Introduction to MATLAB on Communication

Tutorial③

Dr. Victor B. Lawrence
& Ghalib Alshammri
galshamm@stevens.edu

CPE654: Design and Analysis of Network Systems
2017 Fall, Thursday 06:15 – 08:45 PM

Purpose and Objectives

- Discover MATLAB environment.
- Learn about MATLAB features.
- Learn about Basic Communication Channel.
- Discover Communication toolbox at MATLAB.
- Simulation Analysis of Communication
- Develop a beginner level MATLAB application.

Outline

- Overview
- Communication System Components
    - Analog Communications
    - Digital Communications
- Simulink Library
- Summary

Overview

* The ***Toolbox*** help user to create algorithms for commercial and defense wireless and wire systems.
* ***Functions*** for designing the ***physical layer*** of communications links, including source coding, channel coding, modulation, channel model, and equalization.
* Plots such as ***eye diagrams*** and ***constellations*** for visualizing communication signals.
* ***Graphical user interface*** for comparing the bit error rate BER of your system with a wide variety of analytical result.

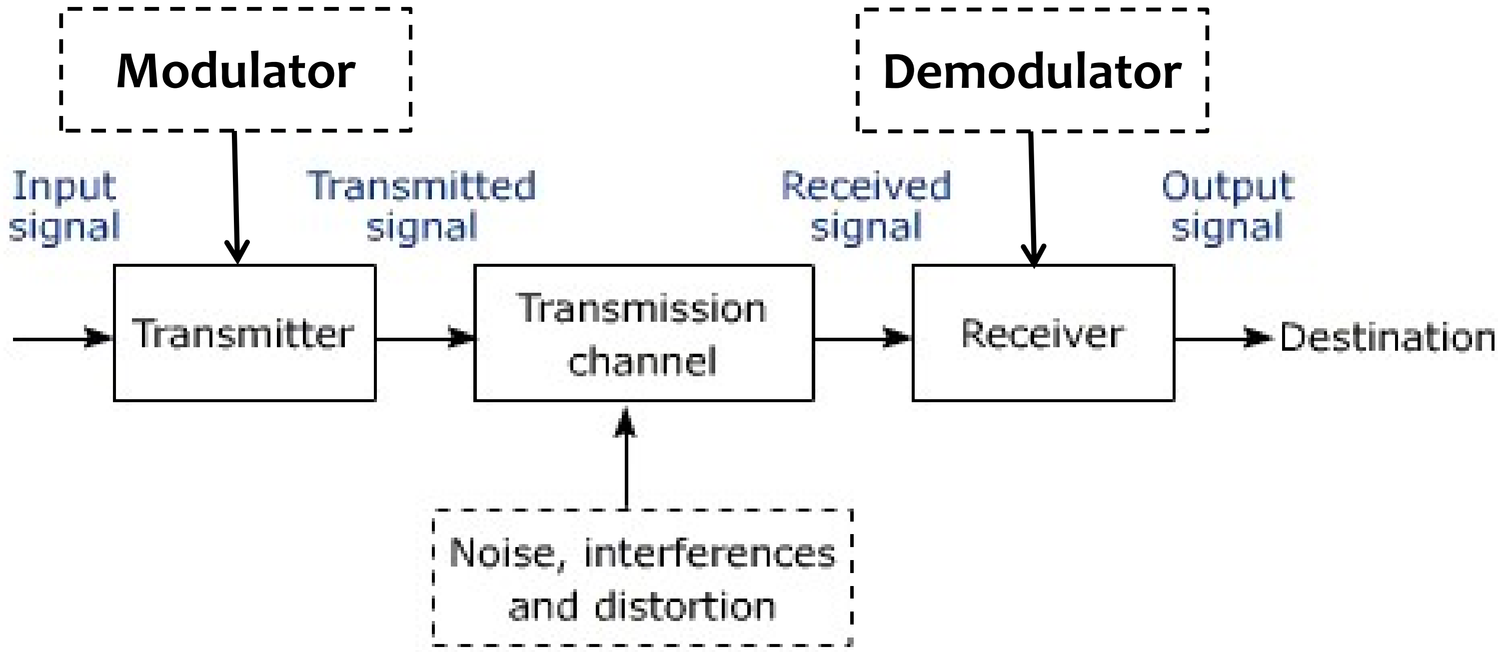
Communication System Components

Analog Communication System

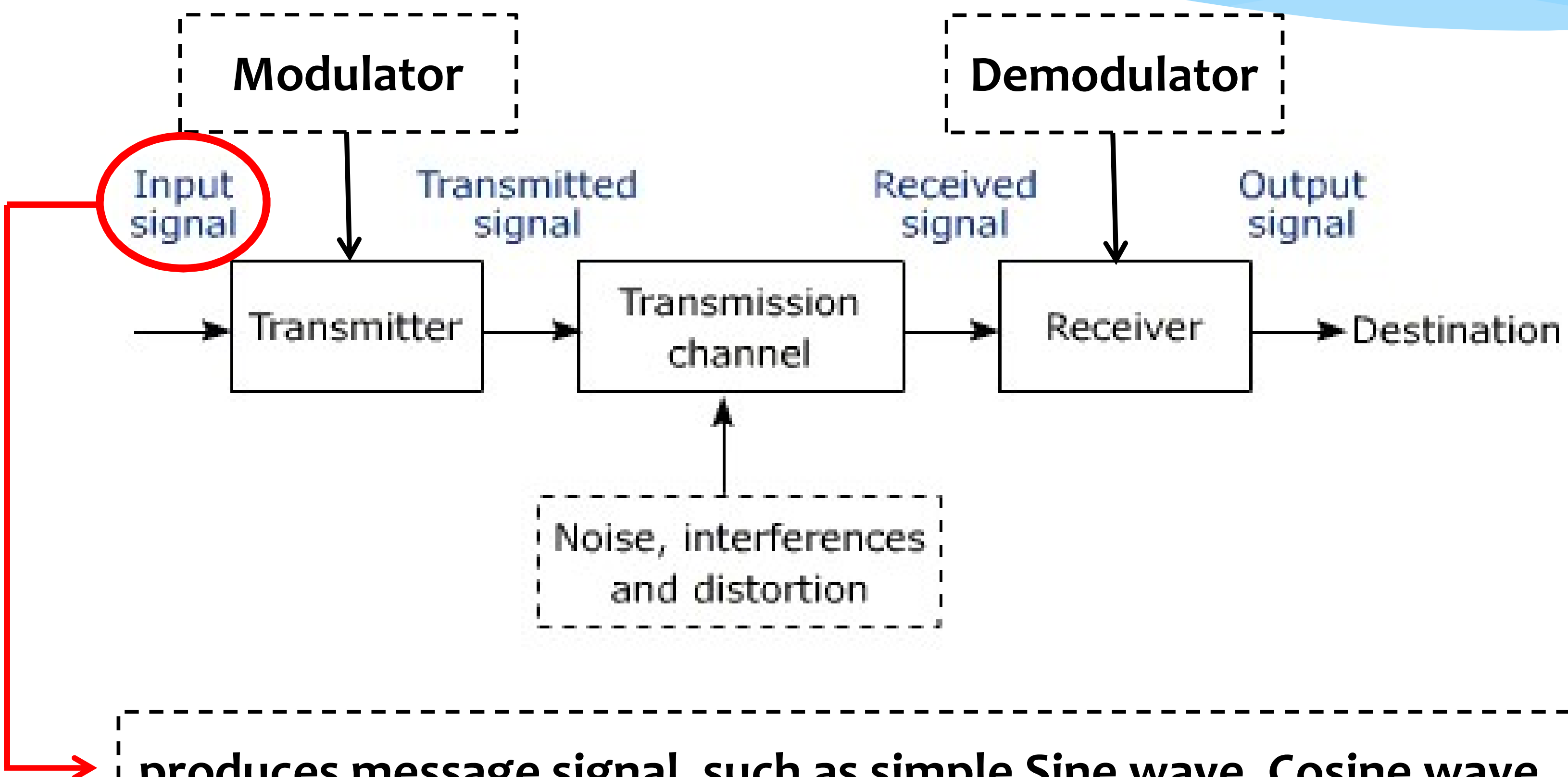
Digital Communication System

Analog Communication System

Analog Communication System



Simulation a Input Signal (Source)



produces message signal, such as simple Sine wave, Cosine wave, … etc.

Simulation a Input Signal (Source)

Generate message signal (e.g. Sine Wave)

$$m(t) = V_m \sin(2\pi f_m t)$$

- ***Define time instants*** (1000 sample points)

```
t_min = 0;
t_max = 10^(-3);
step = (t_max – t_min) / 1000;
t = t_min : step : t_max;
```

- ***Define amplitude and frequency*** (initial phase is zero)

```
vm = 1;                %Amplitude
fm = 2 X 10 ^ 3;       %Frequency
```

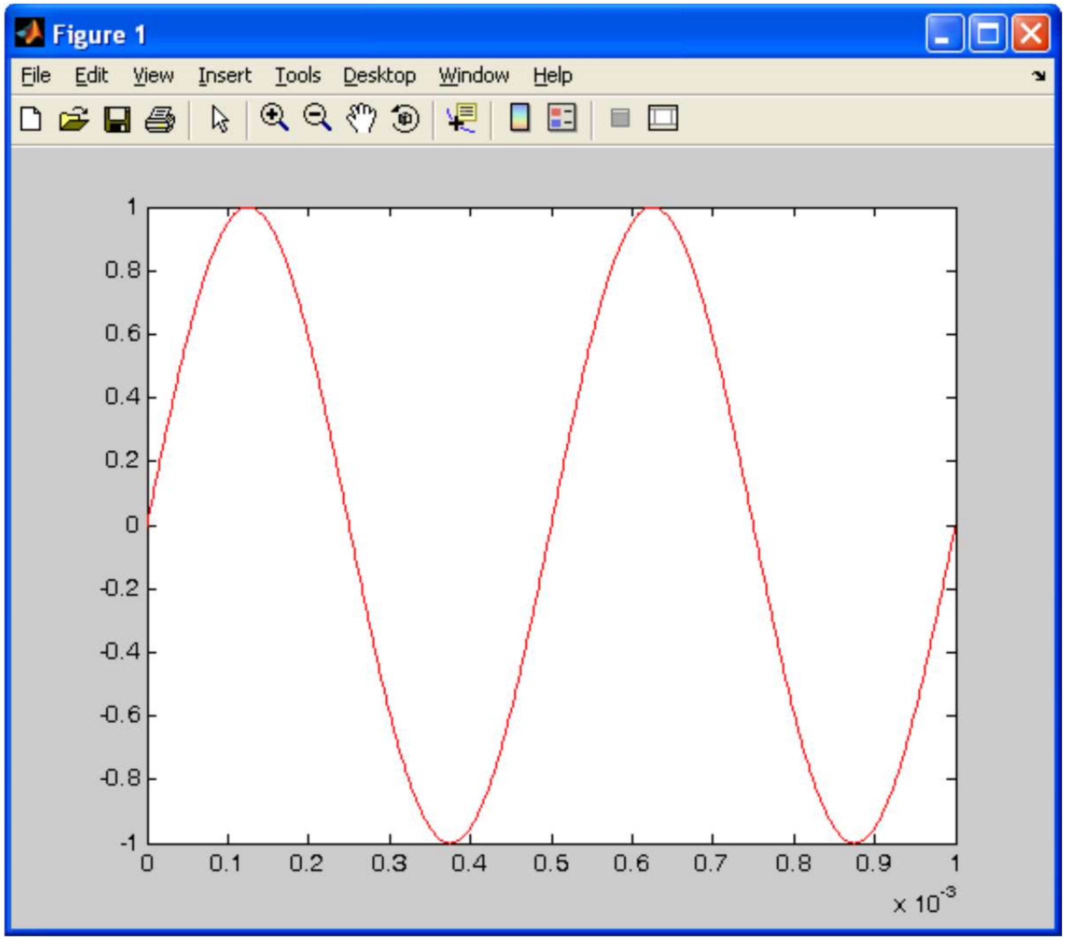
- ***Construct the signal***

$$\mathrm{m} = \mathrm{vm} * \sin(2 * pi * fm * t);$$

- ***View signal***

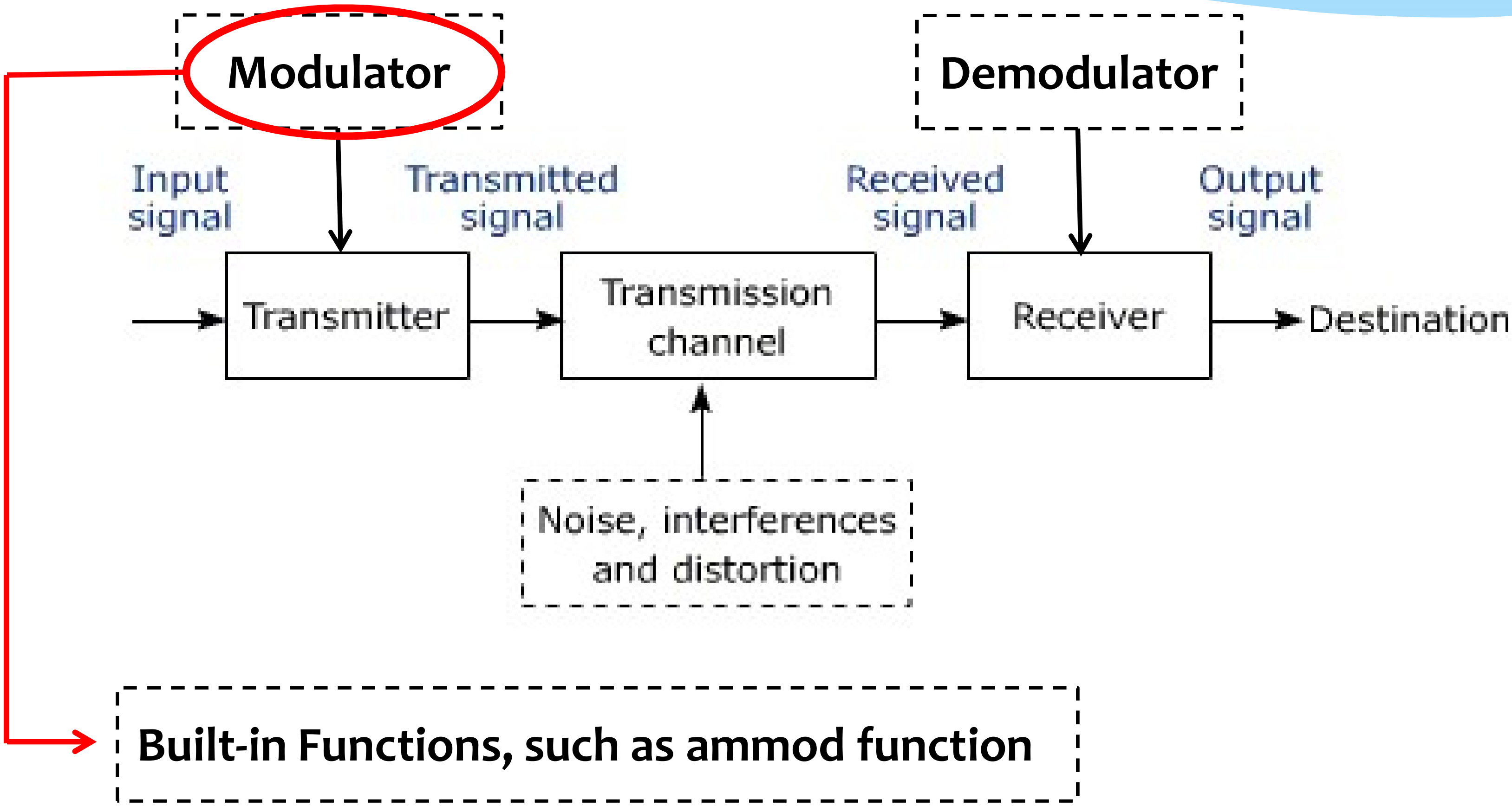
```
plot(t, m, ‘r’);
```

Simulation a Input Signal (Source)




```
t_min = 0;
t_max = 10^(-3);
step = (t_max – t_min) / 1000;
t = t_min : step : t_max;
vm = 1;                %Amplitude
fm = 2 X 10 ^ 3;       %Frequency
m = vm * sin(2 * pi * fm * t);
plot(t, m, ‘r’);
```

Modulation


Modulator
Demodulator
Input signal
Transmitted signal
Received signal
Output signal
Transmitter
Transmission channel
Receiver
Destination
Noise, interferences and distortion
Built-in Functions, such as ammod function

Amplitude Modulation

Syntax

```
y = ammod(x,Fc,Fs)
y = ammod(x,Fc,Fs,ini_phase)
y = ammod(x,Fc,Fs,ini_phase,carramp)
```

Where

x = Analog Signal

Fc = Carrier Signal

Fs = Sampling Frequency

ini_phase = Initial phase of the Carrier

carramp = Carrier Amplitude

Modulation - Example

Simulate with built-in functions

$fs = 8000;$ % Sampling rate is 8000 samples per second

$fc = 300;$ % $Carrier\ frequency\ in\ Hz$

$t = [0: 0.1 * fs]'/fs;$ % $Sampling\ time\ for$ 0.1 $second$

$m = \sin(20 * pi * t);$ % $Representation\ of\ the\ signal$

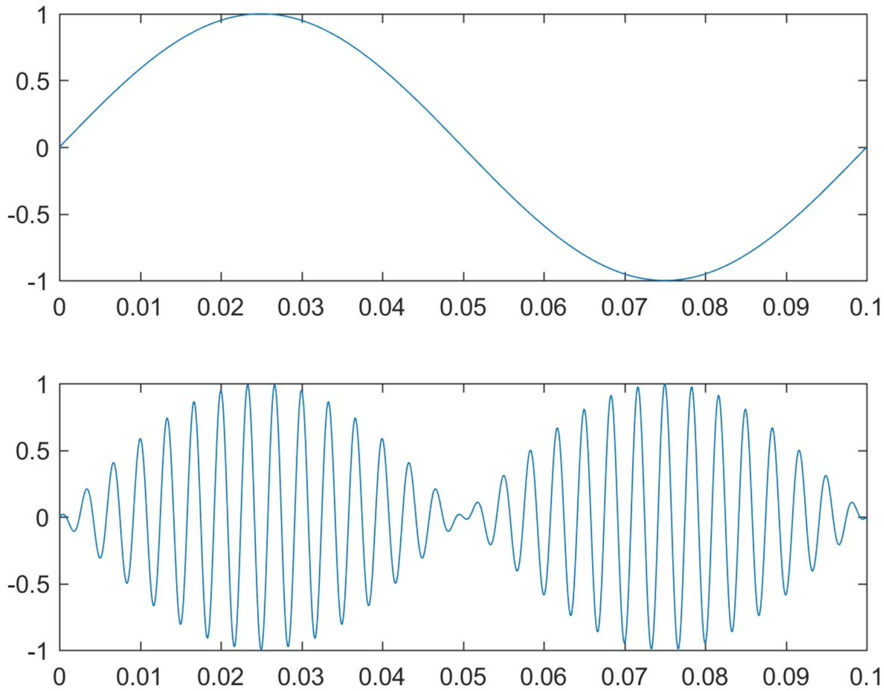
$v = ammod(m, fc, fs);$ % $Modulate\ m\ produce\ v$

$figure(1);$

$subplot(2,1,1); plot(t, m);$ % $Plot\ m\ on\ top$

$subplot(2,1,2); plot(t, v);$ % $Plot\ v\ below$

Modulation



Original Signal

Amplitude Modulation

Amplitude Demodulation

Syntax

```
z = amdemod(y,Fc,Fs)
z = amdemod(y,Fc,Fs,ini_phase)
z = amdemod(y,Fc,Fs,ini_phase,carramp)
z = amdemod(y,Fc,Fs,ini_phase,carramp,num,den)
```

Where

y = Received Analog Signal

Fc = Carrier Signal

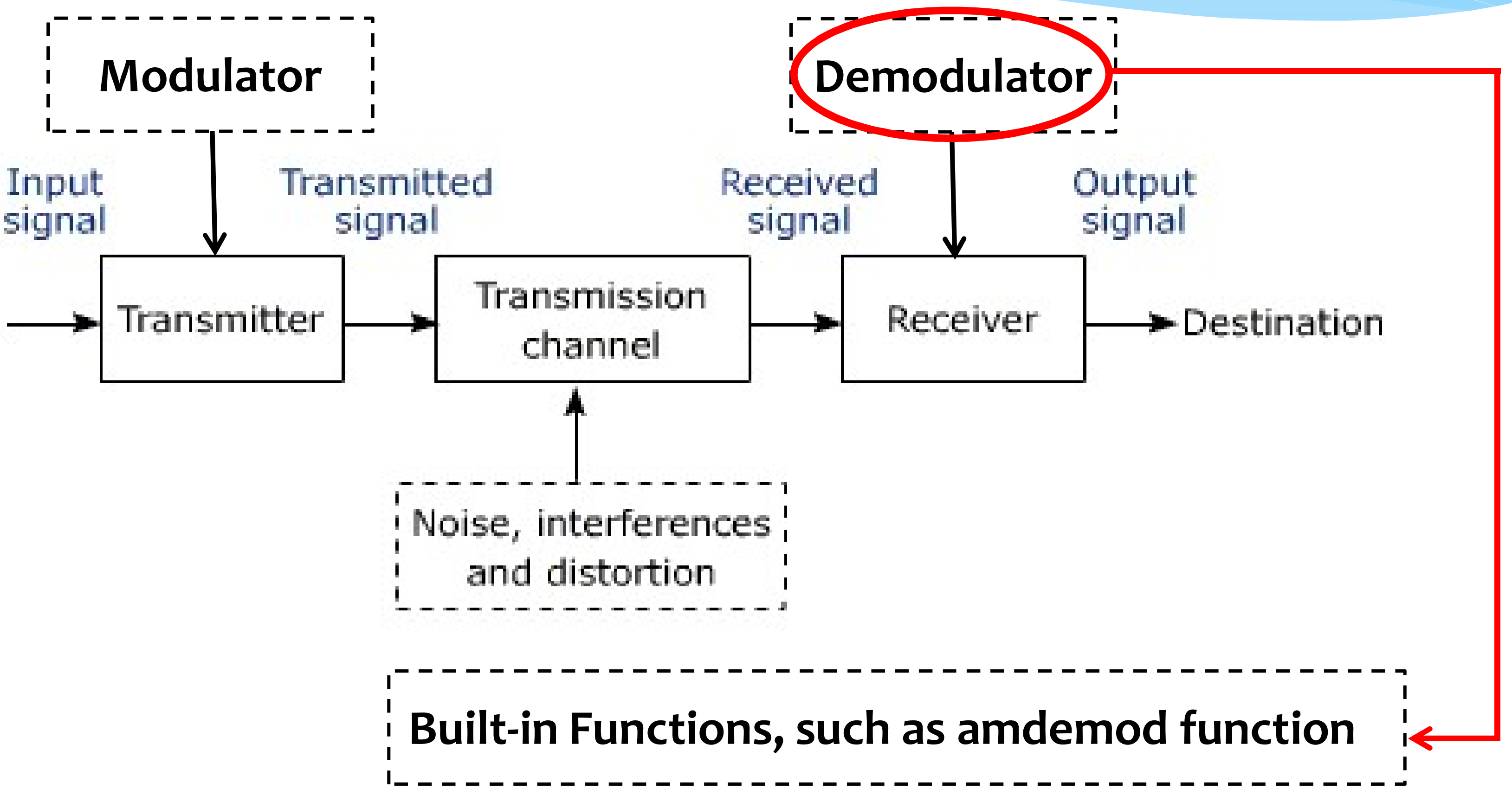
Fs = Sampling Frequency

ini_phase = Initial phase of the Carrier

carramp = Carrier Amplitude

num, den = Coefficients of butterworth low pass filter

Demodulation



Demodulation - Example

Simulate with built-in functions

$fs = 8000;$ % Sampling rate is 8000 samples per second

$fc = 300;$ $\% \, Carrier\ frequency\ in\ Hz$

$t = [0: 0.1 * fs]'/fs;$ $\% \, Sampling\ time\ for\ 0.1\ second$

$m = \sin(20\ *\ pi\ * t)\ ;$ $\% \, Representation\ of\ the\ signal$

$v = ammod(m, fc, fs);$ $\% \, Modulate\ m\ produce\ v$

$figure(1);$

$subplot(2, 1, 1); plot(t, m);$ $\% \, Plot\ m\ on\ top$

$subplot(2, 1, 2); plot(t, v);$ $\% \, Plot\ v\ below$

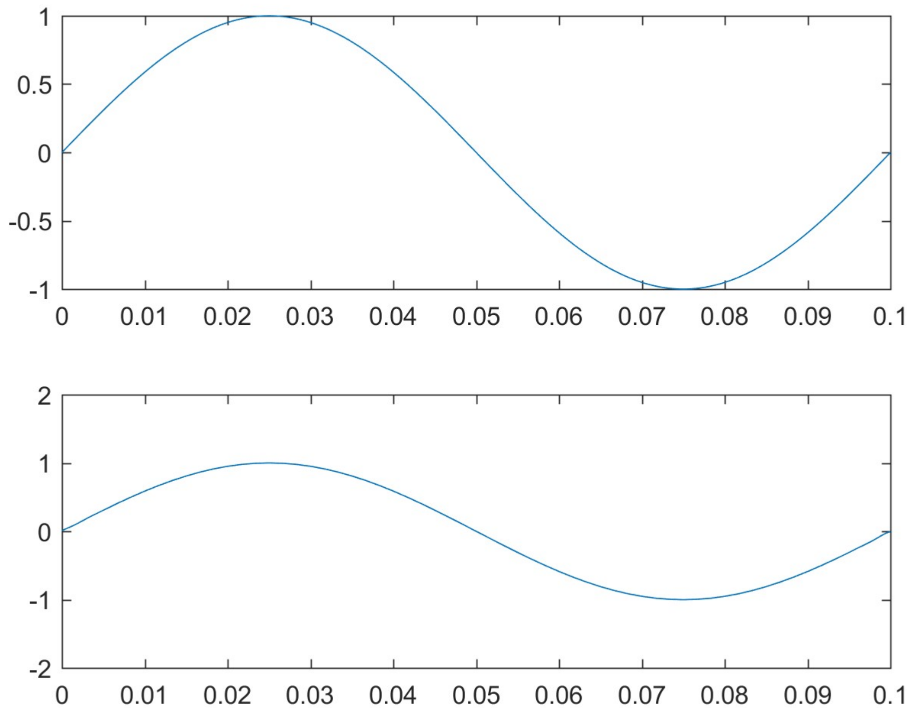
$mr = amdemod(v, fc, fs);$ $\% \, Demodulate\ v\ to\ produce\ m$

$figure(2);$

$subplot(2, 1, 1); plot(t, m);$ $\% \, Plot\ m\ on\ top$

$subplot(2, 1, 2); plot(t, mr);$ $\% \, Plot\ v\ below$

Demodulation



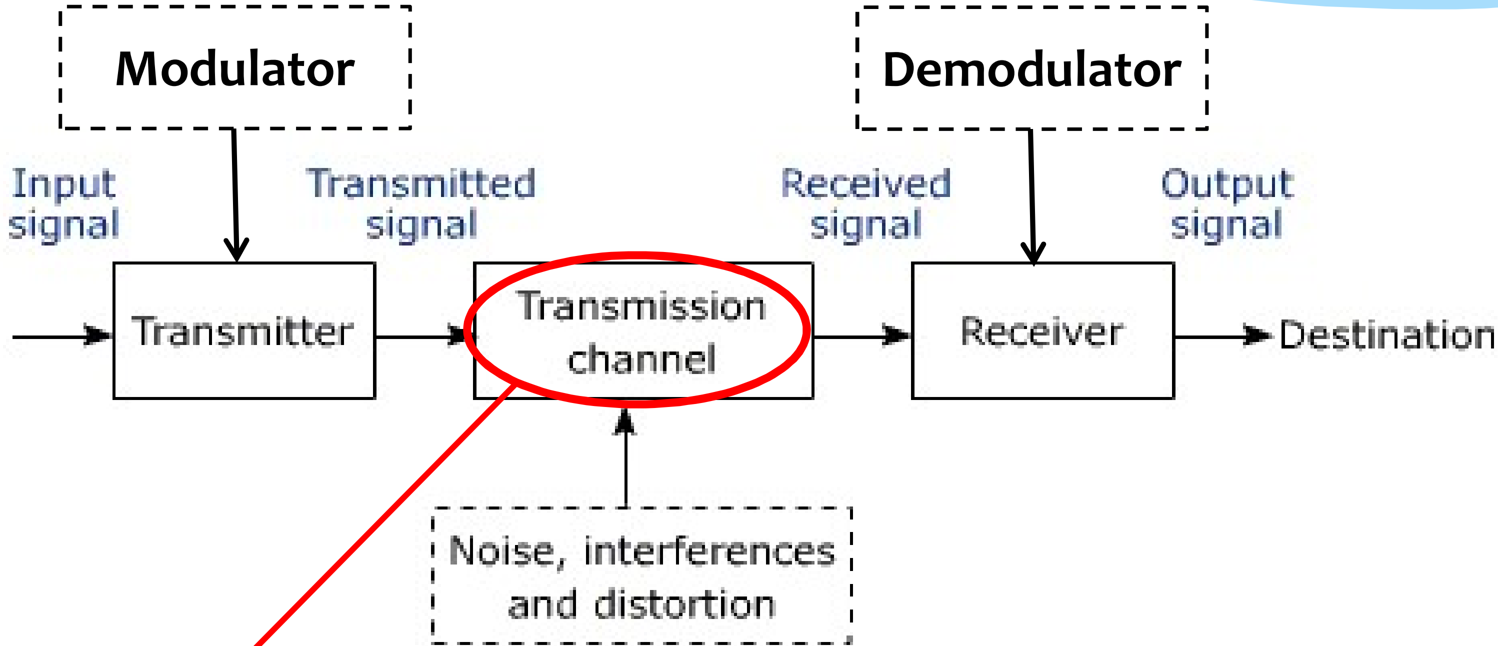
Amplitude Modulation

Amplitude Demodulation

Analog Modulation and Demodulation Functions

Functions	Description
ammod	Amplitude modulation
amdemod	Amplitude demodulation
fmmod	Frequency modulation
fmdemod	Frequency demodulation
pmmod	Phase modulation
pmdemod	Phase demodulation
ssbmod	Single Sideband Amplitude modulation
ssbdemod	Single Sideband Amplitude demodulation

Simulation Transmission Channel



Built-in Functions, such as awgn function

Additive White Gaussian Noise

Syntax

y = awgn(x, snr)
y = awgn(x, snr, sigpower)
y = awgn(x, snr, 'measured')

Where

x is Received Analog Signal

snr is the signal-to-noise ratio per sample

sigpower is the power of x in dBW

measured is the power of x is measured before adding noise

Additive White Gaussian Noise

The AWGN function in any programming language, the following procedure can be used.

- Assume, you have a vector xx to which an AWGN noise needs to be added for a given SNRSNR (specified in dB).
- Measure the power in the vector x

$$E_s = \frac{1}{L}\sum_{i=0}^{L-1}|x[i]|^2; \quad where \quad L = length(x)$$

- Convert given SNR in dB to linear scale (SNR_{lin}) and find the noise vector (from Gaussian distribution of specific noise variance) using the equations below

$$noise = \begin{cases} \sqrt{\frac{E_s}{SNR_{lin}}} * randn(1,L) & if \;\; x \;\; is \;\; real \\ \sqrt{\frac{E_s}{2*SNR_{lin}}} * [randn(1,L) + j * randn(1,L)] & if \;\; x \;\; is \;\; complex \end{cases}$$

- Finally add the generated noise vector to the signal x

$$y = x + noise$$

Additive White Gaussian Noise

```
function y = add_awgn_noise(x,SNR_dB)
    %y=awgn_noise(x,SNR) adds AWGN noise vector to signal 'x' to generate a
    %resulting signal vector y of specified SNR in dB
    rng('default');%set the random generator seed to default (for comparison on
    L=length(x);
    SNR = 10^(SNR_dB/10); %SNR to linear scale
    Esym=sum(abs(x).^2)/(L); %Calculate actual symbol energy
    N0=Esym/SNR; %Find the noise spectral density
    if(isreal(x)),
        noiseSigma = sqrt(N0);%Standard deviation for AWGN Noise when x is real
        n = noiseSigma*randn(1,L);%computed noise
    else
        noiseSigma=sqrt(N0/2);%Standard deviation for AWGN Noise when x is comp
        n = noiseSigma*(randn(1,L)+1i*randn(1,L));%computed noise
    end
    y = x + n; %received signal
end
```

Simulation Transmission Channel

Simulate with built-in functions

$fs = 8000;$ % Sampling rate is 8000 samples per second

$fc = 300;$ % $Carrier\ frequency\ in\ Hz$

$t = [0: 0.1 * fs]'/fs;$ % $Sampling\ time\ for\ 0.1\ second$

$m = \sin(20 * pi * t);$ % $Representation\ of\ the\ signal$

$v = ammod(m, fc, fs);$ % $Modulate\ m\ produce\ v$

mn = awgn(v, 10, 'measured'); *% add Additive White Gaussian Noise*

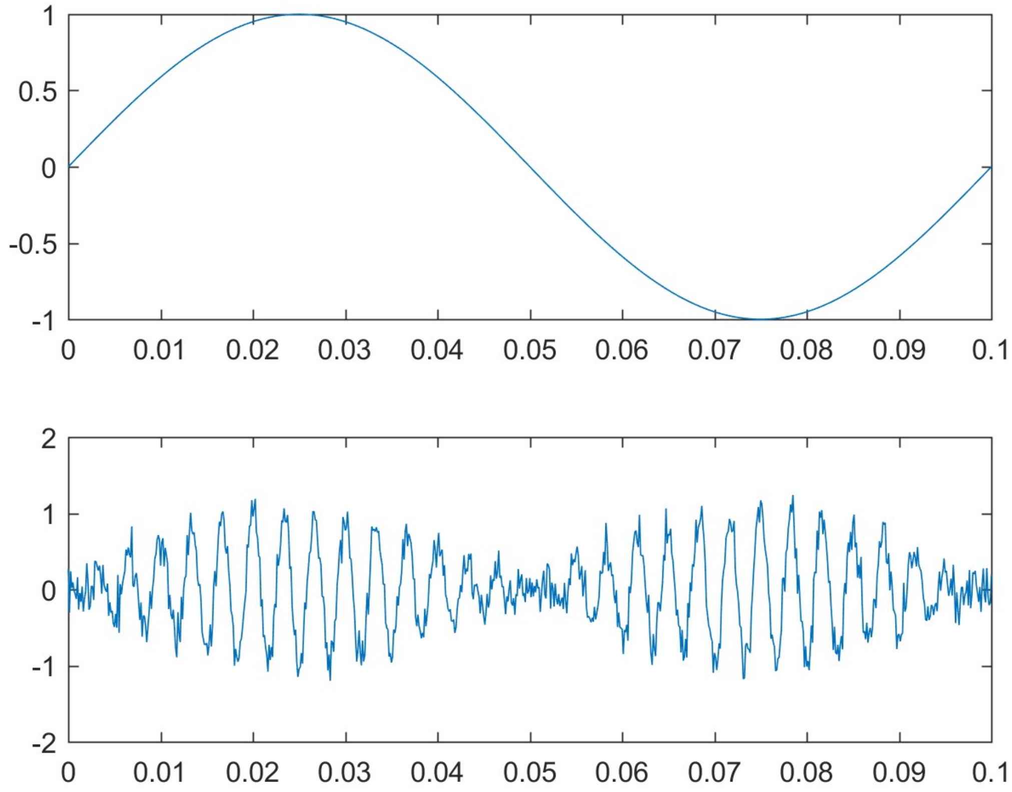
$mr = amdemod(v, fc, fs);$ % $Demodulate\ v\ to\ produce\ m$

$figure(1);$

$subplot(2,1,1); plot(t, m);$ % $Plot\ m\ on\ top$

$subplot(2,1,2); plot(t, mr);$ % $Plot\ v\ below$

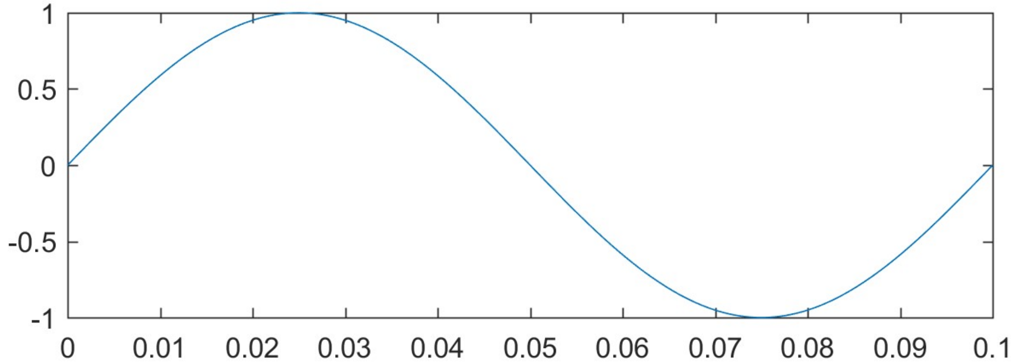
Simulation Transmission Channel



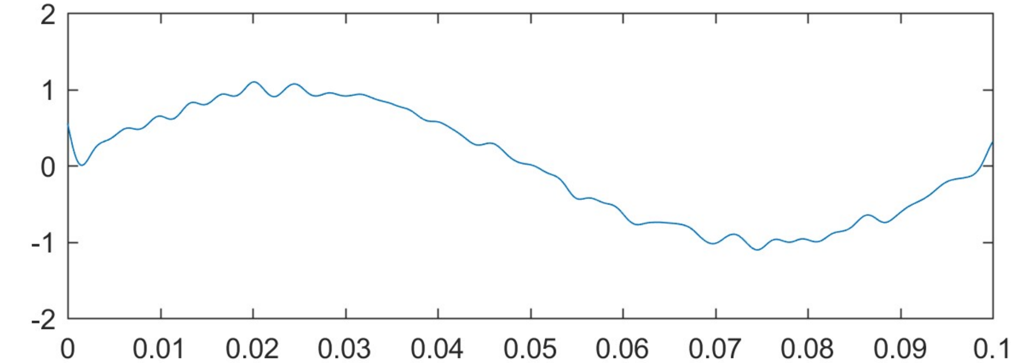
Amplitude Modulation

Amplitude Modulation After Adding noise

Simulation Transmission Channel



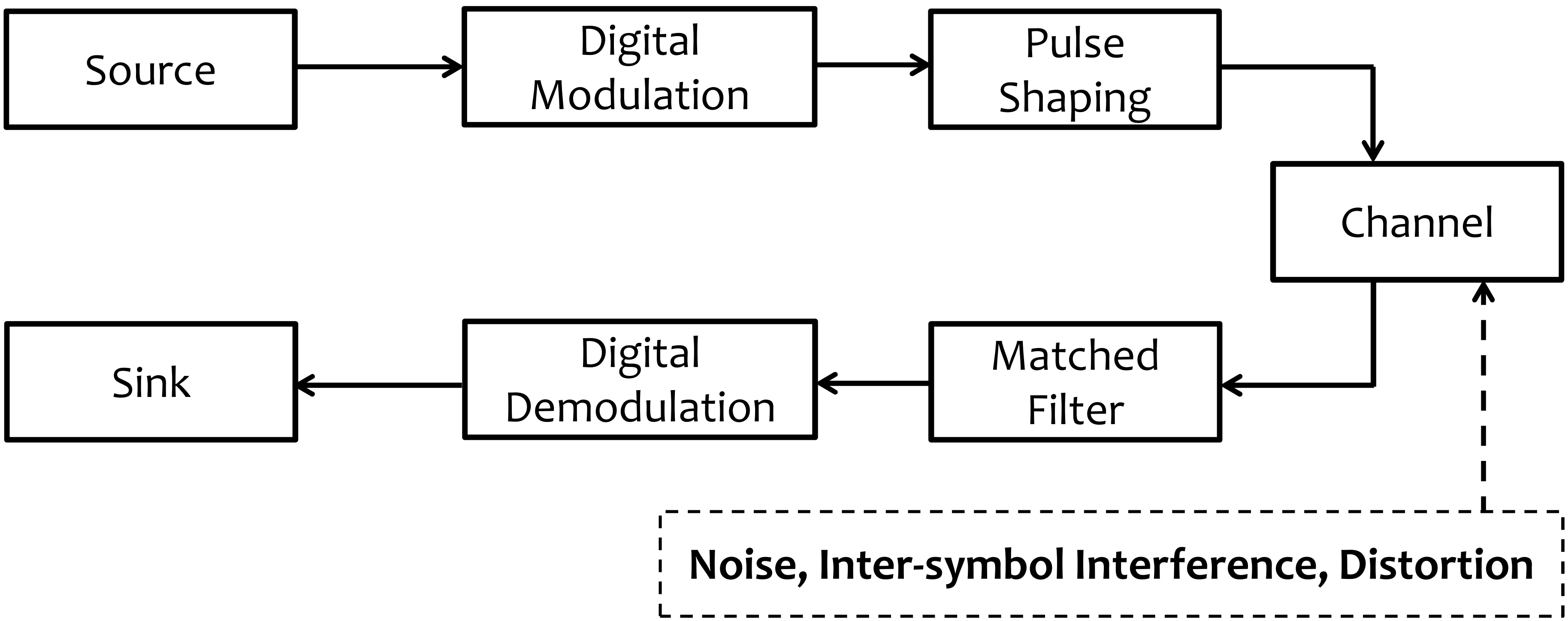
Amplitude Modulation



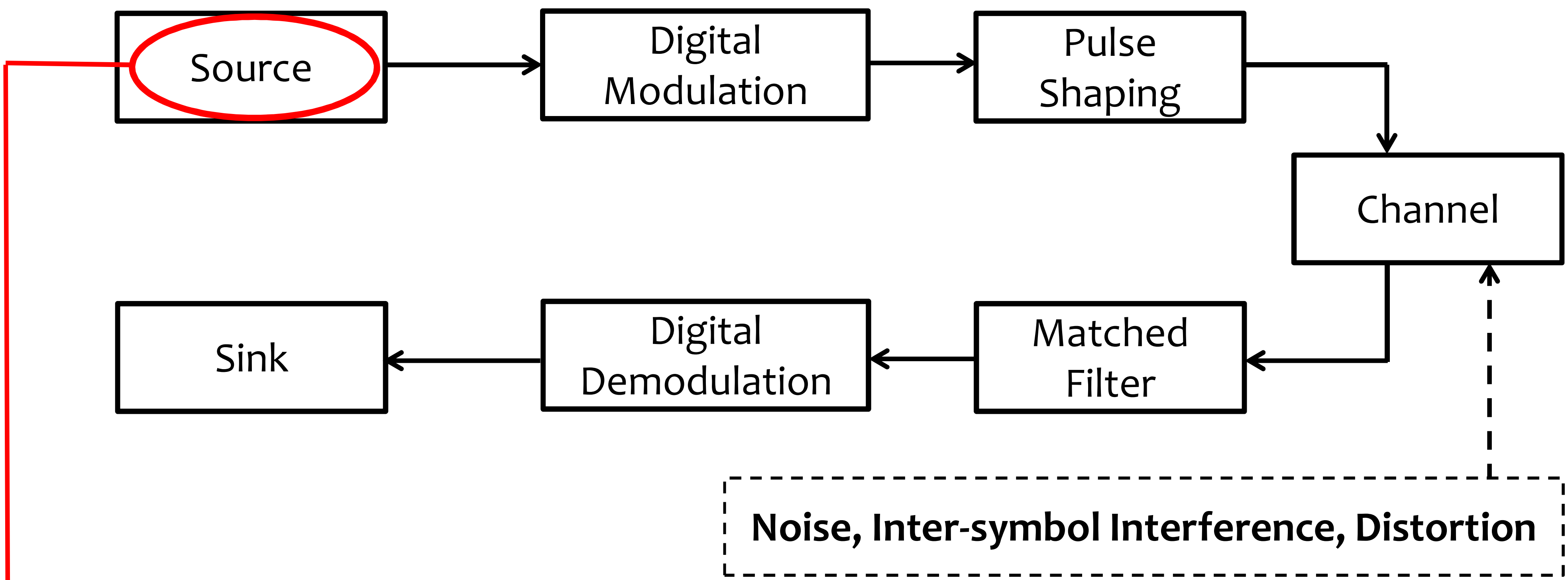
Amplitude Demodulation After Adding noise

Digital Communication System

Digital Communication System



Simulation a Input Signal (Source)



produces a message signal that is in a digital form, such as teletype signal

Simulation a Input Signal (Source)

Syntax

z = randint

z = randint(x)

z = randint(x, y)

Where

x = Binary Matrix m-by-m

y = Binary Matrix m-by-n

Functions	Description
randint	Generate matrix of Uniformly distributed Random Integers
randsrc	Generate Random matrix using prescribed alphabet
randerr	Generate bit error patterns
randi	Generate matrix of Uniformly distributed Random Integers

Simulation a Input Signal (Source)

Example1:

Generate a 10-by-10 matrix whose elements are uniformly distributed in the range from 0 to 7

x = randint(10, 10, [0, 7]);

Or

x = randint(10, 10, 8);

6	7	4	2	4	1	7	6	6	3
6	4	7	0	4	3	0	2	7	1
2	2	1	6	2	1	6	2	7	5
4	4	0	3	6	2	4	3	5	1
4	3	2	7	4	0	2	5	3	4
4	4	6	5	1	3	0	5	6	4
6	3	1	4	4	6	3	7	7	5
7	2	1	4	7	7	7	6	1	7
7	0	3	2	6	6	1	0	7	1
5	5	6	5	7	0	7	2	1	5

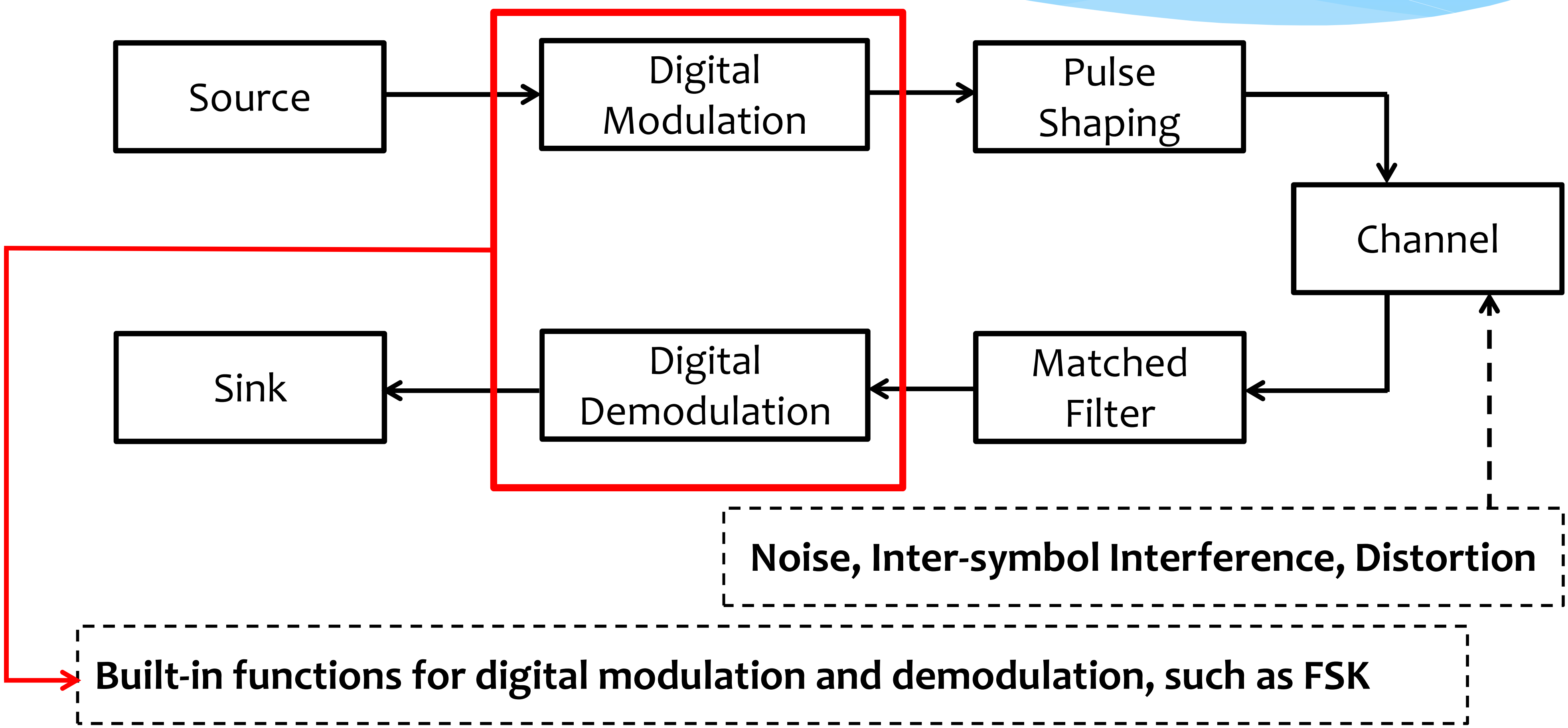
Example2:

Generate a 1-by-7 matrix whose elements are zeros and ones

x = randint(1, 7, [0, 1]);

0	0	1	0	1	0	1

Simulation Digital Modulation and Demodulation



Frequency Shift Keying Modulation

Syntax

y = fskmod(x, M, freq_step, nsamp)
y = fskmod(x, M, freq_step, nsamp, Fs)
y = fskmod(x, M, freq_step, nsamp, Fs, phase_cont)

Where

x is Digital Signal
M is The message signal must consist of integers between 0 and M-1
freq_sep is The desired separation between successive frequencies in Hz
nsamp is the number of samples per symbol in y
Fs is The sampling rate in Hertz
phase_cont is The phase continuity, set ‘cont’ to force phase continuity across boundaries in y, or ‘discount’ to avoid forcing phase continuity. The default is ‘cont’

Frequency Shift Keying Demodulation

Syntax

z = fskdemod(y, M, freq_step, nsamp)

z = fskdemod(y, M, freq_step, nsamp, Fs)

z = fskdemod(y, M, freq_step, nsamp, Fs, symbol_order)

Where

x is Digital Signal

M is The message signal must consist of integers between 0 and M-1

freq_sep is The desired separation between successive frequencies in Hz

nsamp is the required number of samples per symbol

Fs is The sampling rate in Hertz

symbol_order is The function uses a natural binary-coded ordering, set 'bin', and a Gray-coded ordering, set 'gray'

Frequency Shift Keying Demodulation

Set the simulation parameters.

```
M = 2;          % Modulation order
k = log2(M);    % Bits per symbol
EbNo = 5;       % Eb/No (dB)
Fs = 16;        % Sample rate (Hz)
nsamp = 8;      % Number of samples per symbol
freqsep = 10;   % Frequency separation (Hz)
```

Generate random data symbols.

```
data = randi([0 M-1],5000,1);
```

Apply FSK modulation.

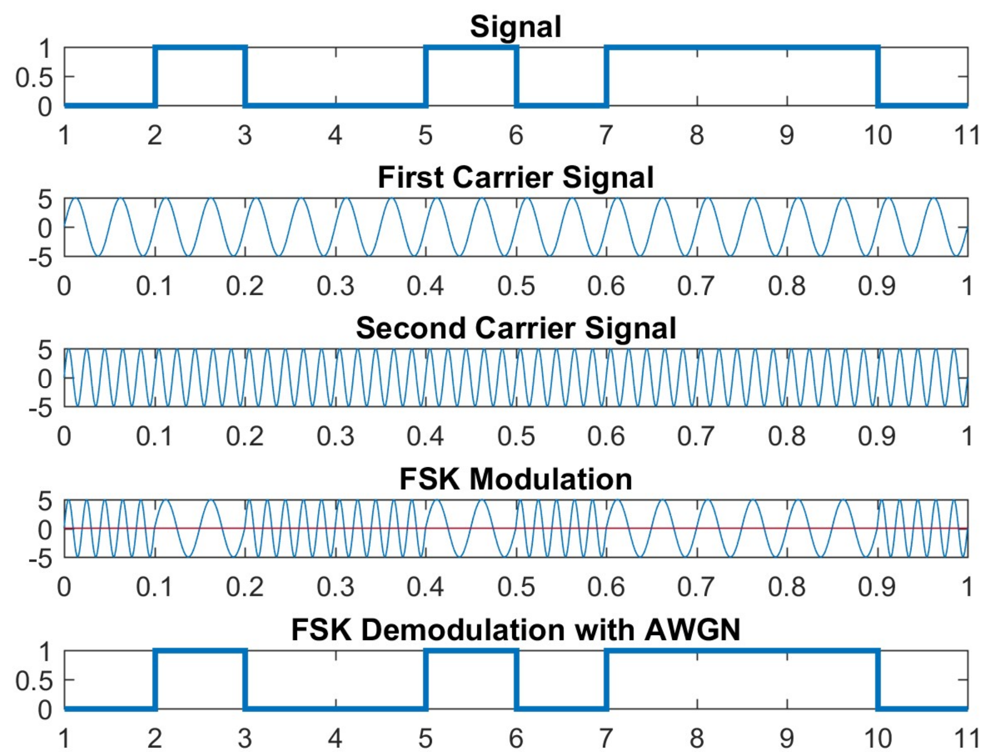
```
txsig = fskmod(data,M,freqsep,nsamp,Fs);
```

Pass the signal through an AWGN channel

```
rxSig  = awgn(txsig,EbNo+10*log10(k)-10*log10(nsamp),...
    'measured',[],'dB');
```

Demodulate the received signal.

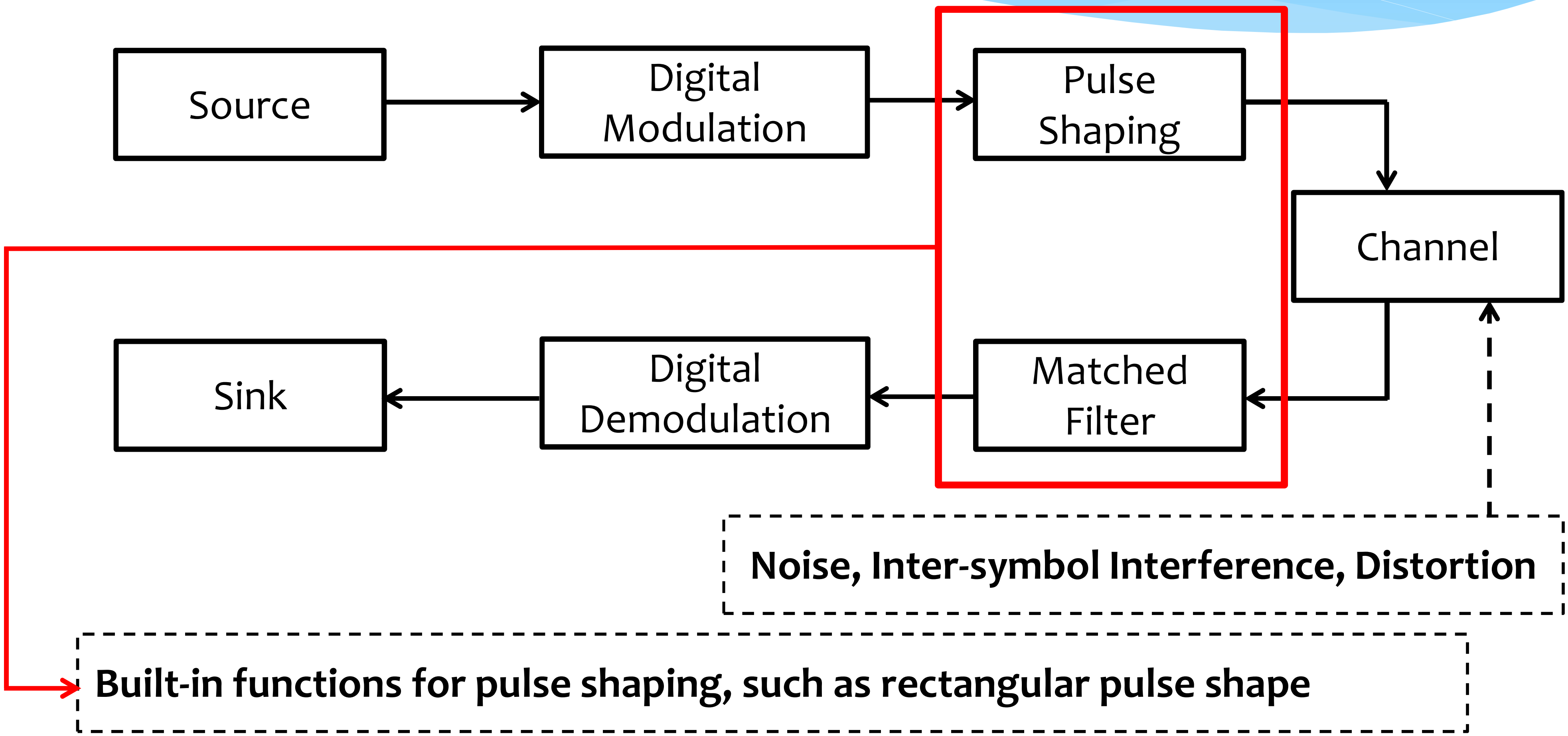
```
dataOut = fskdemod(rxSig,M,freqsep,nsamp,Fs);
```



Digital Modulation and Demodulation Functions

Functions	Description
fskmod	Frequency Shift Keying Modulation
fskdemod	Frequency Shift Keying Demodulation
pskmod	Phase Shift Keying Modulation
pskdemod	Phase Shift Keying Demodulation
mskmod	Minimum Shift Keying Modulation
mskdemod	Minimum Shift Keying Demodulation
qammod	Quadrature Amplitude Modulation
qamdemod	Quadrature Amplitude Demodulation

Pulse Shaping and Matched Filter



Rectangular Pulse Shaping

Syntax

y = rectpulse(x, nsamp);

Where

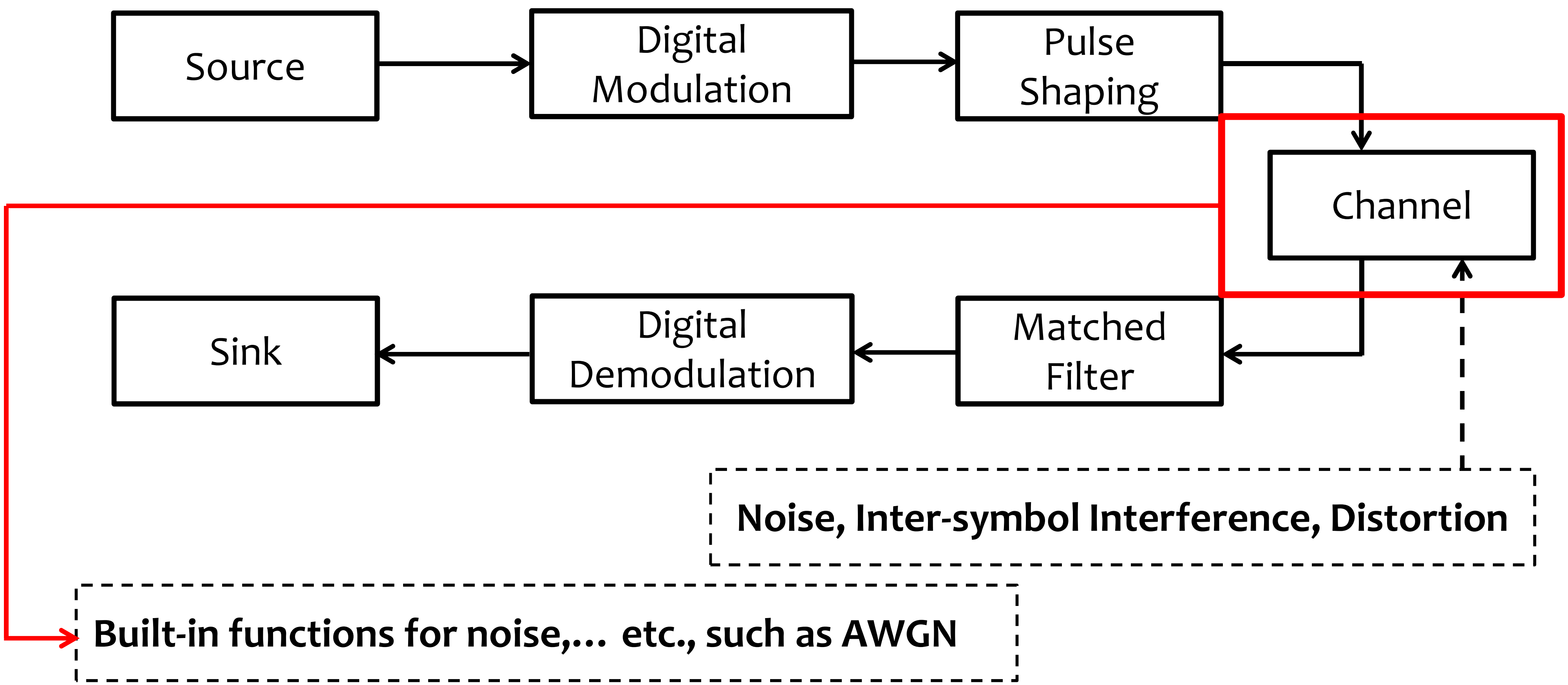
x is Digital Signal

nsamp is the number of samples per symbol in y

Pulse Shaping Functions

Functions	Description
rectpulse	Rectangular pulse shaping
rcosflt	Filter input signal using raised cosine filter
rcosine	Design raised cosine filter

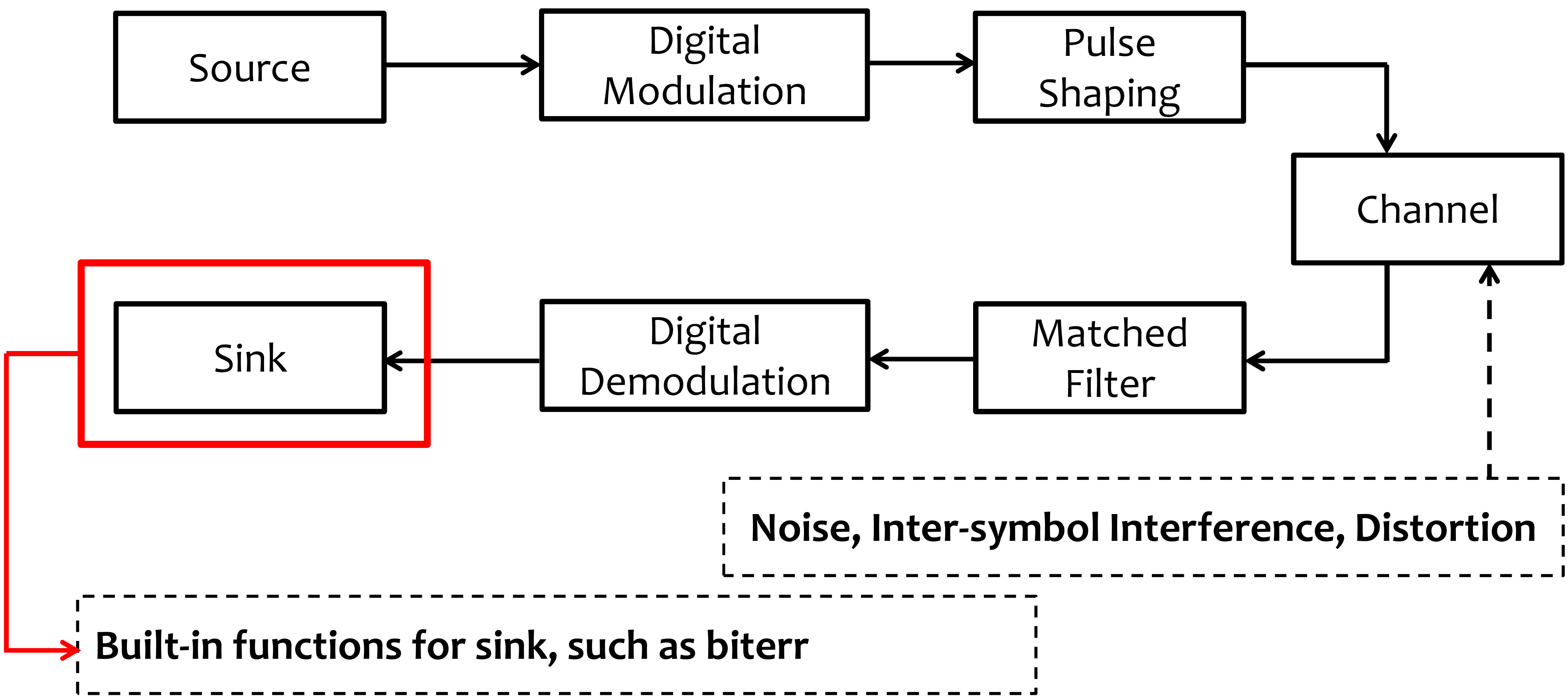
Channel Communication



Channel Communication Functions

Functions	Description
awgn	Add White Gaussian Noise to signal
rayleighchan	Construct Rayleigh Fading Channel Object
ricianchan	Construct Rician Fading Channel Object
bsc	Model Binary Symmetric Channel

Sink


Source
Digital Modulation
Pulse Shaping
Channel
Matched Filter
Digital Demodulation
Sink
Noise, Inter-symbol Interference, Distortion
Built-in functions for sink, such as biterr

Bit Error Rate

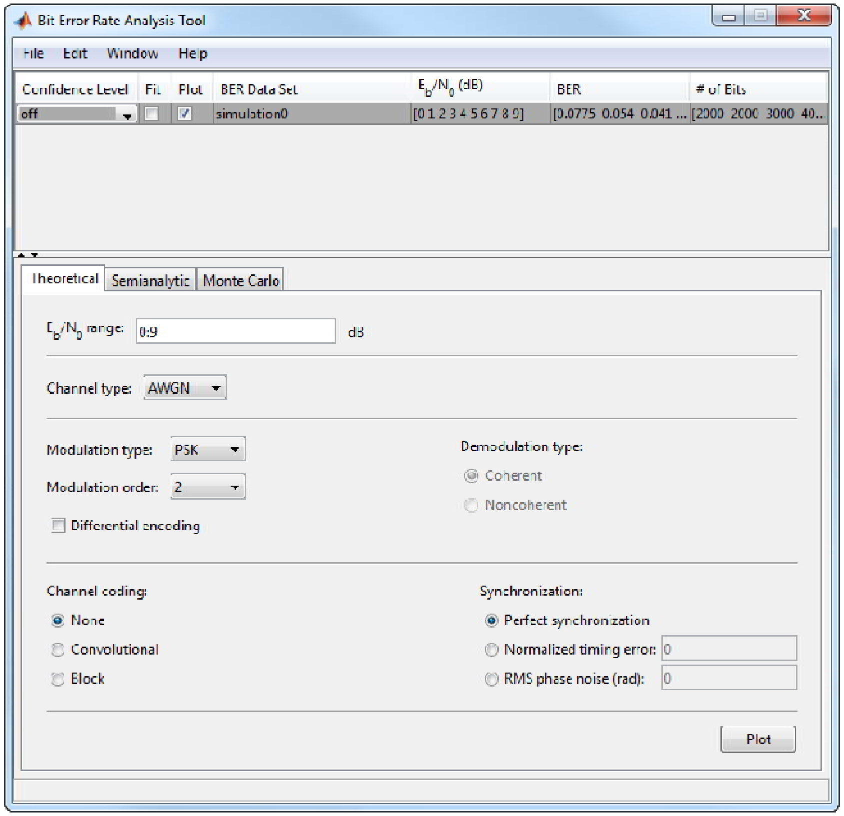
The ***biterr*** function compares unsigned binary representations of elements in x with those in y

Syntax

[number, ratio] = biterr(x, y);

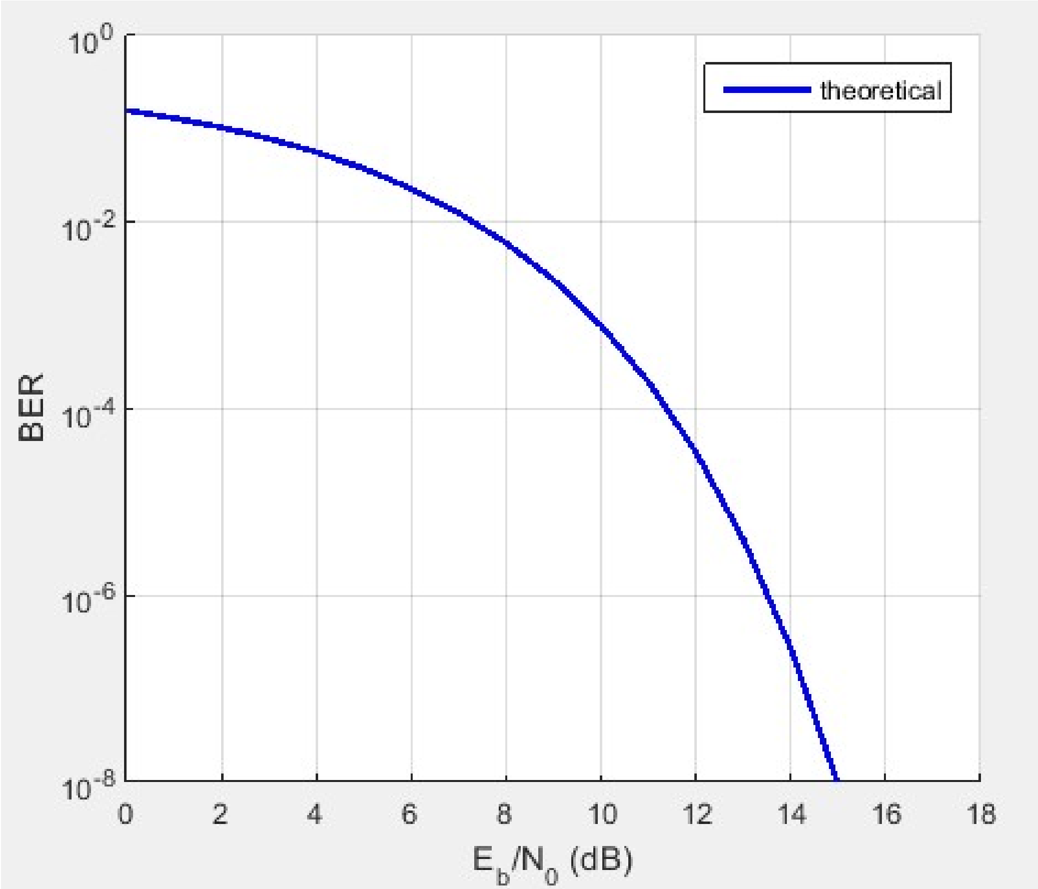
Where

x and **y** are matrices



Note: you can type ***bertool*** in **Command Windows** to show Bit Error Rate Form

Bit Error Rate

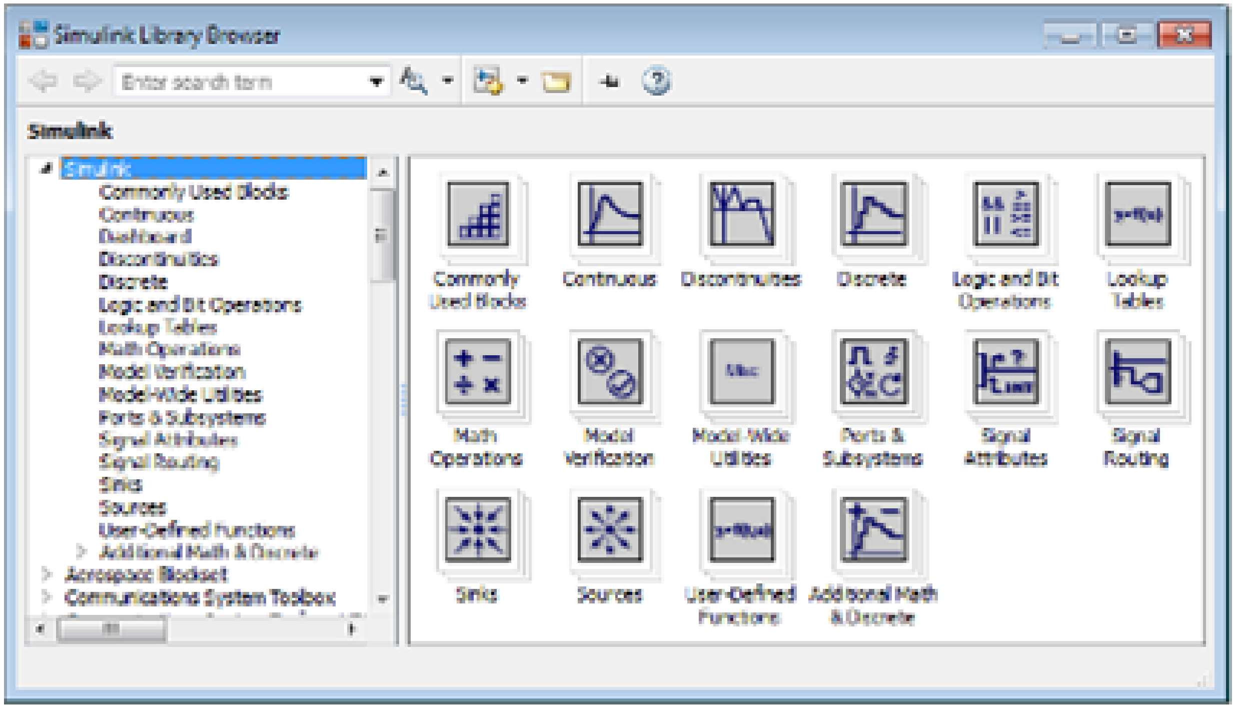


E_0/N_0range: 0 : 18
Channel Type: AWGN
Modulation Type: FSK
Modulation Order: 2
Demodulation Type: Coherent

Simulink Library

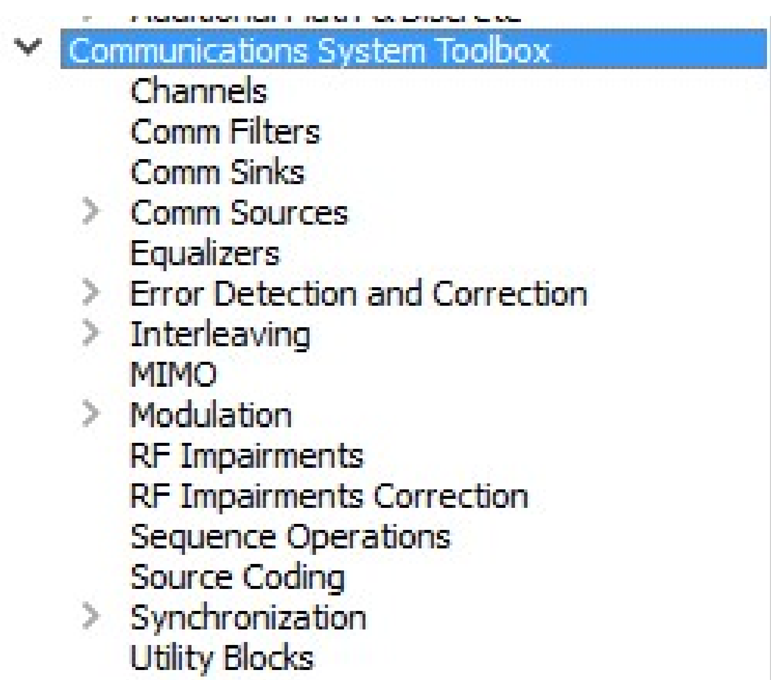
Starting Simulink

* From MATLAB command window, type **Simulink**
* Click on the '**Simulink Library Brower**' button

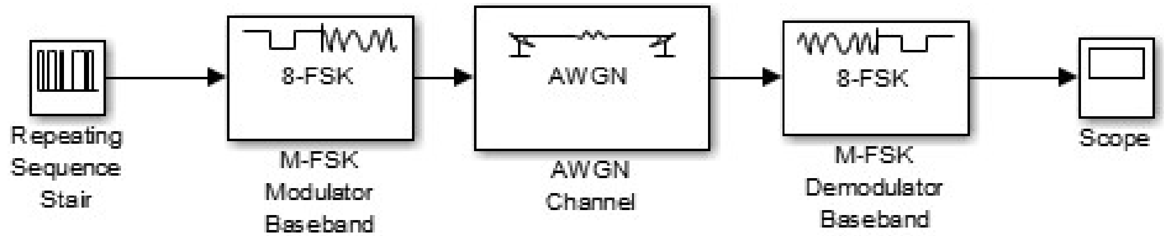


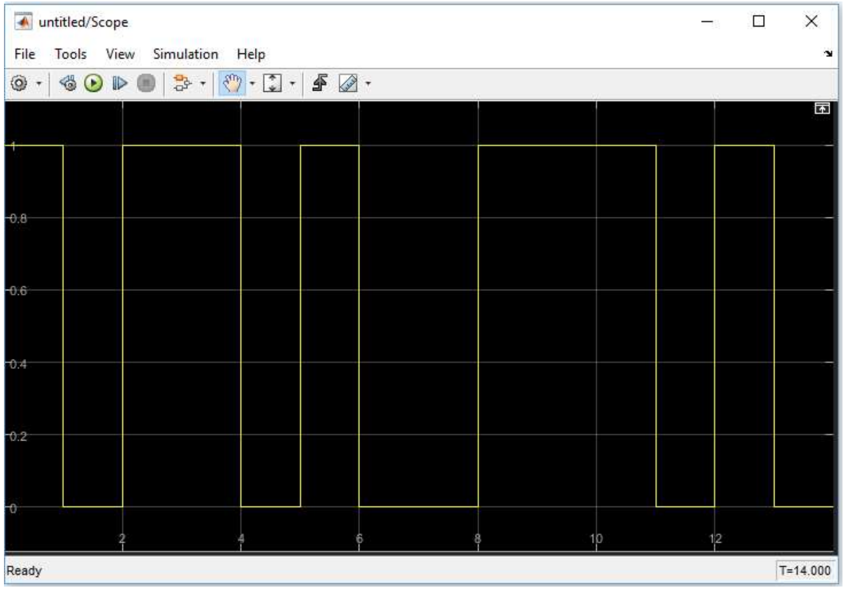
Communication System Toolbox

Communications System Toolbox provides algorithms and applications for the analysis, design, end-to-end simulation, and verification of communications systems in Simulink. Toolbox algorithms, including channel coding, modulation, MIMO, and OFDM, enable you to compose a physical layer model of your system. You can simulate your models to measure performance.



Building a System





Summary

We learn:

- Communication System Components
  - Analog Communication
  - Digital Communication
- Simulink Library

Note that it is brief in MATLAB concepts.