improved.

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



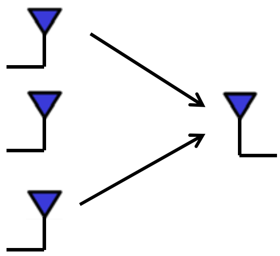
$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_M(t) \end{bmatrix}$$

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



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

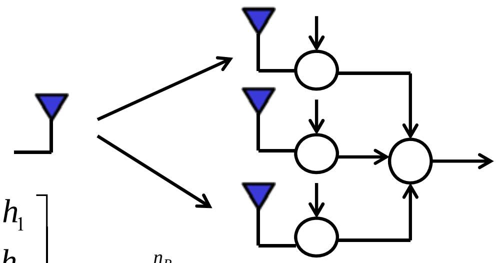
- For multiple-input-single-output (MISO) systems, i.e., n_R=1, we have



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

- The optimum transmit/receive schemes are called beamforming.

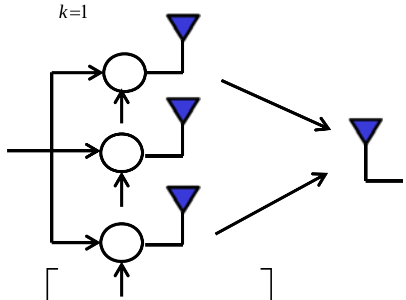
- Receive beamforming:



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

* Assume it is noise free

- Transmit beamforming:



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

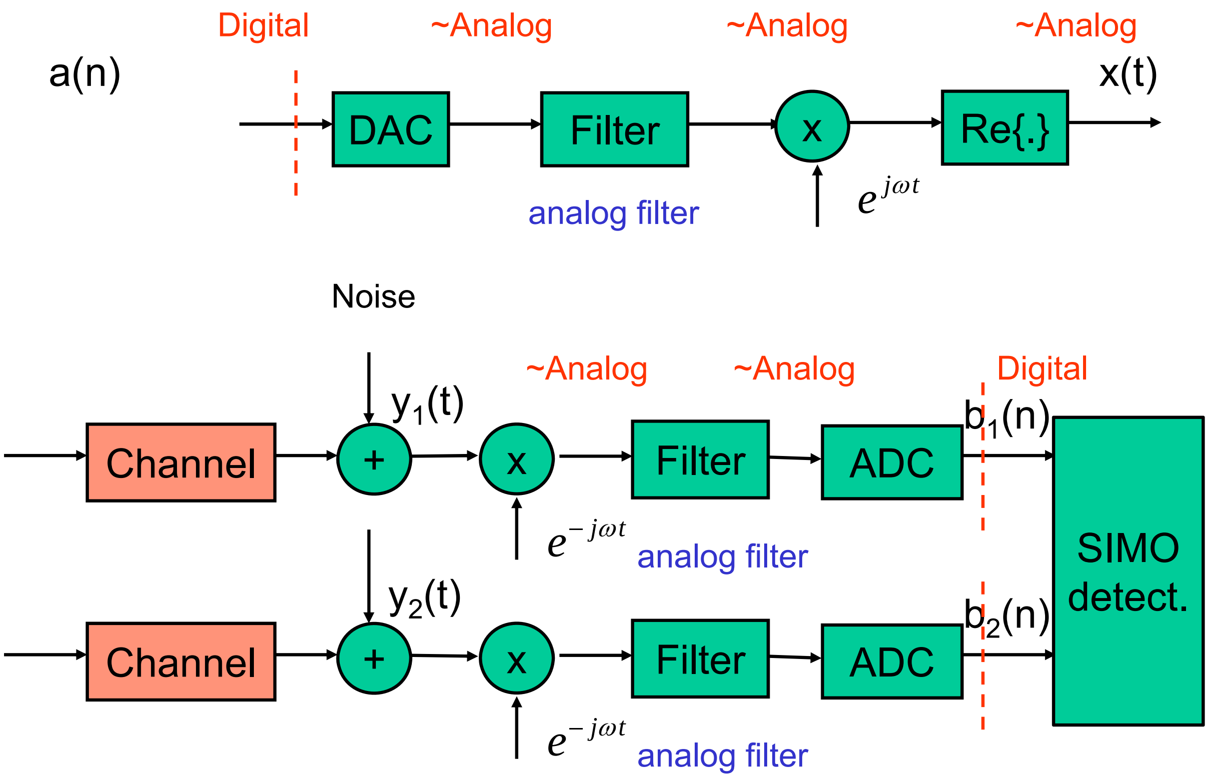
$$= \sqrt{\sum_{k=1}^{n_T} |h_k|^2} \, x(n)$$

- Practice 1:
  - Consider a 1x2 system.
  - Assume a 1x2 flat fading channel (baseband, complex).
  - Consider the baseband (digital) model.
  - Conduct receive beamforming for the system.
  - Show that the receive beamforming can achieve the best result.

- Practice 2:
  - Consider a 1x2 system.
  - Assume a 1x2 flat fading channel (passband, real).
  - Extend the built plateform (the whole link) to 1x2 system and conduct the receive beamforming for the system.

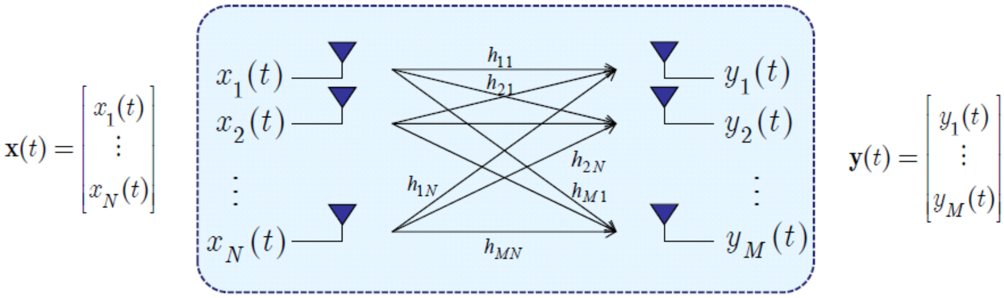
* If no channel delay, the baseband equivalent channel will be the same as that of the passband channel (i.e. real).

- 1x2 channel:



- 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:



- Then we can have

Directions

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

- 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

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

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

- 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-correlator) :

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

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

* Null interference
* Equalize the channel
* 2x2

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

- MMSE detector:

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

- The solution:

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

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

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

- The SINR:

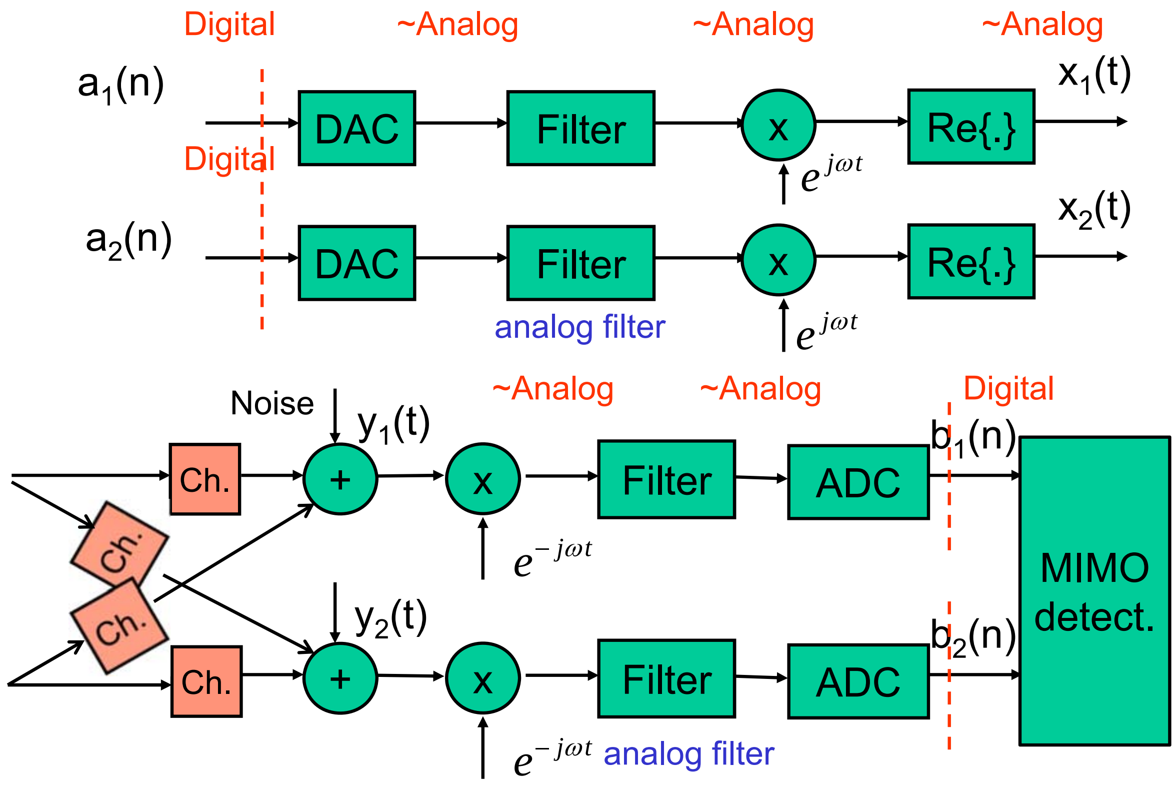
$$\gamma_k=\frac{1}{\left(\mathbf{I}+\rho\mathbf{H}^H\mathbf{H}\right)^{-1}_{kk}}, \quad \text{if } n_T \le n_R$$

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

- Practice 3:
  - Consider a 2x2 system
  - Assume a 2x2 flat fading channel.
  - Consider the baseband (digital) model.
  - Conduct the ZF/MMSE detection for the system.
  - Compare their performance.

- Homework:
  - Consider a 2x2 system.
  - Assume a 2x2 flat fading channel.
  - Extend the built plateform to 2x2 and conduct ZF detection for the system.

- 2x2 channel:



Digital
~Analog
~Analog
~Analog
$a_1(n)$
DAC
Filter
x
Re{.}
$x_1(t)$
Digital
$e^{j\omega t}$
$a_2(n)$
DAC
Filter
x
Re{.}
$x_2(t)$
analog filter
$e^{j\omega t}$
Noise
~Analog
~Analog
Digital
$y_1(t)$
Ch.
Ch.
Ch.
Ch.
+
x
Filter
ADC
$b_1(n)$
MIMO detect.
$e^{-j\omega t}$
$y_2(t)$
+
x
Filter
ADC
$b_2(n)$
$e^{-j\omega t}$
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: 1.2134×10^4

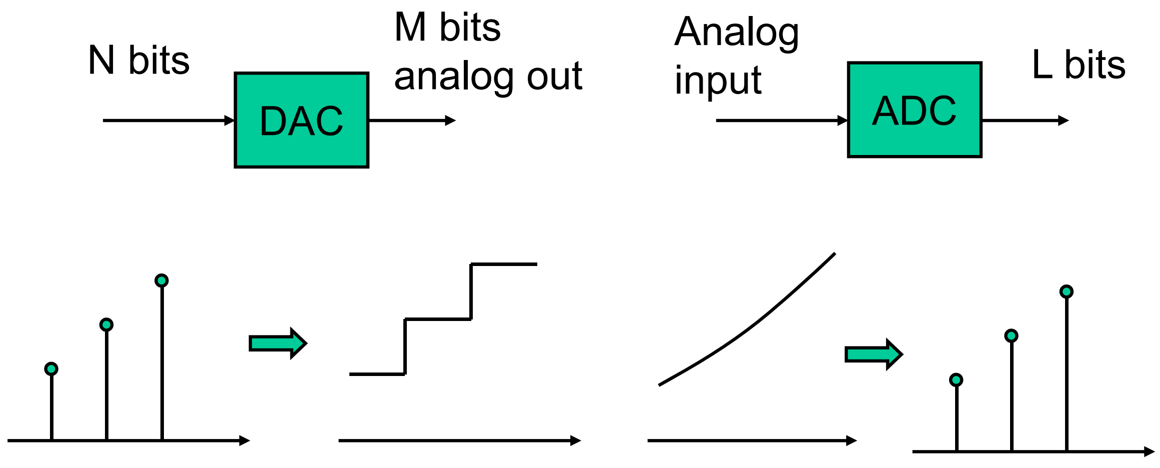
Fixed-point: 12134.1

Decimal point : fixed

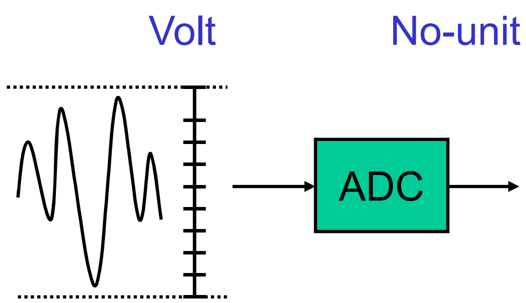
* F.P. : For a signal, its decimal point can be changed dynamically.

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

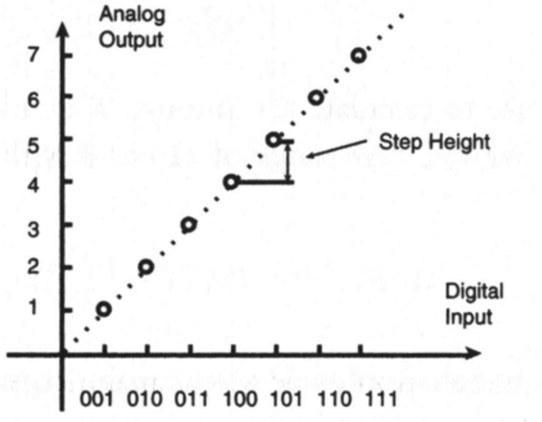
- DAC/ADC:

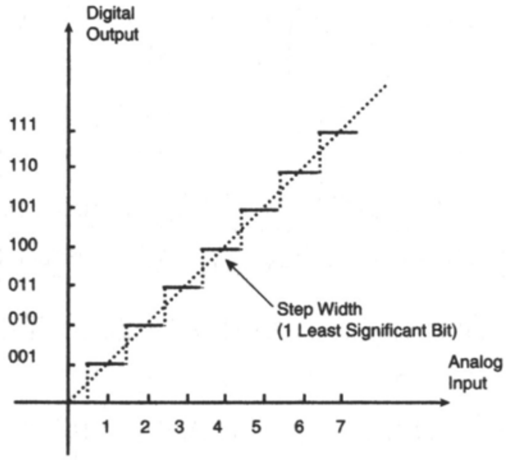


- Parameters of DAC/ADC:
  - Sampling rate
  - Number of quantization levels
  - Dynamic range

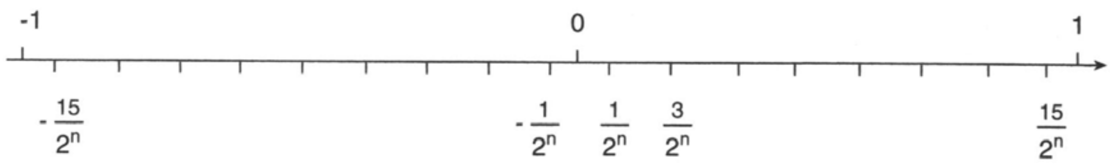


- Ideal DAC and ADC:





- Calculation of quantization 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)

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

- 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.

- Practice 1:
  - Write a function modeling an ADC (inputs: the value to be quantized, the number of bits, the dynamic range, output: the quantized result).
  - Apply the ADC to a white uniform signal ([-1,1]) and calculate the SQNR.
  - Verify the 6dB rule of thumb.
  - Apply the ADC to a white Gaussian signal and calculate the SQNR.

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

- 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

0.111

Binary point is here

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

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

01.11

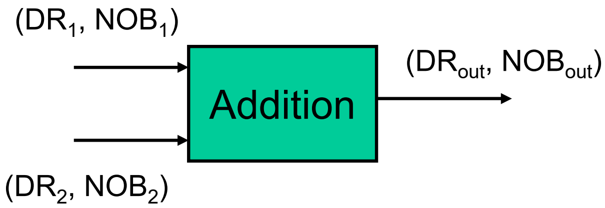
Binary point is here

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

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

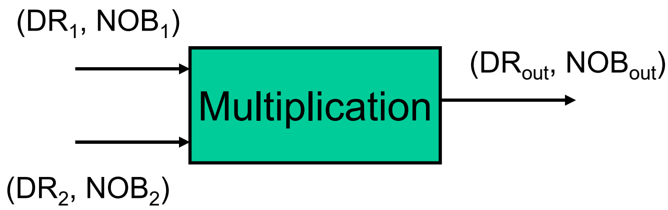
- Fixed-point operations:
  - In real circuits, addition and multiplication are two most common operations implemented.
  - What happened when two fixed-point numbers are added/ multiplied?

- 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:

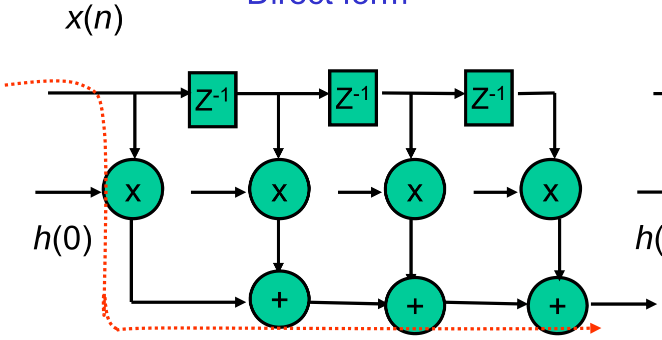


  - In general, $DR_{out}=DR_1 x DR_2$ and $NOB_{out}<NOB_1+NOB_2$ (depends on output requirement)

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

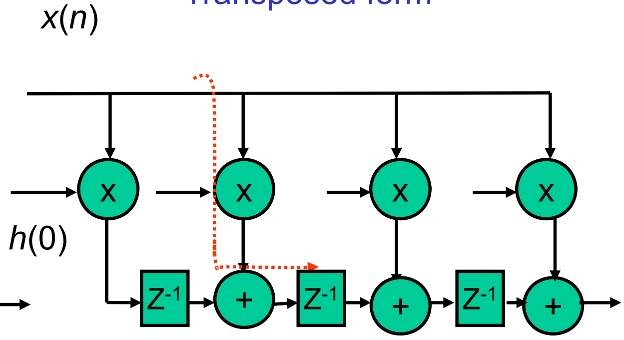
$$y(n)=\sum_{i=0}^{3} h(i)x(n-i)$$

Direct form



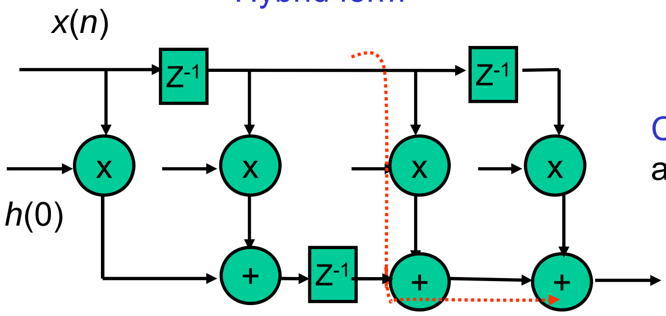
Critical path: 1 multiplier and 3 adders

Transposed form



Critical path: 1 multiplier and 1 adder

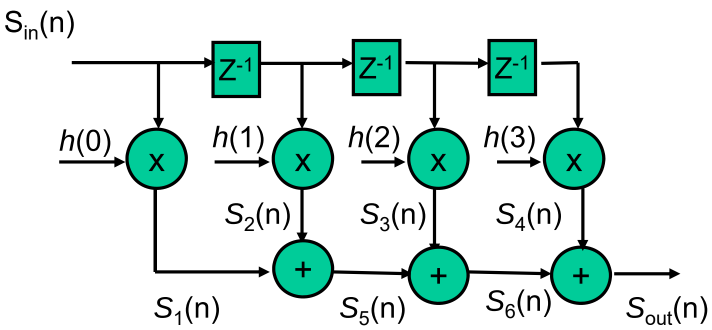
- Another architecture:

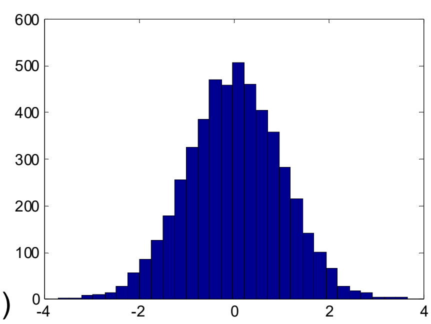


Critical path: 1 multiplier and 2 adders

- 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:
  - Choose an architecture for a filter with four coefficients.
  - Assume that the input is a white Gaussian signal with variance of one.
  - Conduct the floating-point filtering operations with the architecture.
  - Conduct simulations to find the DR for each signal (observe histograms).

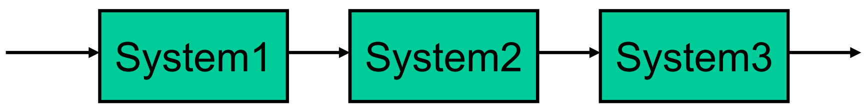




* 8 signals

- Practice 3:
  - Quantize the coefficients and all signals by properly choosing the number of bits and dynamic range (note: has to be 2^n).
  - Conduct fixed-point simulations for the filter and calculate the SQNR.

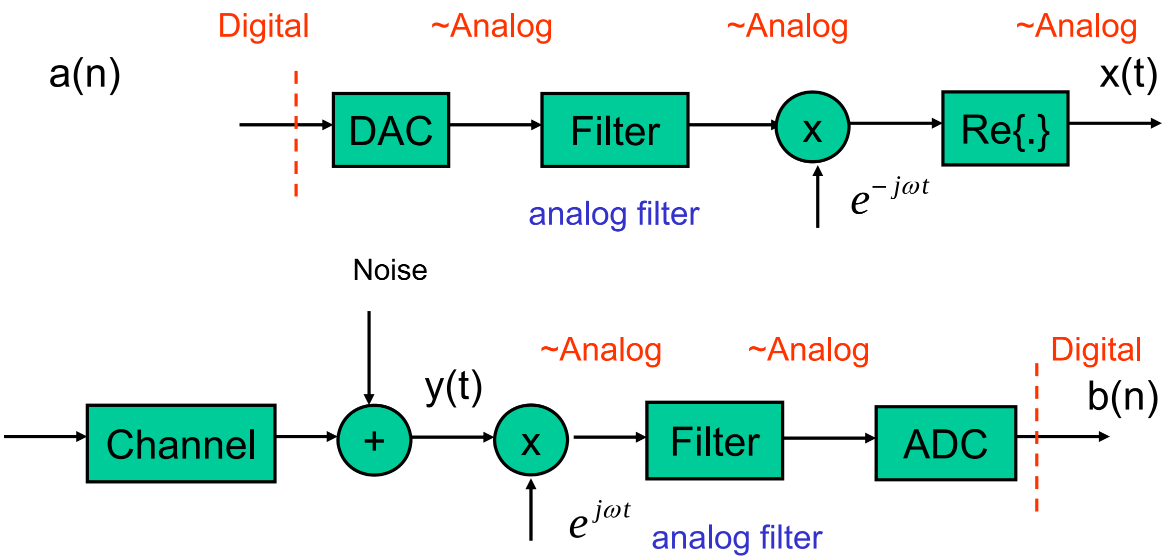
- For a systems with many blocks, how to determine the DR and NOB for each block ?
  - Use floating-point simulations to determine the performance of the system (MSE, SNR, or BER)
  - Quantize the input of the first system and compare the performance with that of the unquantized case. This will determine the NOB for the ADC.
  - Quantize the output of the first system and determine the requirement for SQNR (output NOB).
  - Continue this process until the NOBs of all inputs and outputs are obtained.
  - For each system, we can determine the NOBs of its subsystems to meet the required SQNR.



- Exercise:
  - Model the three structures of the FIR filter and show that they are equivalent.
  - Conduct fixed-point simulations for a communication system that you have built.

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

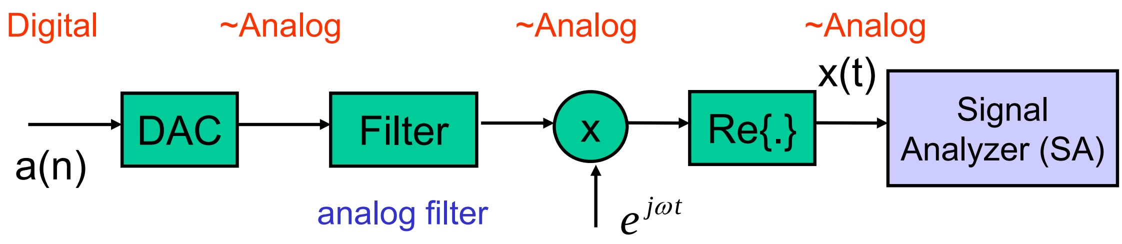
- How to test a communication system?
  - Transmitter/receiver performance (separated)
  - Combine all impairments (combined)
- Transmitter:
  - Transmit signal distortion (for receiver)
  - Spectrum characteristic (for interference control)
- Indices for transmitter
  - Error vector magnitude (EVM)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

* EVM is usually used for QAM signals.

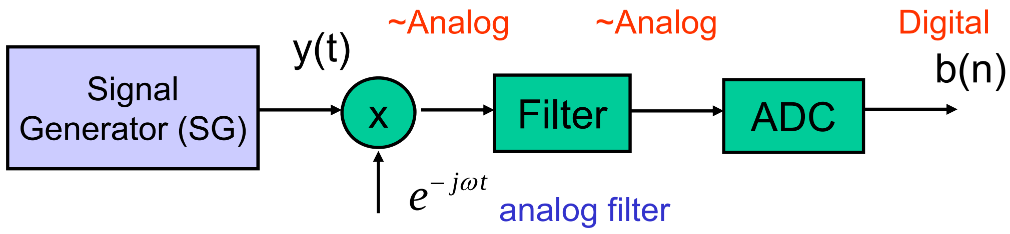
- Index for receiver:
  - Sensitivity
  - Interference performance

- Test equipment:

Transmitter test:



Receiver test:



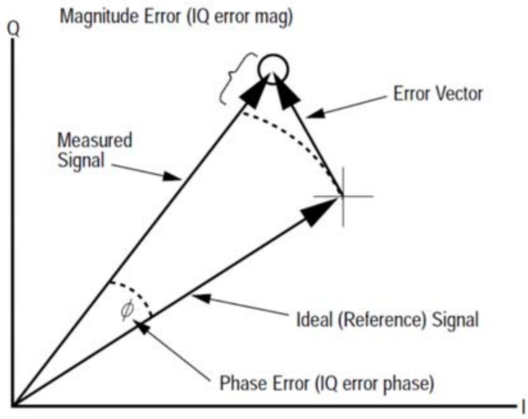
- Case study: Bluetooth 1.0
  - Band : 2.4GHz
  - Bandwidth: 2MHz
  - Data rate : 1Mb/s
  - Modulation: GFSK
- Parameters for Bluetooth
  - Modulation index: $0.28 < h < 0.35$
  - Standard deviation of Gaussian filter: BT=0.5
- Functions to be conducted by transmitter :
  - NRZ signal generation
  -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

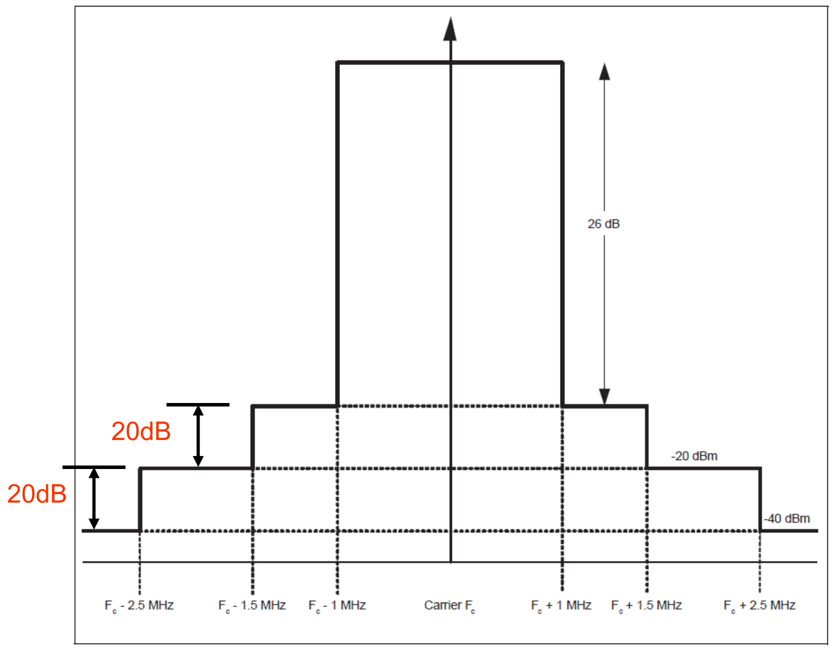
$$\mathbf{EVM(dB)} = 10\log_{10}\frac{E\left\{\left|a(n)-\hat{a}(n)\right|^2\right\}}{E\left\{\left|a(n)\right|^2\right\}}$$

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



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

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



- Practice 1:
  - Let the IF frequency be 2MHz, the oversampling rate for DAC be 16MHz, the number of bits for DAC be 6, and the oversampling rate for analog signal be 64.
  - Design the analog transmit filter for the Bluetooth system fitting the requirement of the spectrum mask.
  - Generate a Bluetooth transmission signal.
  - Write a function to plot the spectrum mask and the transmit signal spectrum. (note: Bluetooth 1.0 does not check EVM).

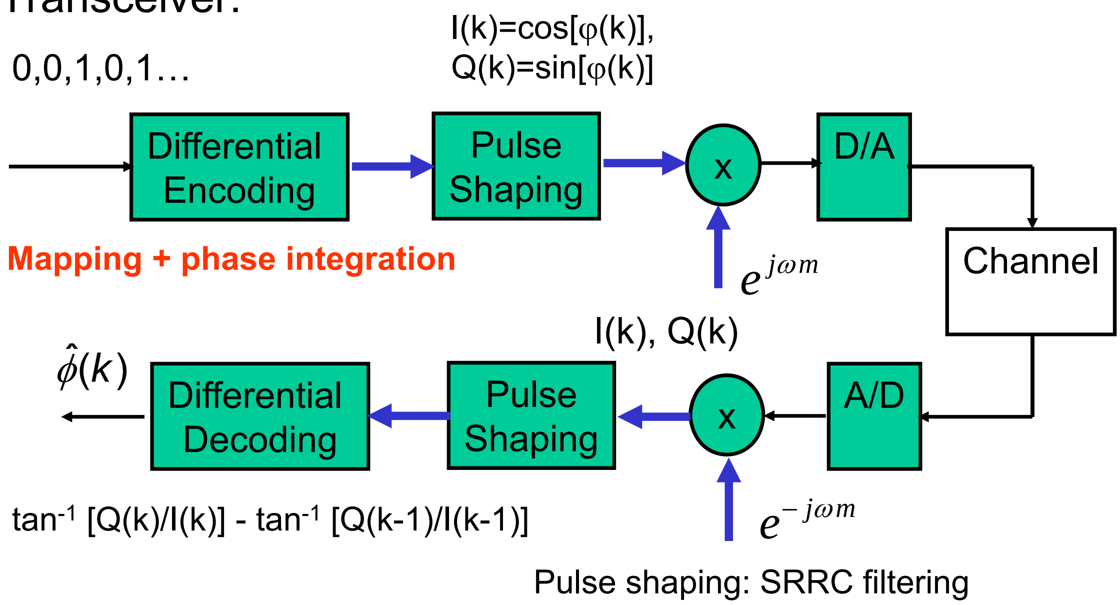
* See spectrum: [px,f]=pwelch(x,[],[],[],1);plot(f,10*log10(px));

- 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 an Bluetooth EDR system for the 2Mpbs mode. Let the parameters of the system are the same as those of Practice 1.
  - Check the spectrum mask.
  - Calculate the EVM.

- 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.

$$\mathrm{dBm} = 10 \times \log 10(Watts \times 1000)$$

- 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 7 dB (after DAC). Calculate the BER.
  - Add AWGN to the system in Practice 2 yielding an SNR of -2dB (after DAC). Calculate the BER.