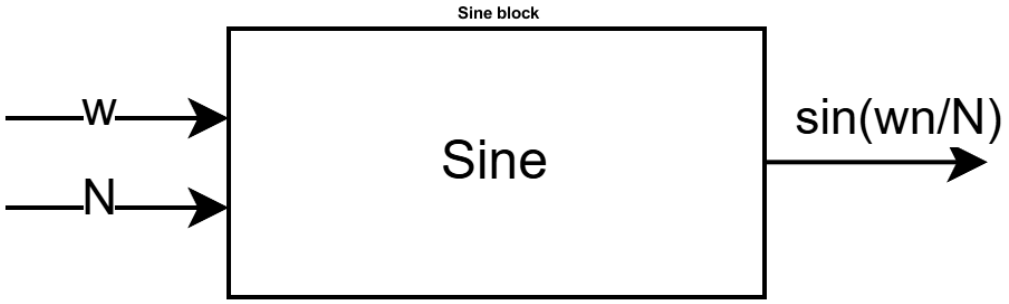
Table of Contents

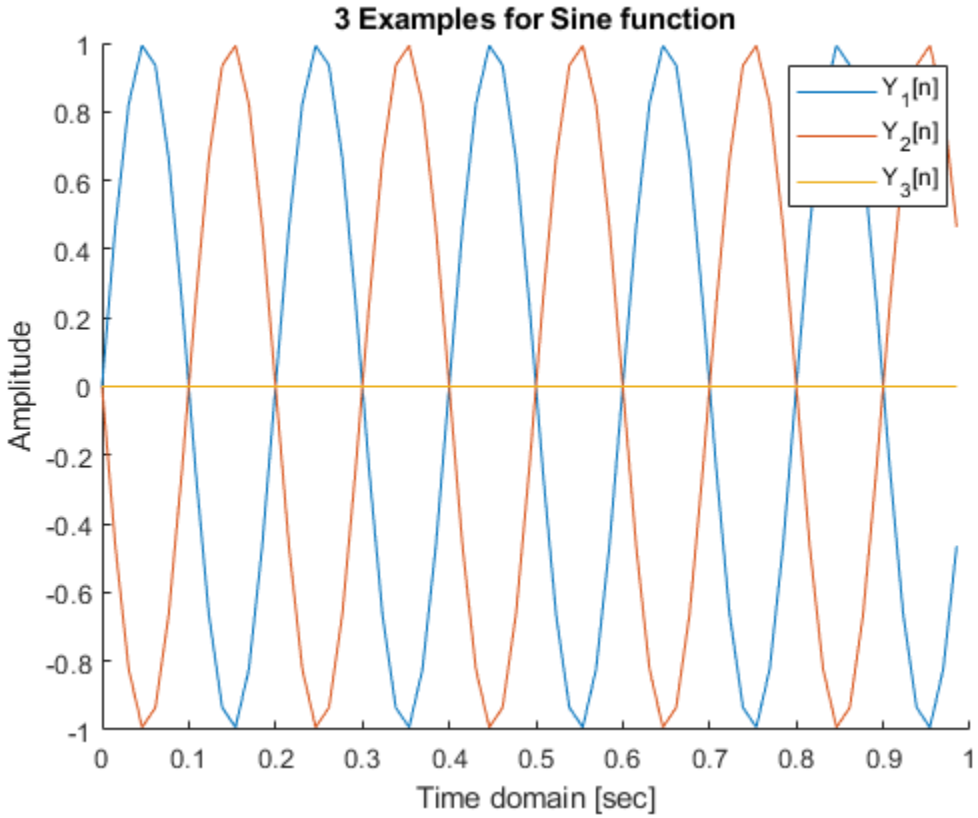
Sine examples

Lets show 3 basic examples using SINE function:

- $y_1[n] = sin(2\pi 5t)$
- $y_2[n] = sin(2\pi 5t + \pi)$
- $y_3[n] = y_1[n] + y_2[n]$

The second signal is delayed by π so we consider to get cosine instead. The third one, is the sum of sine and cosine with the same frequency, so we consider to get ZEROS:



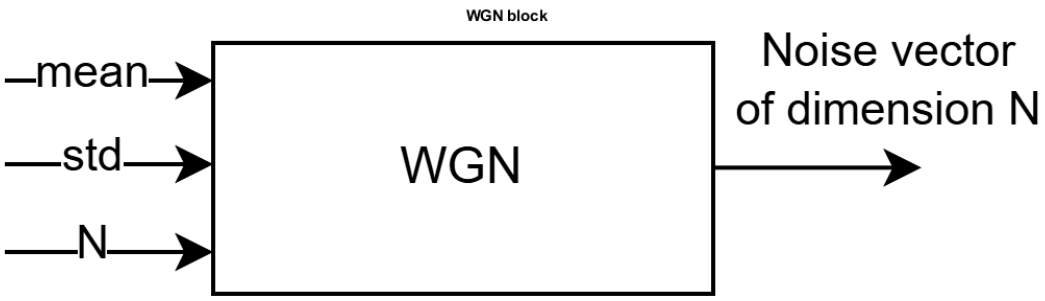


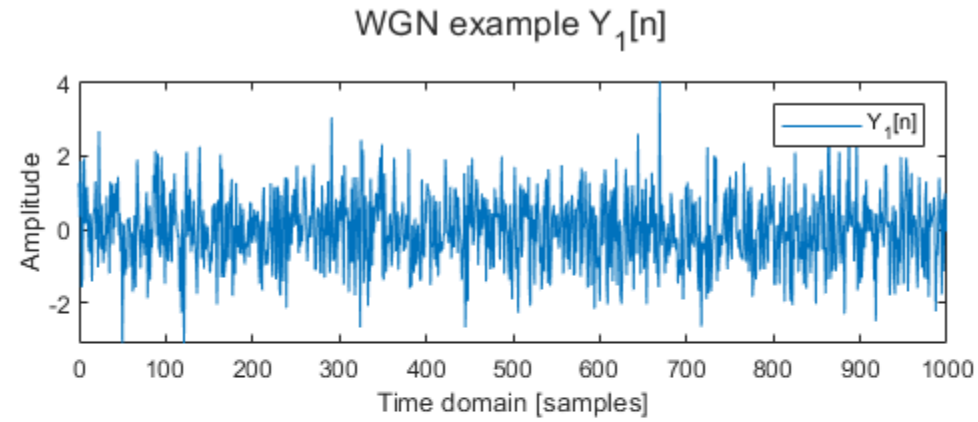
WGN examples

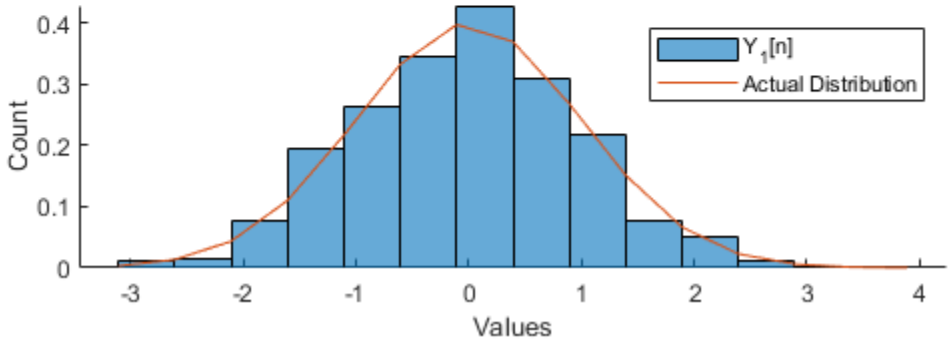
Let's generate 3 different levels of noise:

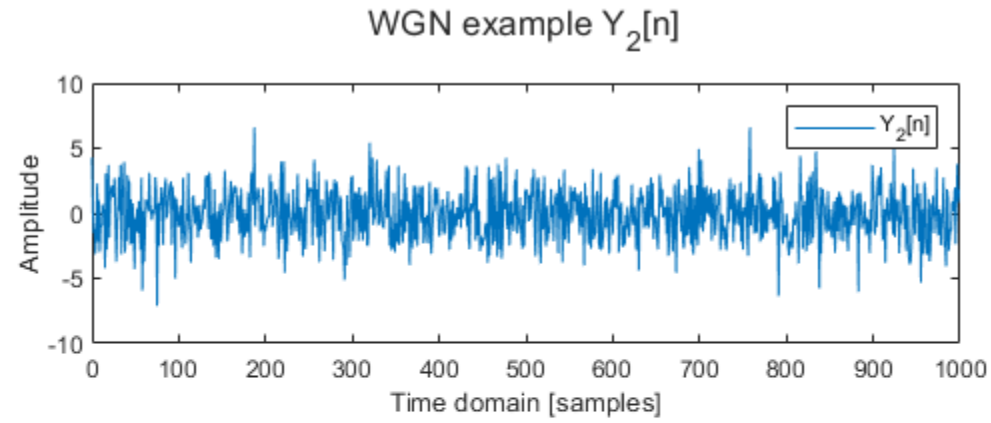
- $y_1[n] = WGN(N, 0, 1)$ - White gaussian noise
- $y_2[n] = WGN(N, 0, 2)$ - White gaussian noise with increased variance
- $y_3[n] = WGN(N, 1, 2)$ - White gaussian noise with increased variance and DC offset

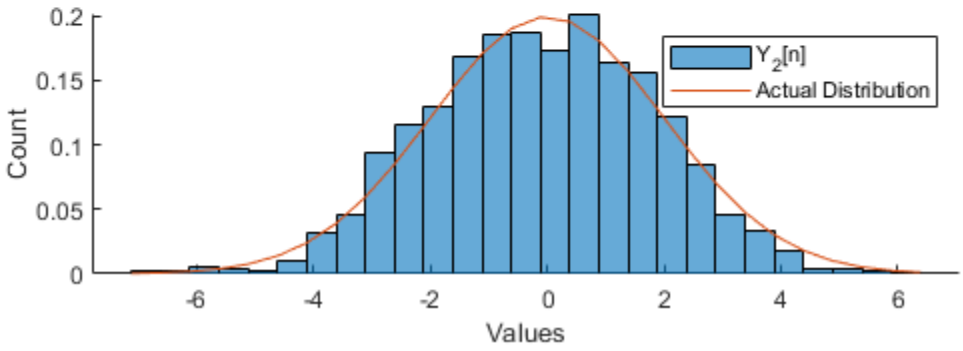
The samples at different times are uncorrelated, so we expect to observe a random signal. Additionally, when examining the empirical probability density function (PDF), we expect to obtain a normal distribution with the parameters above:

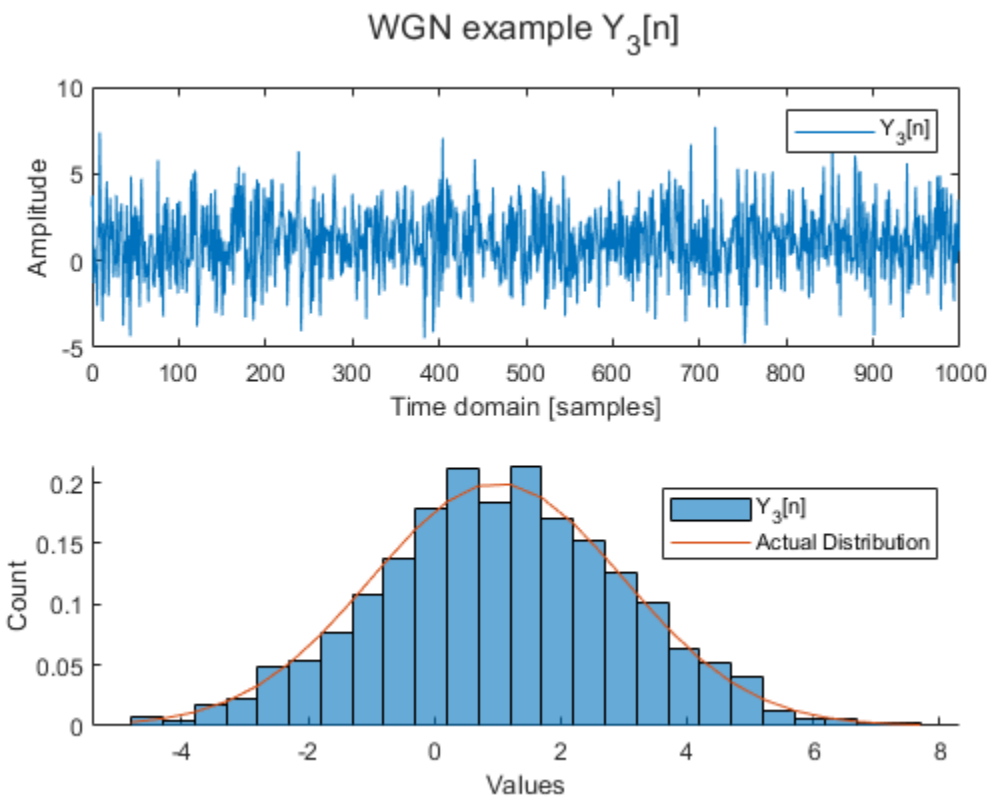



WGN example Y1[n]
Amplitude
Time domain [samples]
Y1[n]


Count
Values
Y1[n]
Actual Distribution


WGN example Y2[n]
Amplitude
Time domain [samples]
Y2[n]


Count
Values
Y2[n]
Actual Distribution

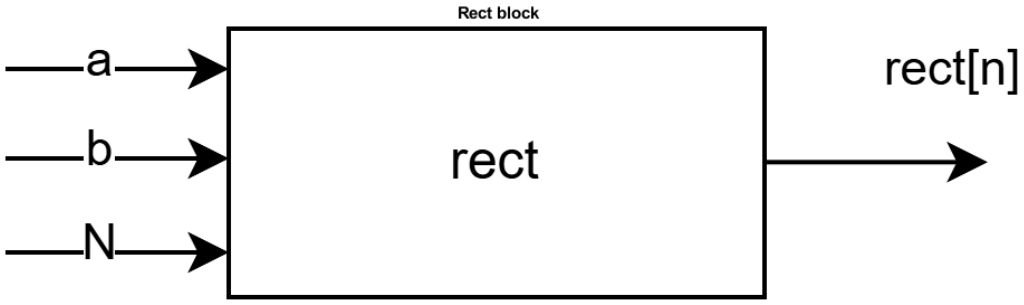


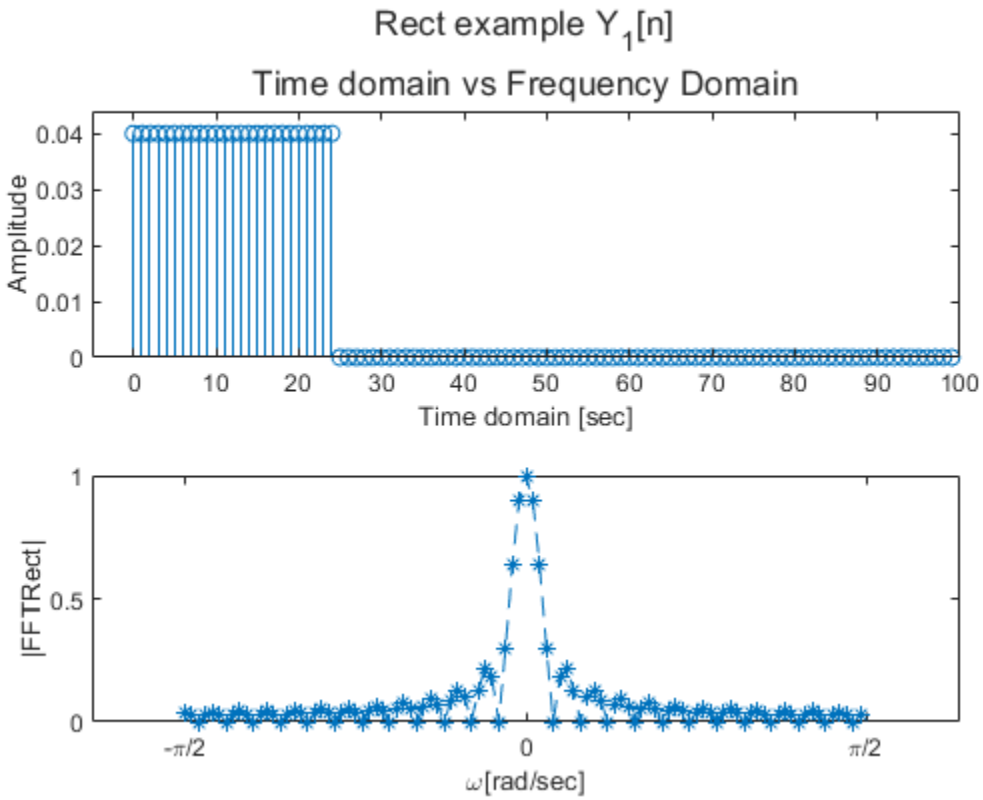
Rect examples

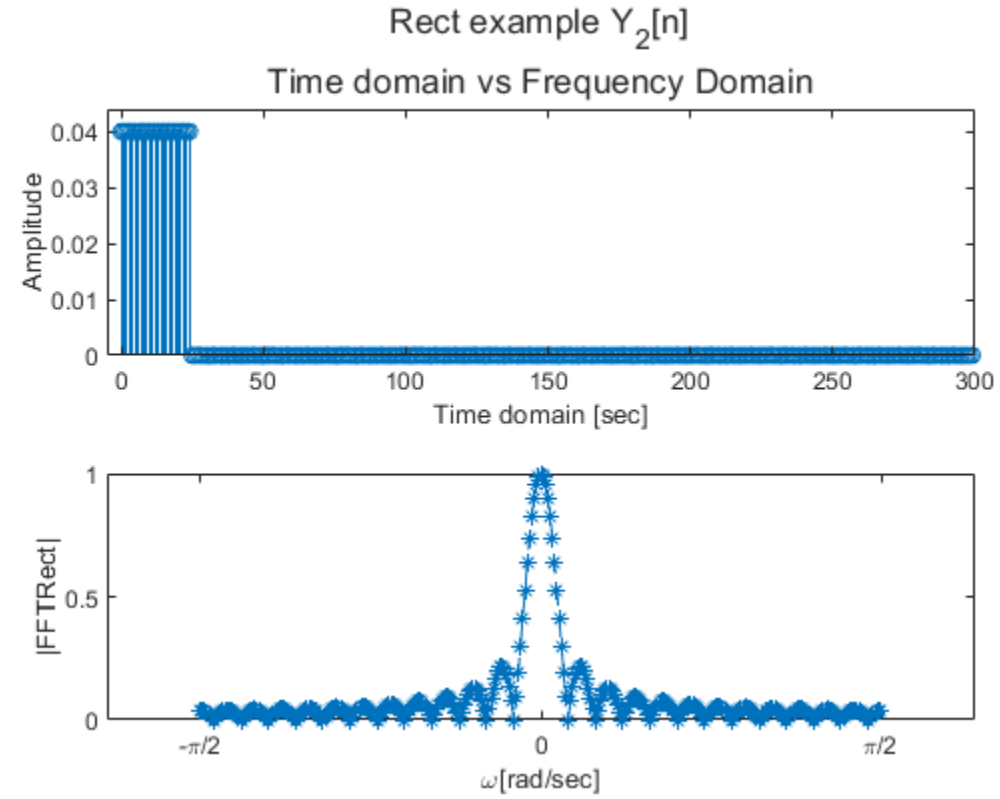
Let's consider 3 examples for Rect function: Let N = 100;

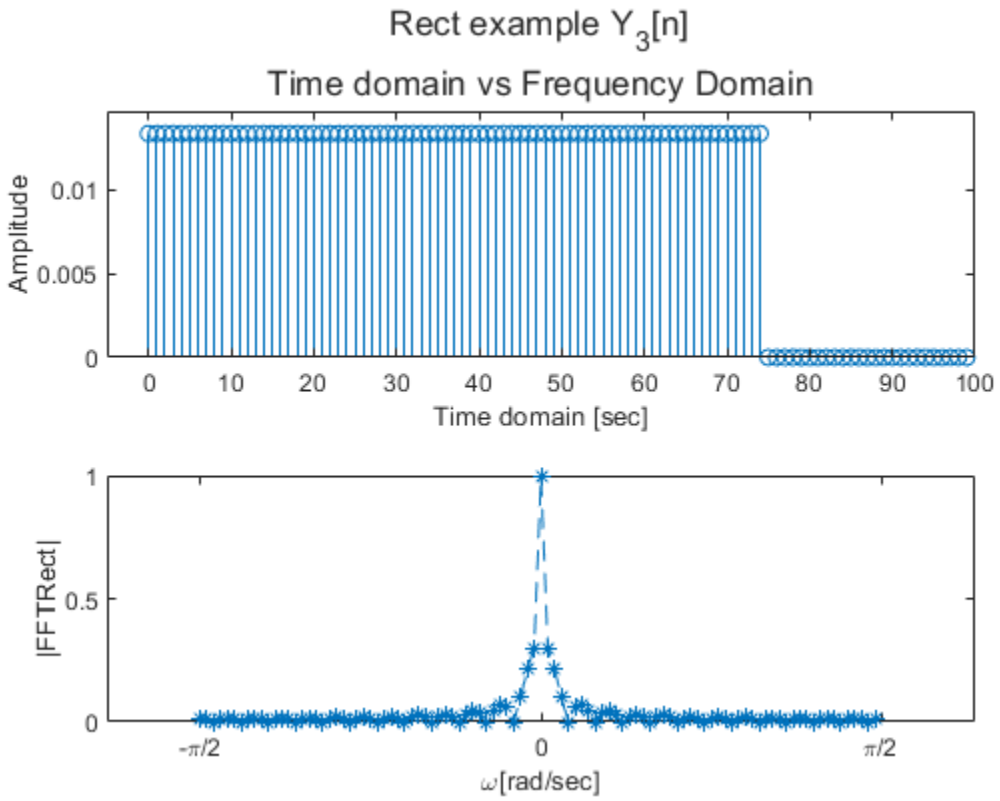
- $y_1[n] = Rect(0, 25, N)$ - window function between 0 and 25 with 100 samples
- $y_2[n] = Rect(0, 25, 3 * N)$ - same as $y_1[n]$ but also padded with additional 200 zeros at the end
- $y_3[n] = Rect(0, 75, N)$ - same as $y_1[n]$ but more time coverage - 0 to 75 with 100 samples

The DFT of $y_1[n]$ is SINC, when the signal is padded with zeros at the end the resolution in frequency domain improved ($y_2[n]$), so we get the same SINC as in case of $y_1[n]$ but with better resolution in frequency domain. And finally, when the window width increase the SINC width in frequency domain decrease approaching to $\delta(\omega)$. Lets show it:


Rect block
a
b
N
rect
rect[n]


Rect example Y1[n]
Time domain vs Frequency Domain
Amplitude
0.04
0.03
0.02
0.01
0
0
10
20
30
40
50
60
70
80
90
100
Time domain [sec]
|FFTRect|
1
0.5
0
-π/2
0
π/2
ω[rad/sec]


Rect example Y2[n]
Time domain vs Frequency Domain
Amplitude
0.04
0.03
0.02
0.01
0
0
50
100
150
200
250
300
Time domain [sec]
|FFTRect|
1
0.5
0
-π/2
0
π/2
ω[rad/sec]


Rect example Y3[n]
Time domain vs Frequency Domain
Amplitude
0.01
0.005
0
0
10
20
30
40
50
60
70
80
90
100
Time domain [sec]
|FFTRect|
1
0.5
0
-π/2
0
π/2
ω[rad/sec]

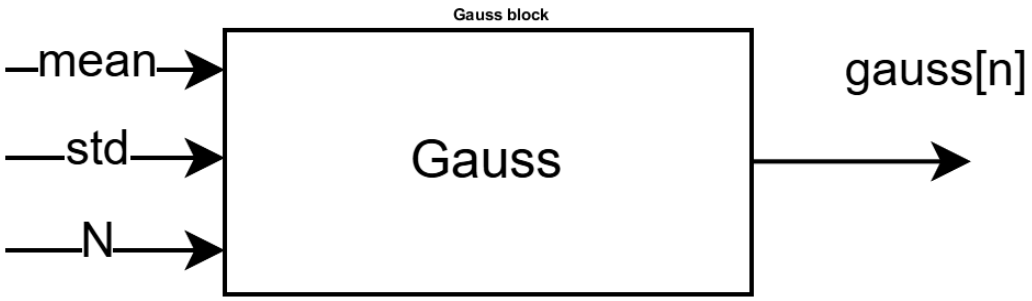
Gauss examples

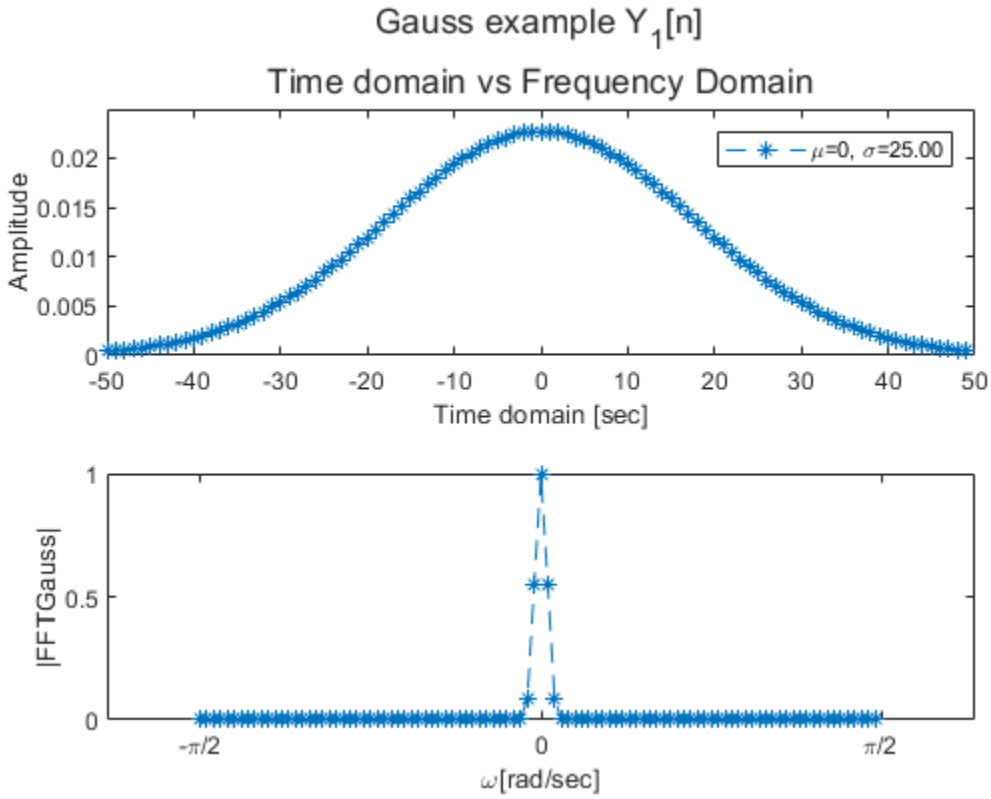
Let's consider 3 examples for Gauss function: Let N = 100;

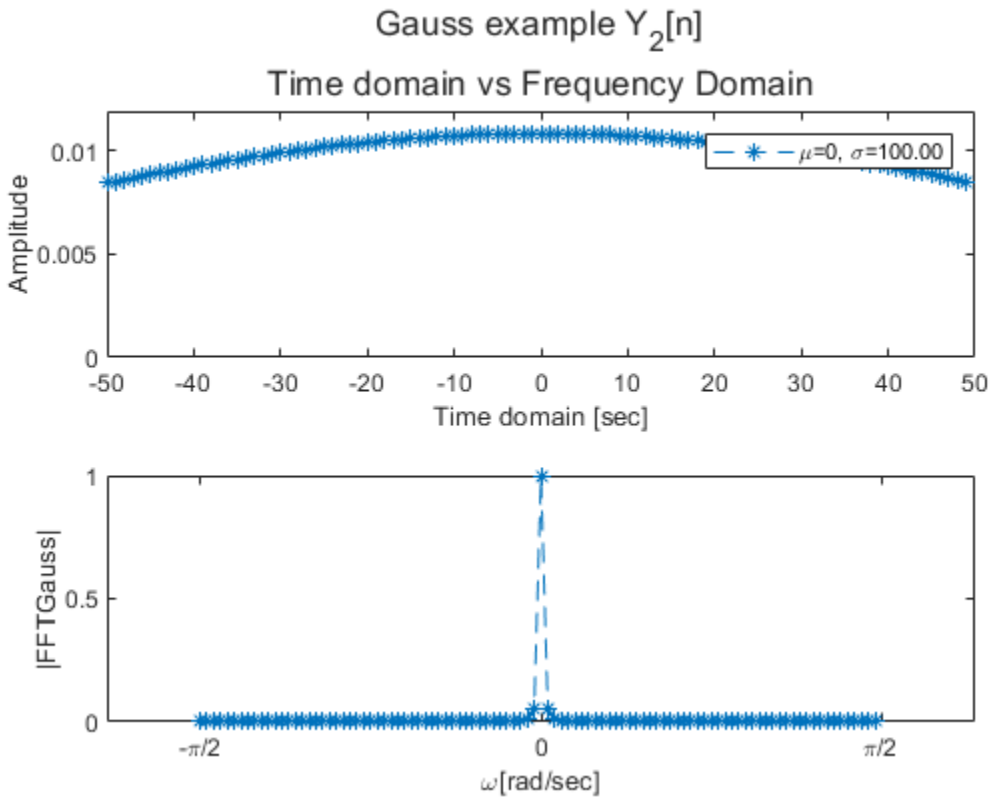
- $y_1[n] = Gauss(0, 25, N)$ - $N(0, 25^2)$, 100 samples
- $y_2[n] = Gauss(0, 100, 2 * N)$ - $N(0, 100^2)$, 100 samples
- $y_3[n] = Gauss(0, 0.1, N)$ - $N(0, 0.1^2)$, 100 samples

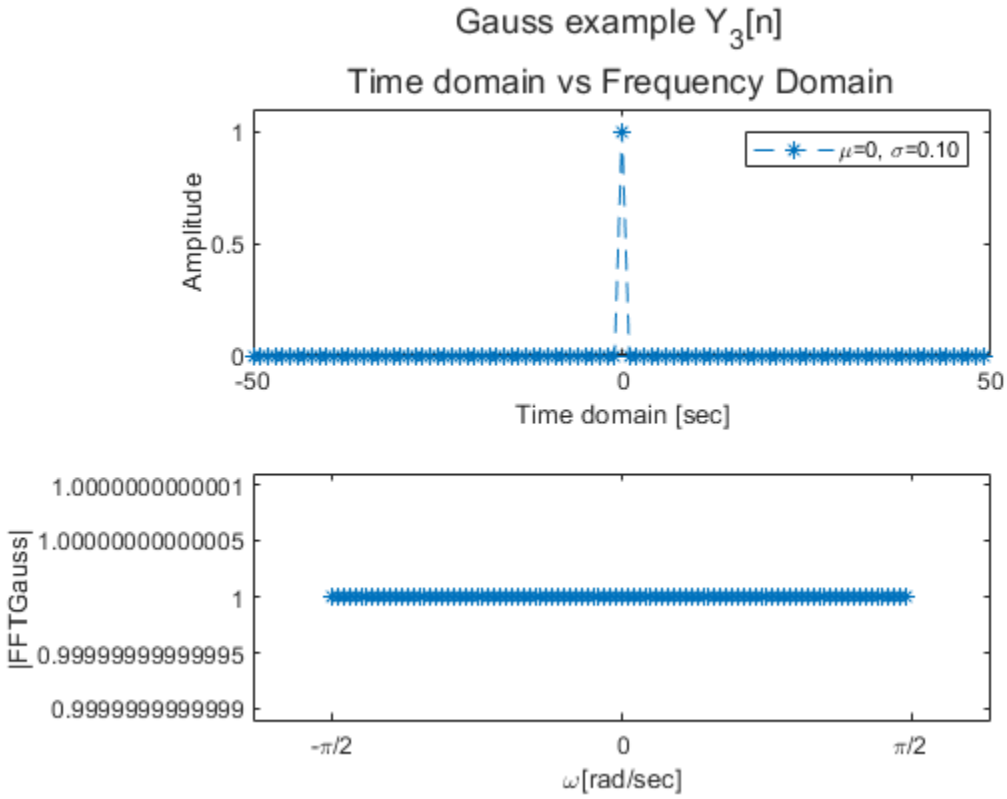
The gaussian can be used as a window function. It's smoother then rect window. One can see, that when:

- $\sigma^2 \to \infty$ - it tends to be like infinite rect window\DC so the spectrum is $\delta(\omega)$
- $\sigma^2 \to 0$, it tends to be $\delta(t)$ so the spectrum is flat.
- $\sigma^2 < \infty$, it can interpretated as smooth window with nice properties.




Gauss example Y1[n]
Time domain vs Frequency Domain
Amplitude
0.02
0.015
0.01
0.005
0
-50
-40
-30
-20
-10
0
10
20
30
40
50
Time domain [sec]
μ=0, σ=25.00
|FFTGauss|
1
0.5
0
-π/2
0
π/2
ω[rad/sec]


Gauss example Y2[n]
Time domain vs Frequency Domain
Amplitude
0.01
0.005
0
-50
-40
-30
-20
-10
0
10
20
30
40
50
Time domain [sec]
μ=0, σ=100.00
|FFTGauss|
1
0.5
0
-π/2
0
π/2
ω[rad/sec]



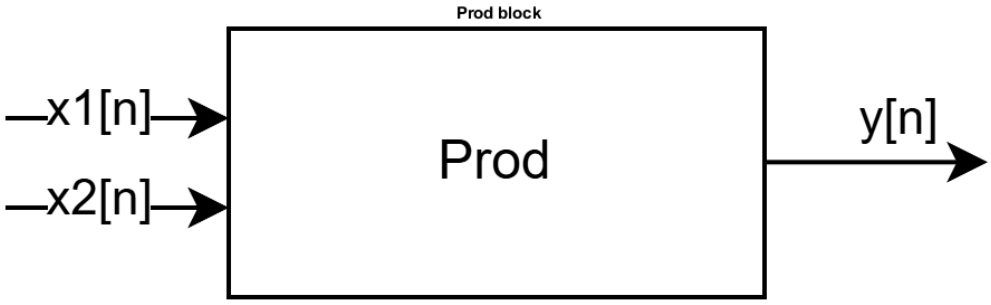
Prod examples

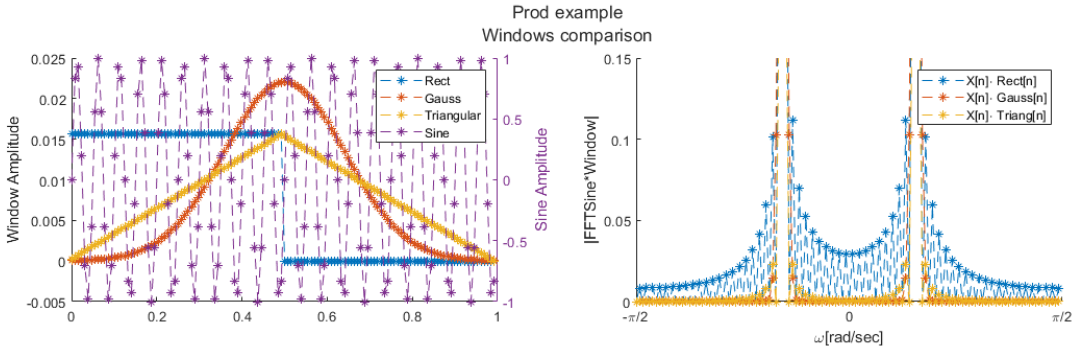
Let's consider 3 examples for Prod function: Let N = 64; $X[n] = sin(2\pi \cdot 5n/N)$ - 1 [sec] of the sine.

- $y_1[n] = X[n] * Rect(0, 0.5, N)$ - X[n] multiplied by rect window (between 0 and 0.5[sec])
- $y_2[n] = X[n] * Gauss(0, 0.2, N)$ - X[n] multiplied by Gauss window ($\mu = 0, \sigma = 0.2$)
- $y_3[n] = X[n] * Triangle$ - X[n] multiplied by triangle window

Windows can be used to reduce artifacts that occured due to fact that the signal is finite. We will consider the 3 windows: Rect, Gauss and Triangle. As we have seen (044198):

- Rect response is Dirichlet func in frequency domain.
- Gauss is smoother than rect, so we expect to get lower sidelobs.
- Triangle is Dirichlet squared, so lobes should be lower comparing to rect.





Add examples

Let's consider 3 examples for Add function: Let N = 64; $X[n] = sin(2\pi \cdot 5n/N)$ - 1 [sec] of the sine. Noises we used few sections above:

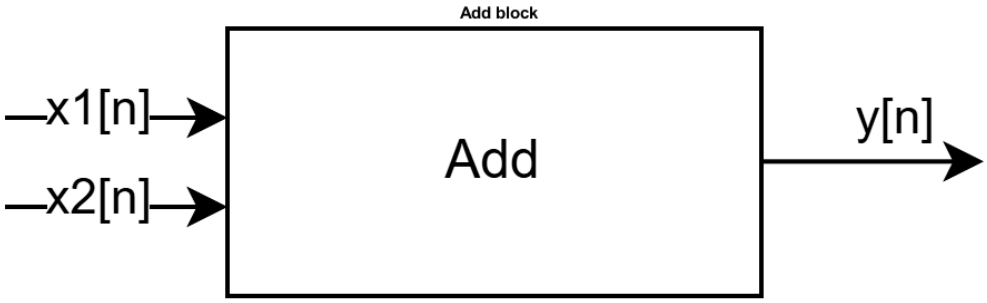
- $n_1 = WGN(N, 0, 0.1)$
- $n_2 = WGN(N, 1, 0.1)$
- $n_2 = WGN(N, 0, 1)$

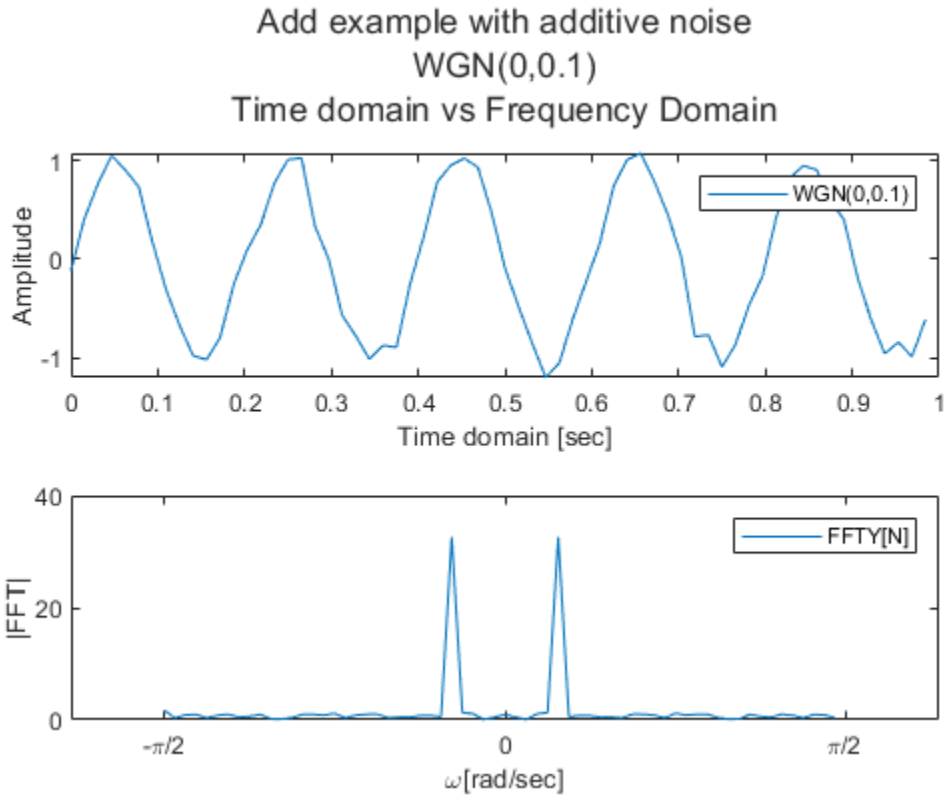
Let's consider:

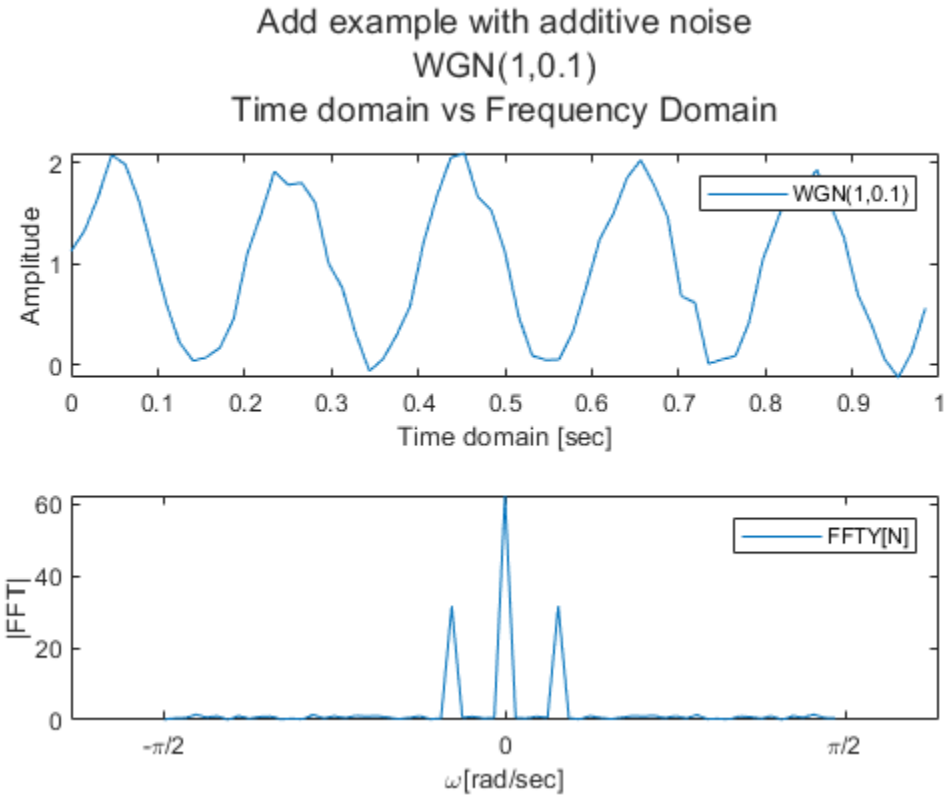
- $y_1[n] = X[n] + n_1[n]$ - X[n] with low power noise
- $y_2[n] = X[n] + n_2[n]$ - X[n] with low power noise with "DC" ($\mu \neq 0$)
- $y_3[n] = X[n] + n_3[n]$ - X[n] with high power noise

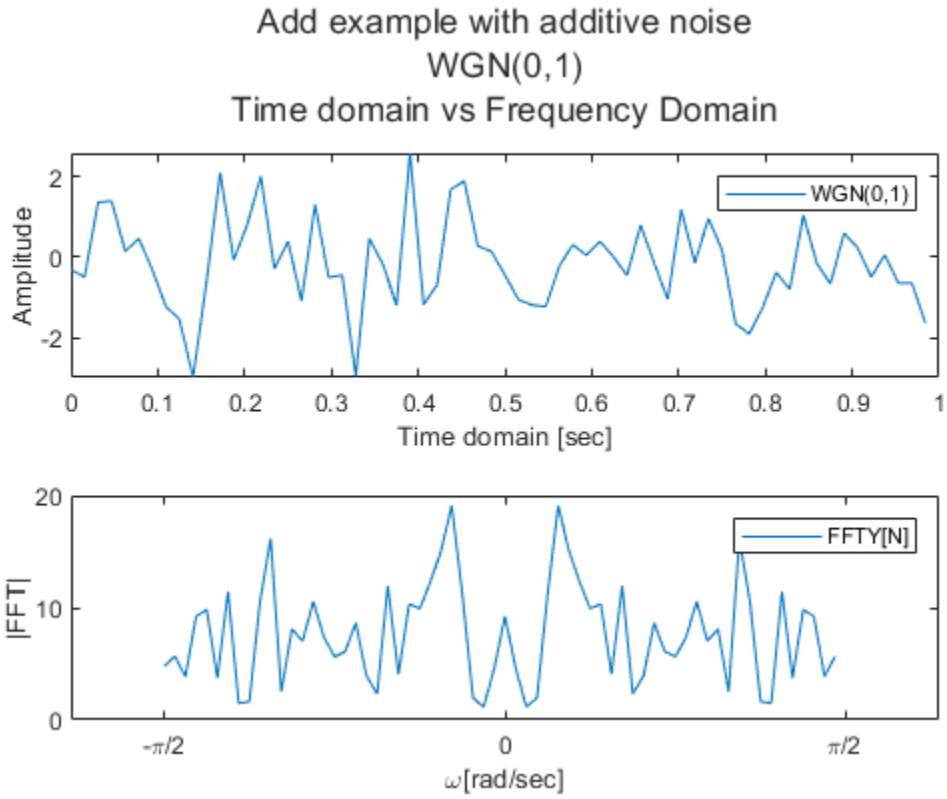
We expect that:

- $y_1[n] \approx X[n]$ because the SNR is good enough
- $y_2[n] \approx y_1[n]$ but there will be DC frequency
- Finally, $y_3[n]$ will look differently than $X[n]$ because the SNR is bad, but in frequency domain we are able to see the deltas. Intuitively, that's because DFT does integration over time. Let's compute add operations:


Add block
x1[n]
x2[n]
Add
y[n]


Add example with additive noise
WGN(0,0.1)
Time domain vs Frequency Domain
Amplitude
WGN(0,0.1)
Time domain [sec]
|FFT|
FFTY[N]
ω[rad/sec]


Add example with additive noise
WGN(1,0.1)
Time domain vs Frequency Domain
Amplitude
Time domain [sec]
WGN(1,0.1)
|FFT|
ω[rad/sec]
FFTY[N]
-π/2
0
π/2


Add example with additive noise
WGN(0,1)
Time domain vs Frequency Domain
Amplitude
Time domain [sec]
WGN(0,1)
|FFT|
ω[rad/sec]
FFTY[N]
-π/2
0
π/2

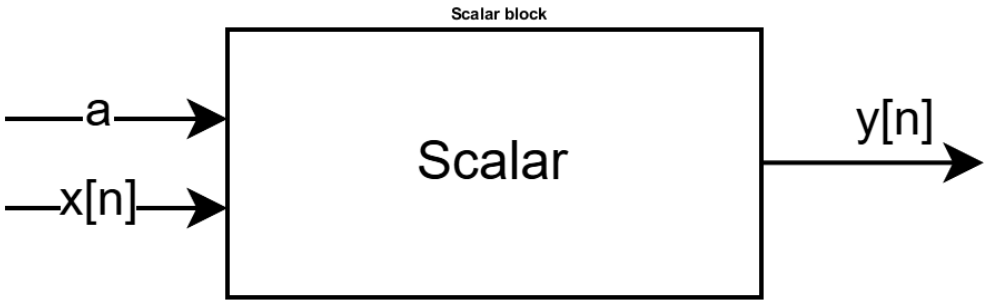
Scalar examples - Rotated 16QAM

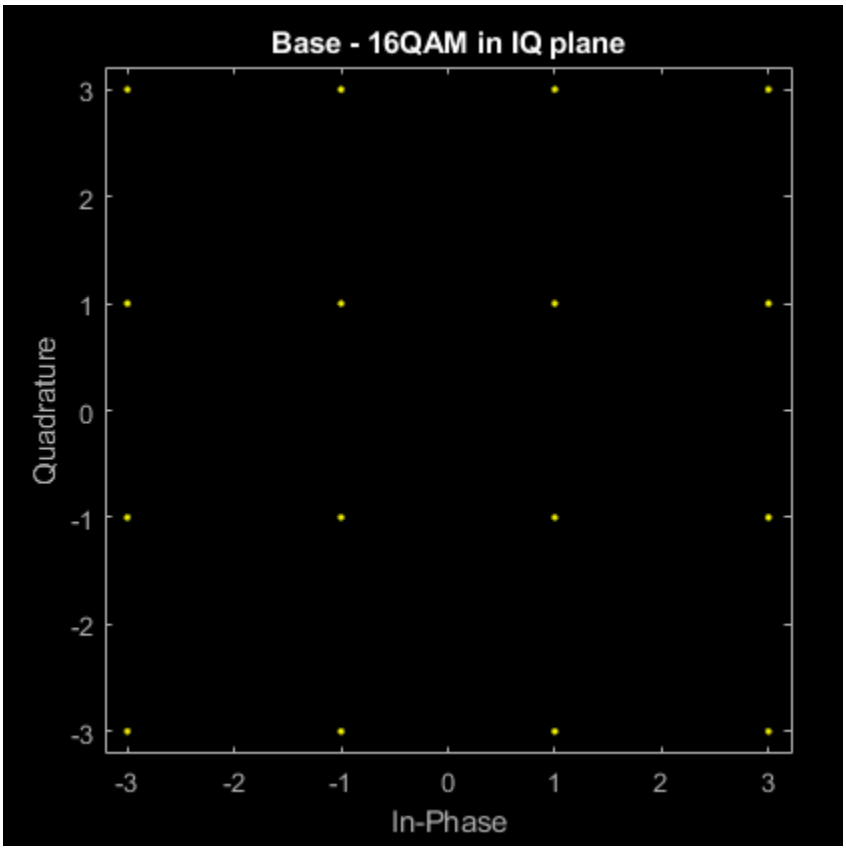
Let's consider 3 examples for Scalar function.

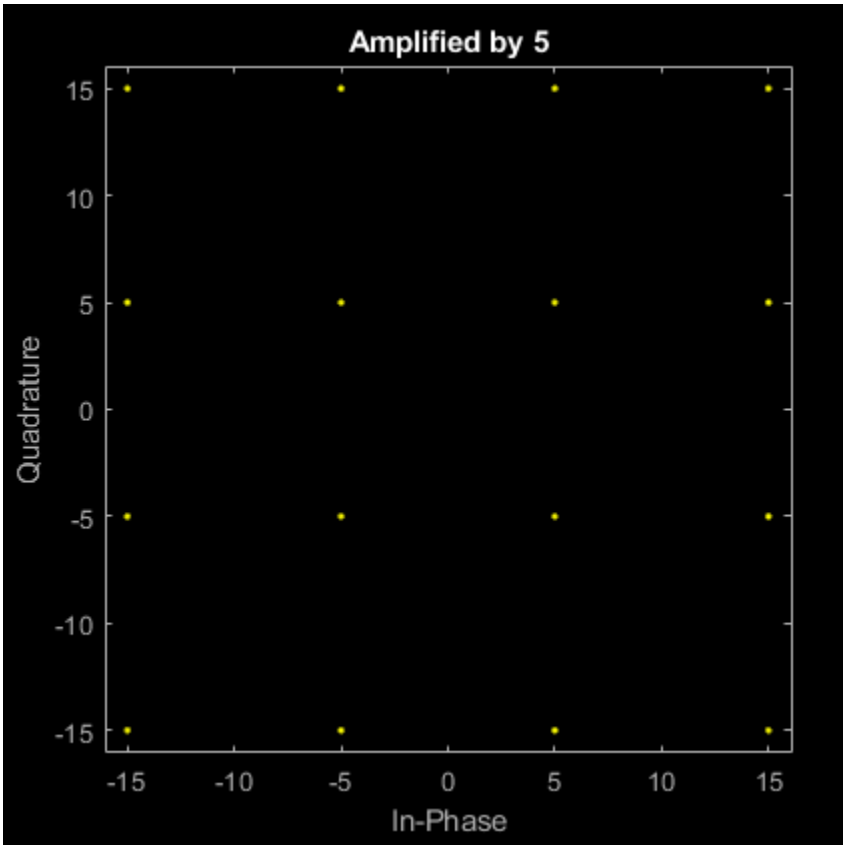
We will define set of complex numbers with 2 energy levels and phases $n\pi/4$ (16QAM symbols).

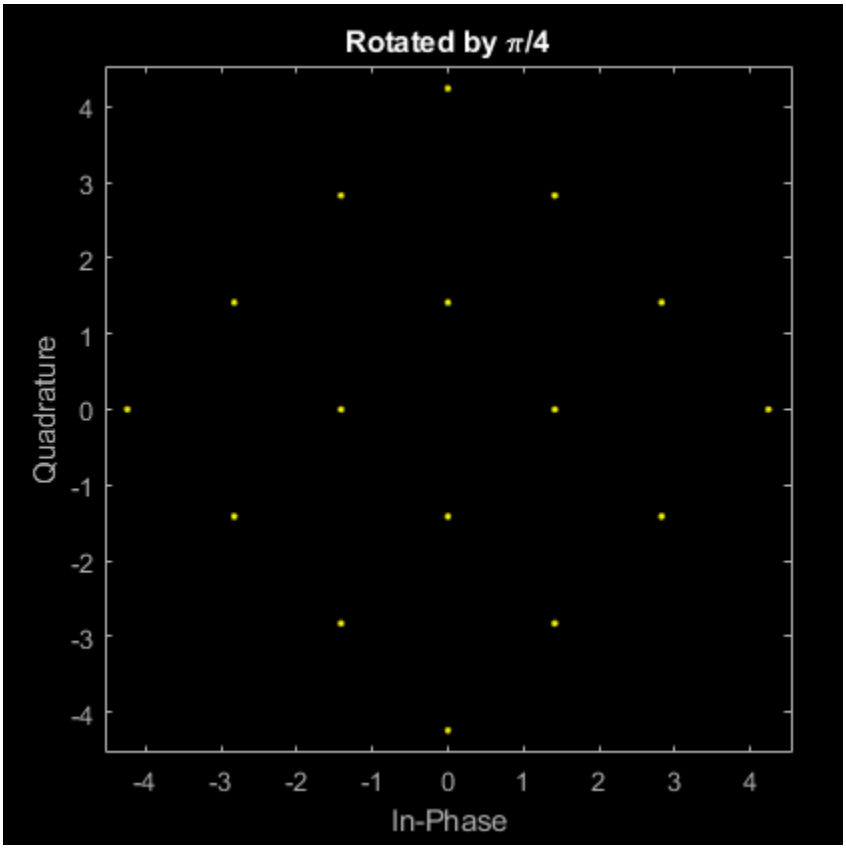
By multiplying these numbers by a scalar, we can observe interesting phenomenas. We will multiply by:

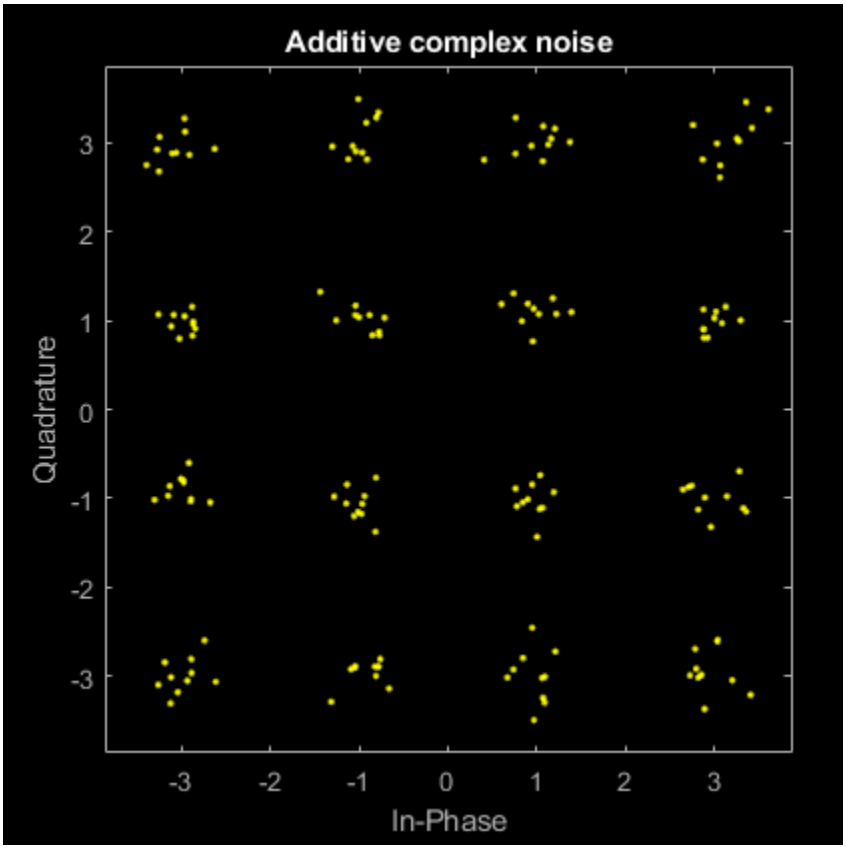
- a (Real scalar) - we expect that the energy of each symbol will increase by a.
- $exp(j\pi/4)$ - all of the symbols will be rotated by $\pi/4$
- Add additive complex noise - the symbols will be normally distributed around the expected QAMs locations. Let's apply Scalar multiplications:






Amplified by 5
15
10
5
0
-5
-10
-15
Quadrature
-15
-10
-5
0
5
10
15
In-Phase


Rotated by π/4
4
3
2
1
0
-1
-2
-3
-4
Quadrature
-4
-3
-2
-1
0
1
2
3
4
In-Phase



Filter examples

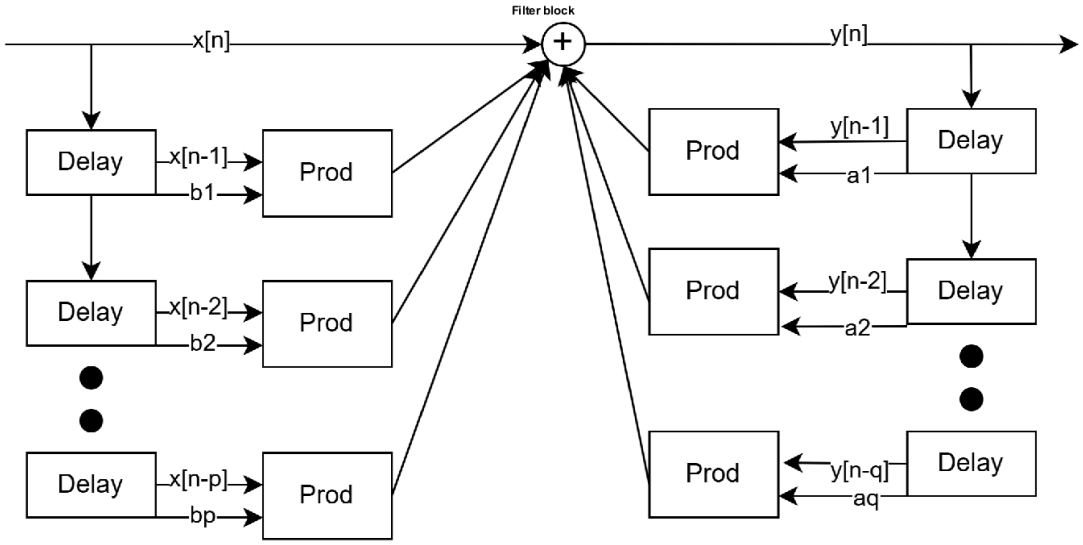
Let's consider 3 examples for Filter function:

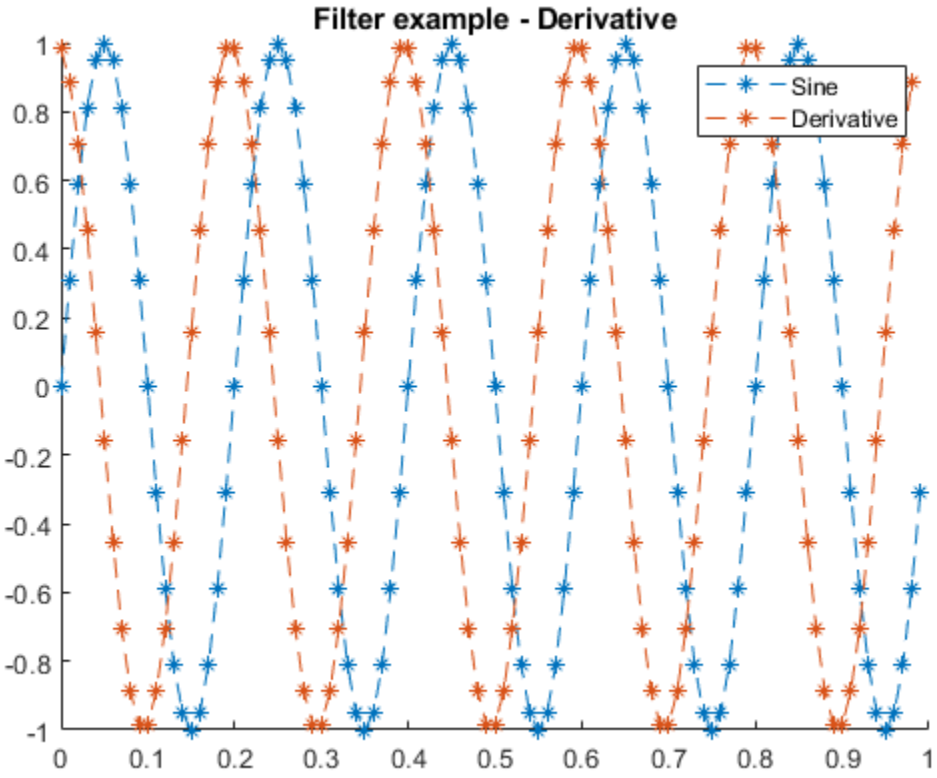
- $a_n = 1, b_n = [1, -1] * c$ - Derivative filter
- $a_n = 1, b_n = ones(1, M)$ - Matched filter
- $a_n = 1, b_n = \bar{\alpha}$ - Lowpass filter

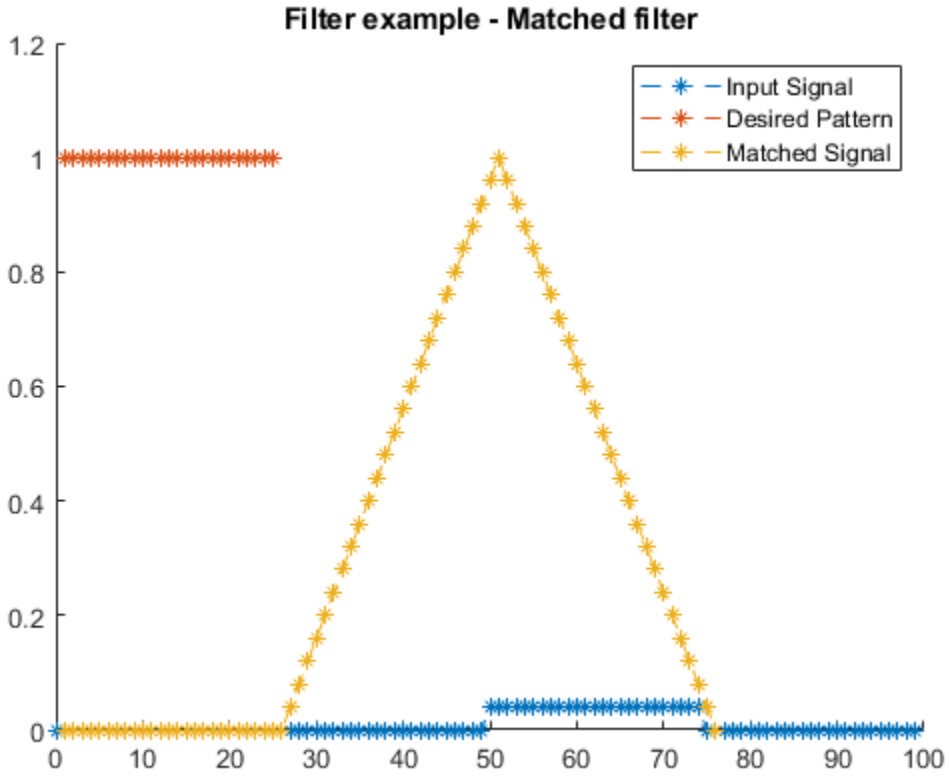
With derivative filter we will use $X(t) = sin(2\pi \cdot 5t)$ as an input signal. We expect to get cosine.

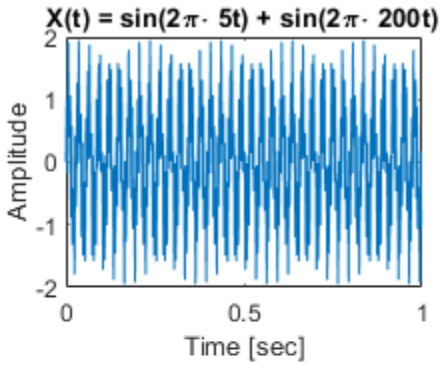
With Matched filter we will use delayed Rect as an input signal and Rect as a pattern. We expect to get triangle as a matched signal results (convolution of rects), when the maximum will be placed at the begining of the delayed rect on input signal.

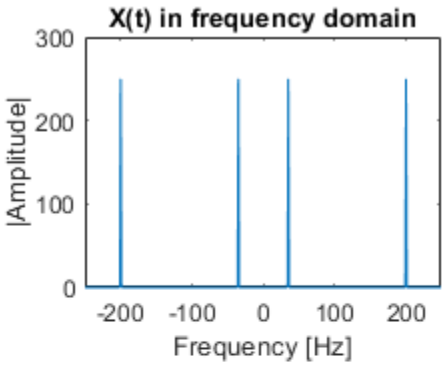
With Lowpass filter we will use $X(t) = sin(2\pi \cdot 35t) + sin(2\pi \cdot 200t)$ as an input signal. We will design FIR lowpass filter, with $F_{cutoff} = 50[Hz]$. We expect to get only the $sin(2\pi \cdot 35t)$ as result. Let's apply filters:

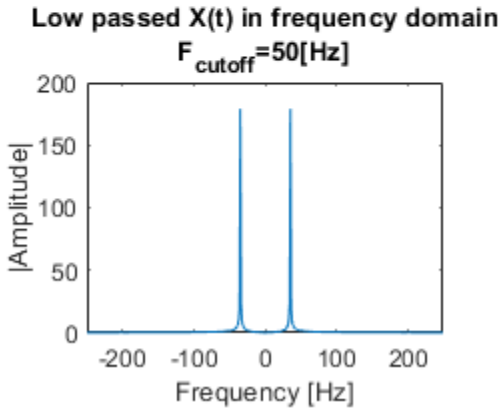

Filter block
x[n]
y[n]
Delay
x[n-1]
b1
Prod
Delay
x[n-2]
b2
Prod
Delay
x[n-p]
bp
Prod
Prod
y[n-1]
a1
Delay
Prod
y[n-2]
a2
Delay
Prod
y[n-q]
aq
Delay


Filter example - Derivative
Sine
Derivative
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Filter example - Matched filter
Input Signal
Desired Pattern
Matched Signal
1.2
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
50
60
70
80
90
100


X(t) = sin(2π· 5t) + sin(2π· 200t)
Amplitude
Time [sec]
2
1
0
-1
-2
0
0.5
1


X(t) in frequency domain
|Amplitude|
Frequency [Hz]
300
200
100
0
-200
-100
0
100
200


Low passed X(t) in frequency domain
F cutoff=50[Hz]
|Amplitude|
Frequency [Hz]
200
150
100
50
0
-200
-100
0
100
200

Interpolate examples

Let's consider 3 examples for Interpolate function:

- $x_1[n] = sin(2\pi \cdot 5n/64)$ - sine wave
- $x_2[n] = Rect(0, 0.5, 64)$ - rect window
- $x_3[n] = Gauss(0, 0.75, 64)$ - gauss window

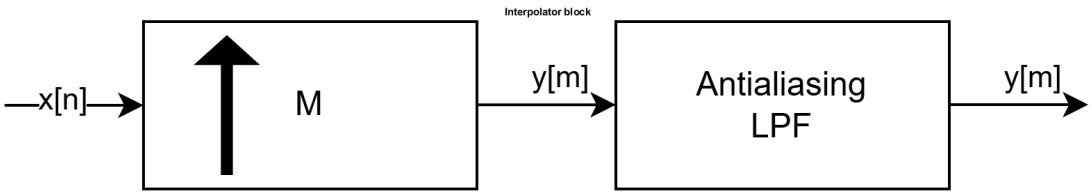
The signals have the following BW:

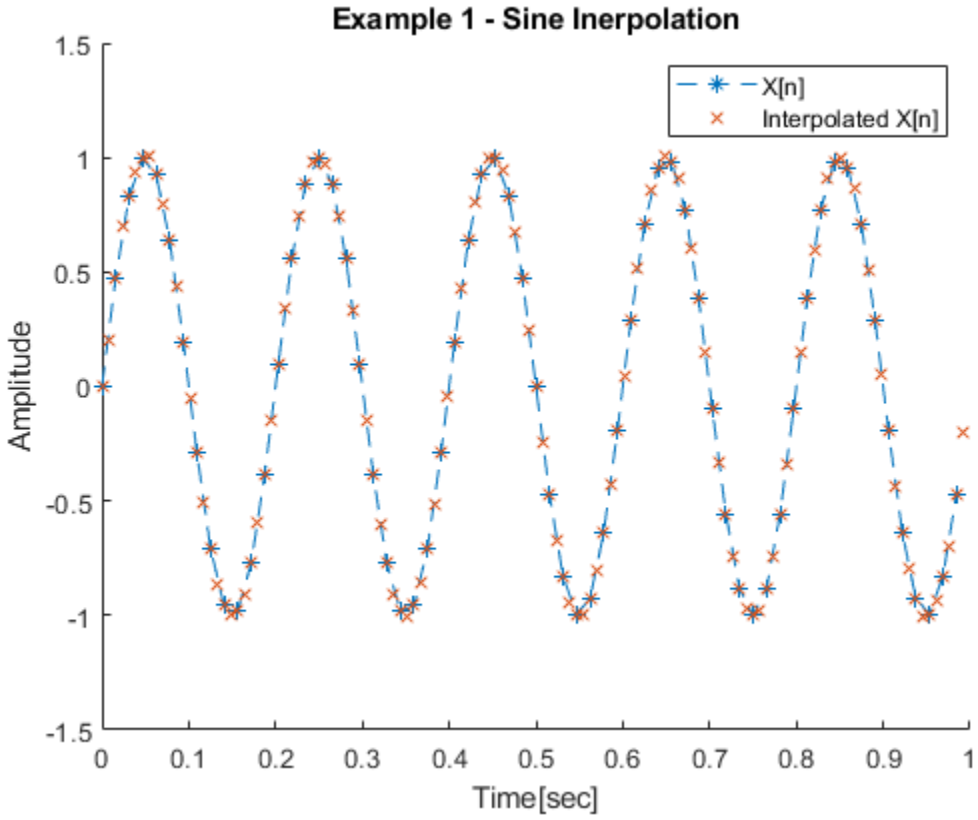
- $x_1[n]$ - $\delta(f + -5)$, so it's finite BW
- $x_2[n]$ - infinite sum of deltas in frequency domain, so it's infinite BW
- $x_3[n]$ - depending on σ^2, it tends to include more\less high frequencies. if $\sigma^2 \to \infty$, the window has fast changes otherwise it changes "slowly", so we expect to get wide and narrow BW respectively

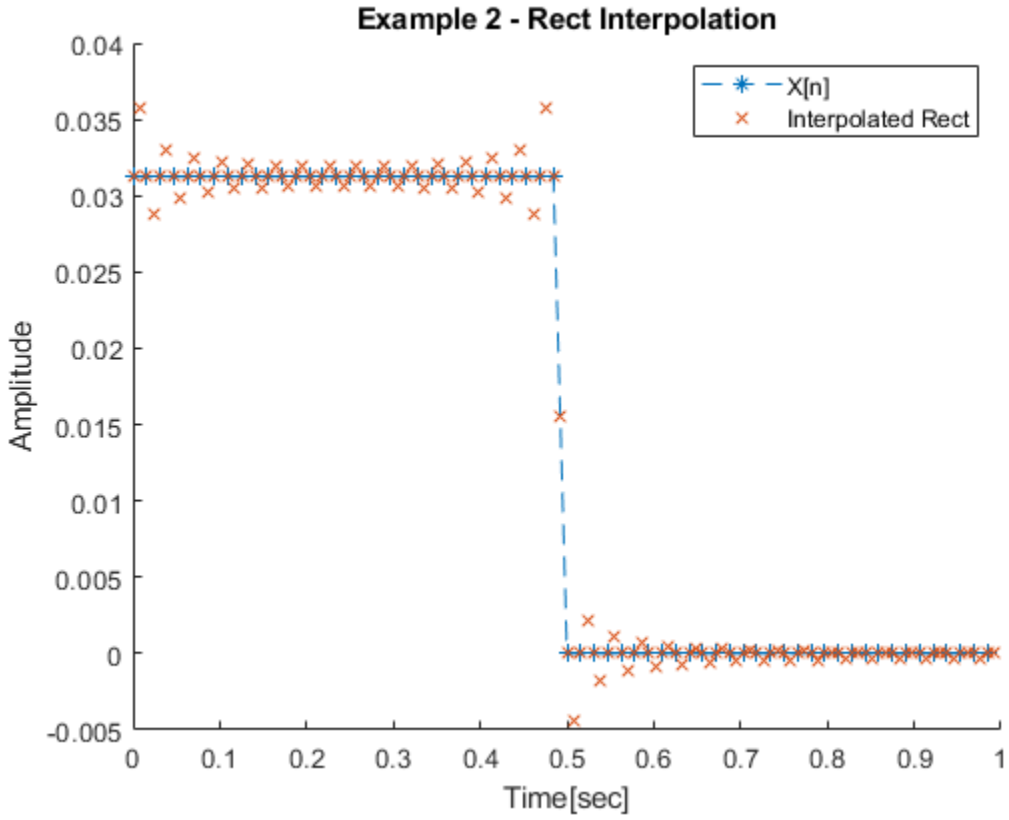
The fast changes implies kind of 'discontinuity' and we know (044198) that the reconstructed signal will converge to $(f(a^+) + f(a^-))/2$ when a represents a point of discontinuity. Furthermore, spectral artifacts such as the Gibbs phenomenom will be introduced into the signal.

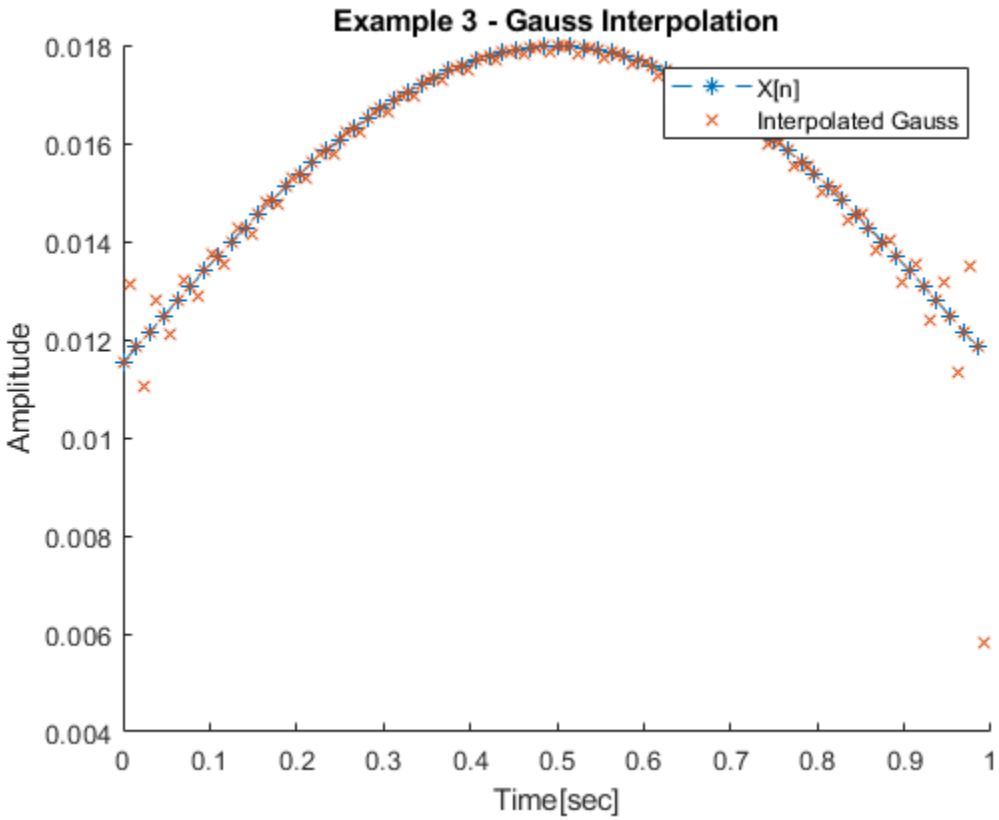
So we expect that:

- $x_1[n]$ - no distortion at all beacuse there is no discontinuity (if we take full period)
- $x_2[n]$ - Gibbs distortion will occur because the signal has discontinuity
- $x_3[n]$ - will be slightly distorted beacuse there is 'smaller' discontinuity (only at the edges)




Example 1 - Sine Inerpolation
X[n]
Interpolated X[n]
Amplitude
Time[sec]
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Example 2 - Rect Interpolation
X[n]
Interpolated Rect
Amplitude
Time[sec]
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
-0.005
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



Decimate examples

Let's consider 3 examples for Decimate function: (Same as for intepolation)

- $x_1[n] = sin(2\pi \cdot 5n/64)$ - sine wave
- $x_2[n] = Rect(0, 0.5, 64)$ - rect window
- $x_3[n] = Gauss(0, 0.75, 64)$ - gauss window

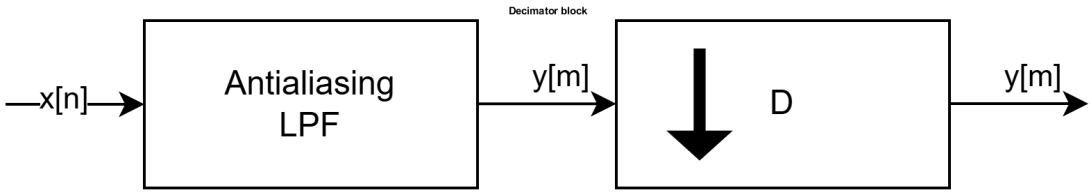
Each of the signals above has differnet BW:

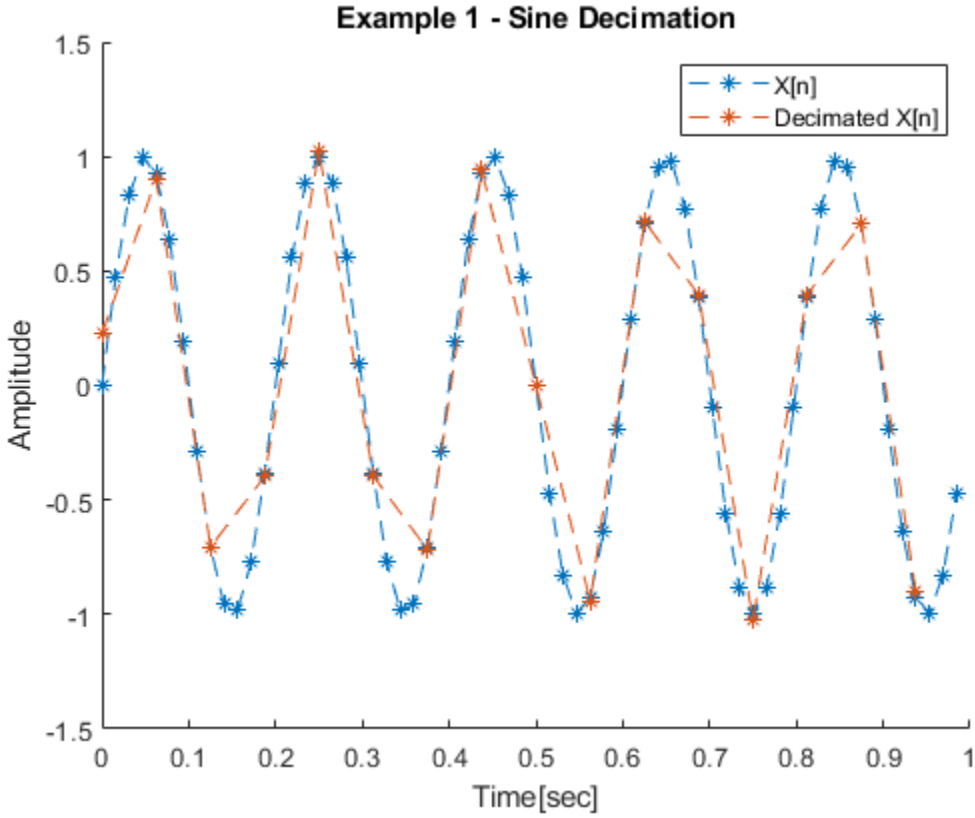
- $x_1[n]$ - $\delta(f + -5)$, so it's finite BW
- $x_2[n]$ - infinite sum of deltas in frequency domain, so it's infinite BW
- $x_3[n]$ - depending on σ^2, it tends to include more\less high frequencies. if $\sigma^2 \to \infty$, the window has fast changes otherwise it changes "slowly", so we expect to get wide and narrow BW respectively.

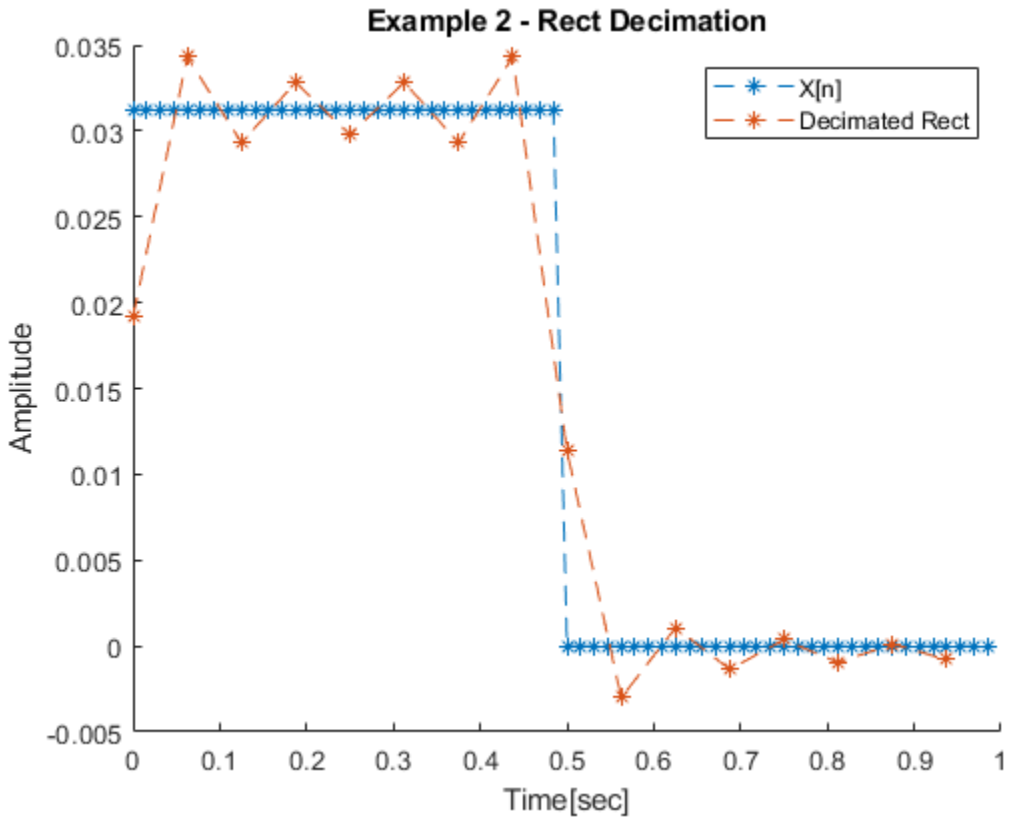
As a part of decimation, we filter out some of the high frequencies in order to avoid aliasing, so the reconstructed signal is composed of smaller BW. So we expect that decimated signals:

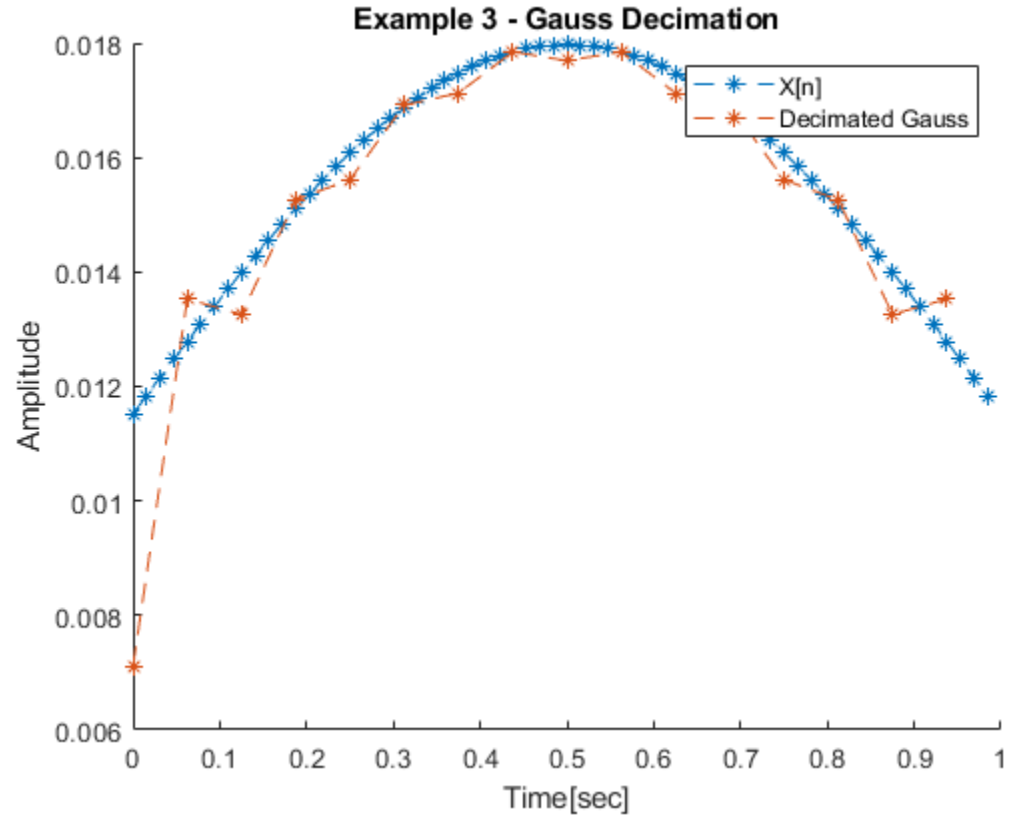
- $x_1[n]$ - still can be seen as periodic wave (if the F_{cutoff} high enough), otherwise will be filtered out.
- $x_2[n]$ - the reconstructed signal will be composed with less amount of deltas, so we will get more distortion/Gibbs phenomenom.

- $x_3[n]$ - similarly to $x_2[n]$ but $x_3[n]$ is smoother, so we expect get less distortion than in $x_2[n]$ case (because it contains less high frequencies).






Example 2 - Rect Decimation
X[n]
Decimated Rect
Amplitude
Time[sec]
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
-0.005
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Example 3 - Gauss Decimation
X[n]
Decimated Gauss
Amplitude
Time[sec]
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Published with MATLAB® R2024a