Part II

Introduction: From Theory to Simulation

Introduction: From Theory to Simulation

Introduction to digital communications and simulation of digital communications systems.

- A simple digital communication system and its theoretical underpinnings
  - Introduction to digital modulation
  - Baseband and passband signals: complex envelope
  - Noise and Randomness
  - The matched filter receiver
  - Bit-error rate
- Example: BPSK over AWGN, simulation in MATLAB

Outline

Part I: Learning Objectives

Elements of a Digital Communications System

Digital Modulation

Channel Model

Receiver

MATLAB Simulation

Learning Objectives

- Theory of Digital Communications.
  - Principles of Digital modulation.
  - Communications Channel Model: Additive, White Gaussian Noise.
  - The Matched Filter Receiver.
  - Finding the Probability of Error.
- Modeling a Digital Communications System in MATLAB.
  - Representing Signals and Noise in MATLAB.
  - Simulating a Communications System.
  - Measuring Probability of Error via MATLAB Simulation.

Outline

Part I: Learning Objectives

Elements of a Digital Communications System

Digital Modulation

Channel Model

Receiver

MATLAB Simulation

Elements of a Digital Communications System

Source: produces a sequence of information symbols b.

Transmitter: maps bit sequence to analog signal $s(t)$.

Channel: models corruption of transmitted signal $s(t)$.

Receiver: produces reconstructed sequence of information symbols $\hat{b}$ from observed signal $R(t)$.

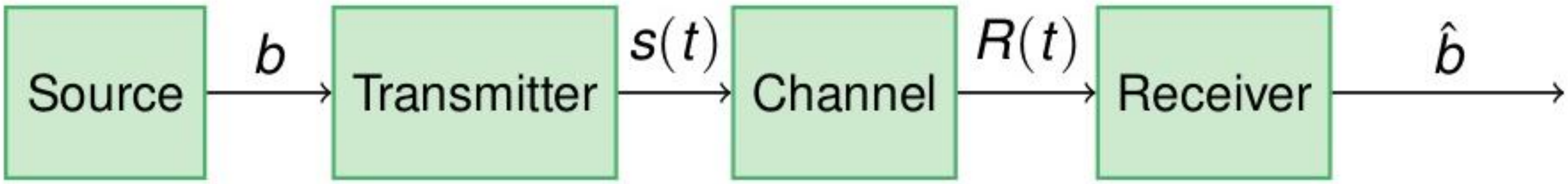


Figure: Block Diagram of a Generic Digital Communications System

The Source

- The source models the statistical properties of the digital information source.
- Three main parameters:

  Source Alphabet: list of the possible information symbols the source produces.

  - Example: $\mathcal{A} = \{0, 1\}$; symbols are called bits.
  - Alphabet for a source with M (typically, a power of 2) symbols: $\mathcal{A} = \{0, 1, \ldots, M-1\}$ or $\mathcal{A} = \{\pm 1, \pm 3, \ldots, \pm(M-1)\}$.
  - Alphabet with positive and negative symbols is often more convenient.
  - Symbols may be complex valued; e.g., $\mathcal{A} = \{\pm 1, \pm j\}$.

A priori Probability: relative frequencies with which the source produces each of the symbols.

- Example: a binary source that produces (on average) equal numbers of 0 and 1 bits has $\pi_0 = \pi_1 = \frac{1}{2}$.
- Notation: π_n denotes the probability of observing the n-th symbol.
- Typically, a-priori probabilities are all equal, i.e., $\pi_n = \frac{1}{M}$.
- A source with M symbols is called an M-ary source.
  - binary ($M = 2$)
  - ternary ($M = 3$)
  - quaternary ($M = 4$)

Symbol Rate: The number of information symbols the source produces per second. Also called the baud rate R.

- Closely related: information rate R_b indicates the number of bits the source produces per second.
- Relationship: $R_b = R \cdot \log_2(M)$.
- Also, $T = 1/R$ is the symbol period.

Bit 1	Bit 2	Symbol
0	0	0
0	1	1
1	0	2
1	1	3

Table: Two bits can be represented in one quaternary symbol.

Remarks

- This view of the source is simplified.
- We have omitted important functionality normally found in the source, including
  - error correction coding and interleaving, and
  - mapping bits to symbols.
- This simplified view is sufficient for our initial discussions.
- Missing functionality will be revisited when needed.

Modeling the Source in MATLAB

- **Objective:** Write a MATLAB function to be invoked as:

```
Symbols = RandomSymbols( N, Alphabet, Priors);
```

- The input parameters are
  - `N`: number of input symbols to be produced.
  - `Alphabet`: source alphabet to draw symbols from.
    - Example: `Alphabet = [1 -1];`
  - `Priors`: a priori probabilities for the input symbols.
    - Example:

      ```
      Priors = ones(size(Alphabet))/length(Alphabet);
      ```
- The output `Symbols` is a vector
  - with `N` elements,
  - drawn from `Alphabet`, and
  - the number of times each symbol occurs is (approximately) proportional to the corresponding element in `Priors`.

Reminders

- MATLAB's basic data units are vectors and matrices.
  - Vectors are best thought of as lists of numbers; vectors often contain samples of a signal.
  - There are many ways to create vectors, including
    - Explicitly: `Alphabet = [1 -1];`
    - Colon operator: `nn = 1:10;`
    - Via a function: `Priors=ones(1,5)/5;`
  - This leads to very concise programs; `for`-loops are rarely needed.
- MATLAB has a very large number of available functions.
  - Reduces programming to combining existing building blocks.
  - Difficulty: find out what is available; use built-in **`help`**.

Writing a MATLAB Function

- A MATLAB function must
  - begin with a line of the form

    ```
    function [out1,out2] = FunctionName(in1, in2, in3)
    ```
  - be stored in a file with the same name as the function name and extension '.m'.
  - For our symbol generator, the file name must be `RandomSymbols.m` and
  - the first line must be

    ```
    function Symbols = RandomSymbols(N, Alphabet, Priors)
    ```

Writing a MATLAB Function

- A MATLAB function should
  - have a second line of the form

    `%FunctionName - brief description of function`
    - This line is called the "H1 header."
  - have a more detailed description of the function and how to use it on subsequent lines.
    - The detailed description is separated from the H1 header by a line with only a `%`.
    - Each of these lines must begin with a `%` to mark it as a comment.
  - These comments become part of the built-in help system.

The Header of Function `RandomSymbols`

```
function Symbols = RandomSymbols(N, Alphabet, Priors)
% RandomSymbols - generate a vector of random information symbols
%
% A vector of N random information symbols drawn from a given
% alphabet and with specified a priori probabilities is produced.
%
% Inputs:
%   N        - number of symbols to be generated
%   Alphabet - vector containing permitted symbols
%   Priors   - a priori probabilities for symbols
%
% Example:
%   Symbols = RandomSymbols(N, Alphabet, Priors)
```

Algorithm for Generating Random Symbols

- For each of the symbols to be generated we use the following algorithm:
  - Begin by computing the cumulative sum over the priors.
    - Example: Let `Priors = [0.25 0.25 0.5]`, then the cumulative sum equals `CPriors = [0 0.25 0.5 1]`.
  - For each symbol, generate a uniform random number between zero and one.
    - The MATLAB function **`rand`** does that.
  - Determine between which elements of the cumulative sum the random number falls and select the corresponding symbol from the alphabet.
    - Example: Assume the random number generated is 0.3.
    - This number falls between the second and third element of `CPriors`.
    - The second symbol from the alphabet is selected.

MATLAB Implementation

- In MATLAB, the above algorithm can be "vectorized" to work on the entire sequence at once.

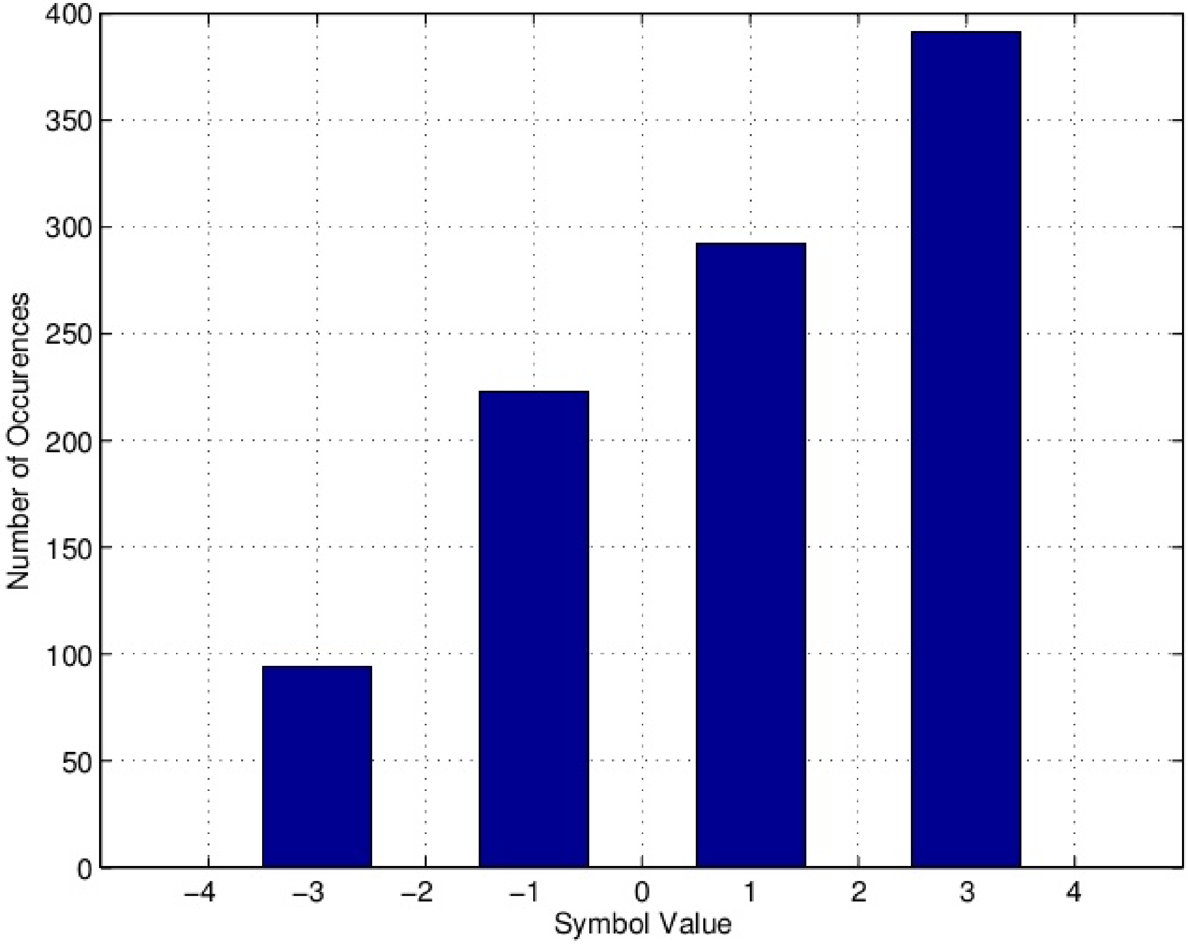
```
CPriors = [0 cumsum( Priors )];
rr = rand(1, N);

for kk=1:length(Alphabet)
    Matches = rr > CPriors(kk) & rr <= CPriors(kk+1);
    Symbols( Matches ) = Alphabet( kk );
end
```

Testing Function `RandomSymols`

- We can invoke and test the function `RandomSymbols` as shown below.
- A histogram of the generated symbols should reflect the specified a priori probabilities.

```
%% set parameters
N        = 1000;
Alphabet = [-3 -1 1 3];
Priors   = [0.1 0.2 0.3 0.4];

%% generate symbols and plot histogram
Symbols = RandomSymbols( N, Alphabet, Priors );
hist(Symbols, -4:4 );
grid
xlabel('Symbol␣Value')
ylabel('Number␣of␣Occurences')
```

Resulting Histogram



The Transmitter

- The transmitter translates the information symbols at its input into signals that are “appropriate” for the channel, e.g.,
  - meet bandwidth requirements due to regulatory or propagation considerations,
  - provide good receiver performance in the face of channel impairments:
    - noise,
    - distortion (i.e., undesired linear filtering),
    - interference.
- A digital communication system transmits only a discrete set of information symbols.
  - Correspondingly, only a discrete set of possible signals is employed by the transmitter.
  - The transmitted signal is an analog (continuous-time, continuous amplitude) signal.

Illustrative Example

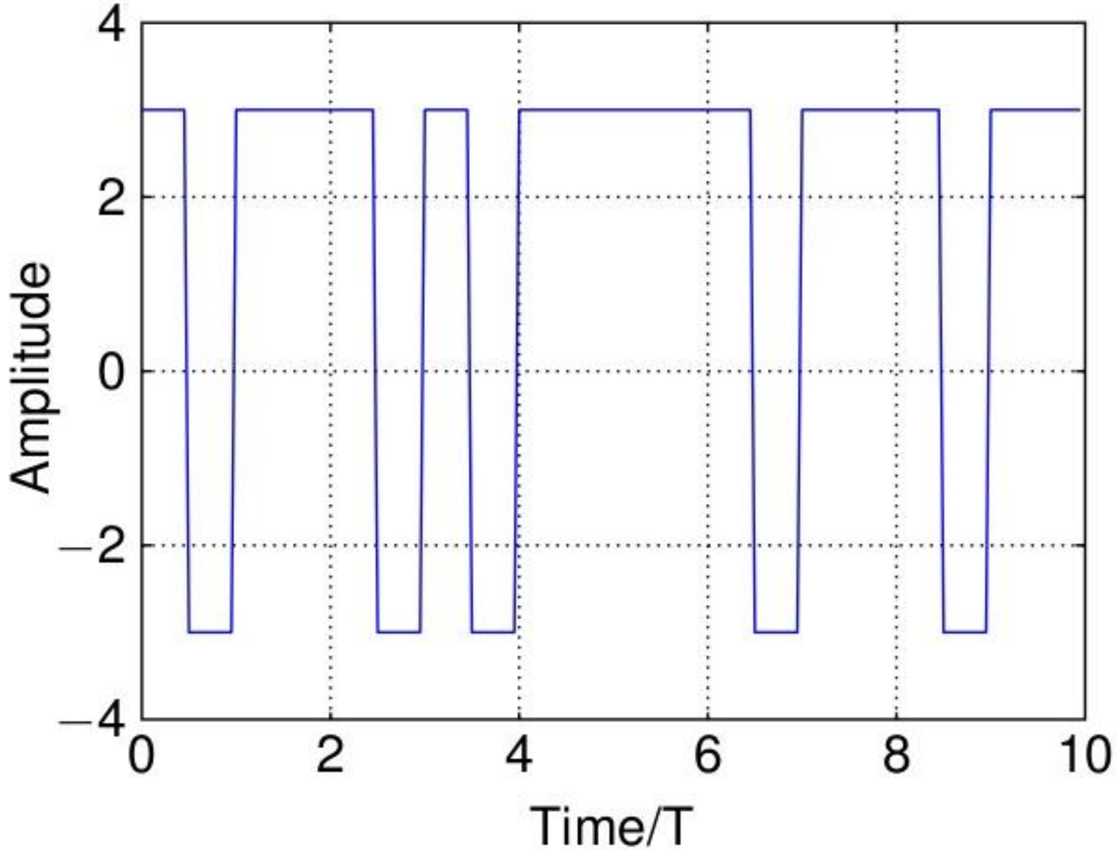
- The sources produces symbols from the alphabet $\mathcal{A} = \{0, 1\}$.
- The transmitter uses the following rule to map symbols to signals:
  - If the n-th symbol is $b_n = 0$, then the transmitter sends the signal

$$s_0(t) = \begin{cases} A & \text{for } (n-1)T \leq t < nT \\ 0 & \text{else.} \end{cases}$$

  - If the n-th symbol is $b_n = 1$, then the transmitter sends the signal

$$s_1(t) = \begin{cases} A & \text{for } (n-1)T \leq t < (n-\frac{1}{2})T \\ -A & \text{for } (n-\frac{1}{2})T \leq t < nT \\ 0 & \text{else.} \end{cases}$$

Symbol Sequence $b = \{1, 0, 1, 1, 0, 0, 1, 0, 1, 0\}$



MATLAB Code for Example

Listing : plot_TxExampleOrth.m

```
b = [ 1 0 1 1 0 0 1 0 1 0];       %symbol sequence
fsT = 20;                          % samples per symbol period
A = 3;

Signals = A*[ ones(1,fsT);         % signals, one per row
              ones(1,fsT/2) -ones(1,fsT/2)];

tt = 0:1/fsT:length(b)-1/fsT;      % time axis for plotting

%% generate signal ...
TXSignal = [];
for kk=1:length(b)
    TXSignal = [ TXSignal Signals( b(kk)+1, : ) ];
end
```

MATLAB Code for Example

Listing : plot_TxExampleOrth.m

```
%% ... and plot
plot(tt, TXSignal)
axis([0 length(b) -(A+1) (A+1)]);
grid
xlabel('Time/T')
```

The Communications Channel

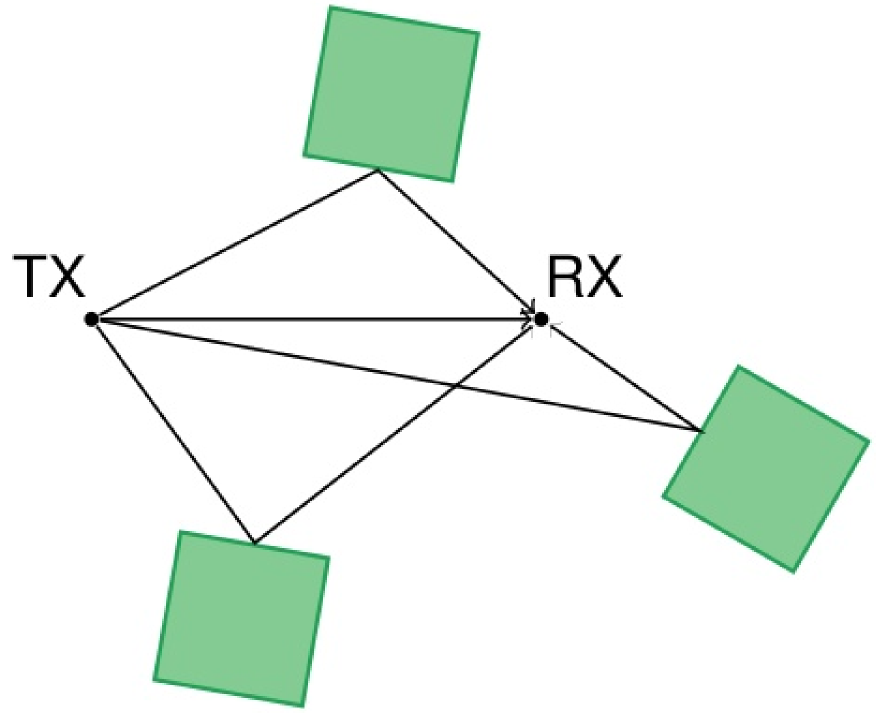
- The communications channel models the degradation the transmitted signal experiences on its way to the receiver.
- For wireless communications systems, we are concerned primarily with:
  - **Noise:** random signal added to received signal.
    - Mainly due to thermal noise from electronic components in the receiver.
    - Can also model interference from other emitters in the vicinity of the receiver.
    - Statistical model is used to describe noise.
  - **Distortion:** undesired filtering during propagation.
    - Mainly due to multi-path propagation.
    - Both deterministic and statistical models are appropriate depending on time-scale of interest.
    - Nature and dynamics of distortion is a key difference to wired systems.

Thermal Noise

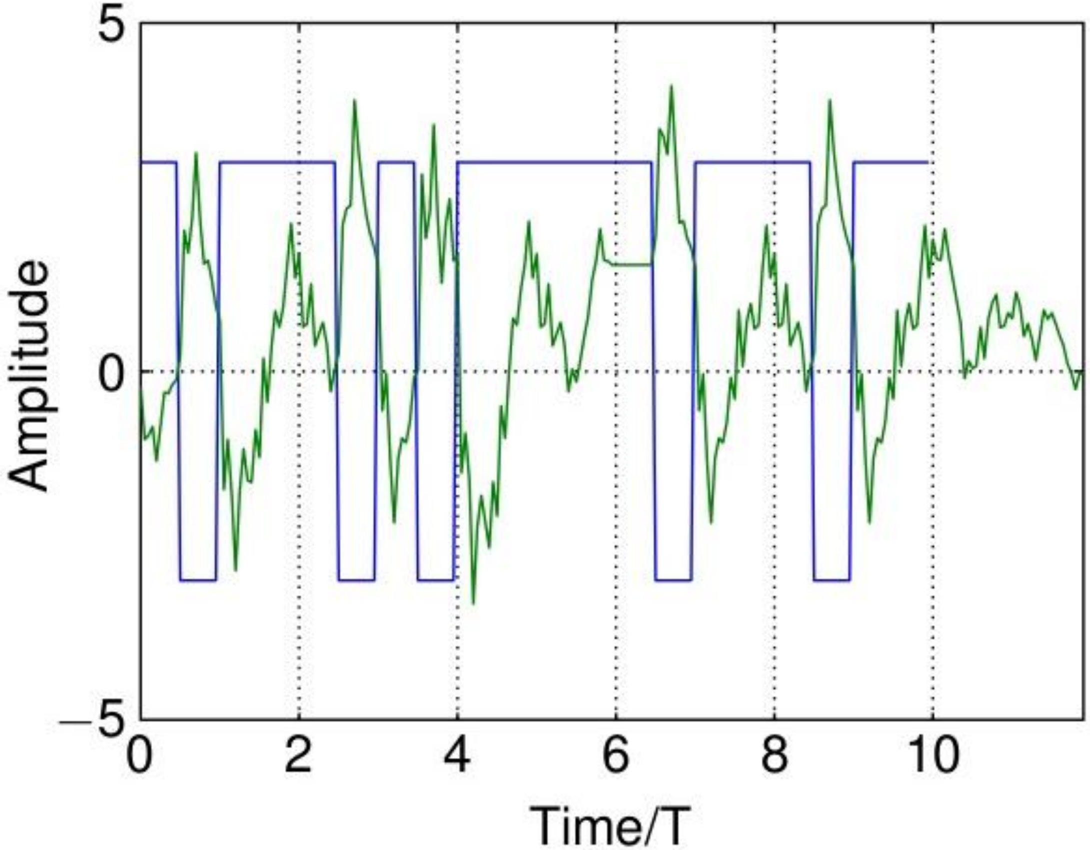
- At temperatures above absolute zero, electrons move randomly in a conducting medium, including the electronic components in the front-end of a receiver.
- This leads to a random waveform.
  - The power of the random waveform equals $P_N = kT_0B$.
    - k: Boltzmann's constant ($1.38 \cdot 10^{-23}$ Ws/K).
    - T_0: temperature in degrees Kelvin (room temperature ≈ 290 K).
    - For bandwidth equal to 1 MHz, $P_N \approx 4 \cdot 10^{-15}$ W (-114 dBm).
- Noise power is small, but power of received signal decreases rapidly with distance from transmitter.
  - Noise provides a fundamental limit to the range and/or rate at which communication is possible.

Multi-Path

- In a multi-path environment, the receiver sees the combination of multiple scaled and delayed versions of the transmitted signal.



Distortion from Multi-Path



- Received signal "looks" very different from transmitted signal.
- Inter-symbol interference (ISI).
- Multi-path is a very serious problem for wireless systems.

The Receiver

- The receiver is designed to reconstruct the original information sequence b.
- Towards this objective, the receiver uses
  - the received signal $R(t)$,
  - knowledge about how the transmitter works,
    - Specifically, the receiver knows how symbols are mapped to signals.
  - the a-priori probability and rate of the source.
- The transmitted signal typically contains information that allows the receiver to gain information about the channel, including
  - training sequences to estimate the impulse response of the channel,
  - synchronization preambles to determine symbol locations and adjust amplifier gains.

The Receiver

- The receiver input is an analog signal and it's output is a sequence of discrete information symbols.
  - Consequently, the receiver must perform analog-to-digital conversion (sampling).
- Correspondingly, the receiver can be divided into an analog front-end followed by digital processing.
  - Modern receivers have simple front-ends and sophisticated digital processing stages.
  - Digital processing is performed on standard digital hardware (from ASICs to general purpose processors).
  - Moore's law can be relied on to boost the performance of digital communications systems.

Measures of Performance

- The receiver is expected to perform its function optimally.
- **Question:** optimal in what sense?
  - Measure of performance must be statistical in nature.
    - observed signal is random, and
    - transmitted symbol sequence is random.
  - Metric must reflect the reliability with which information is reconstructed at the receiver.
- **Objective:** Design the receiver that minimizes the probability of a symbol error.
  - Also referred to as symbol error rate.
  - Closely related to bit error rate (BER).

Summary

- We have taken a brief look at the elements of a communication system.
  - Source,
  - Transmitter,
  - Channel, and
  - Receiver.
- We will revisit each of these elements for a more rigorous analysis.
  - **Intention:** Provide enough detail to allow simulation of a communication system.

Digital Modulation

- Digital modulation is performed by the transmitter.
- It refers to the process of converting a sequence of information symbols into a transmitted (analog) signal.
- The possibilities for performing this process are virtually without limits, including
  - varying, the amplitude, frequency, and/or phase of a sinusoidal signal depending on the information sequence,
  - making the currently transmitted signal on some or all of the previously transmitted symbols (modulation with memory).
- Initially, we focus on a simple, yet rich, class of modulation formats referred to as linear modulation.

Linear Modulation

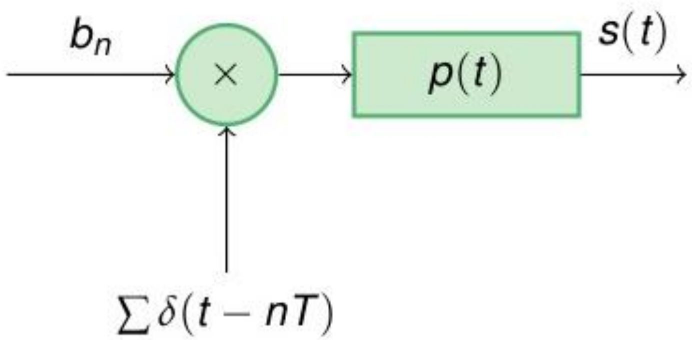
- Linear modulation may be thought of as the digital equivalent of amplitude modulation.
  - The instantaneous amplitude of the transmitted signal is proportional to the current information symbol.
- Specifically, a linearly modulated signal may be written as

$$s(t) = \sum_{n=0}^{N-1} b_n \cdot p(t - nT)$$

where,

  - b_n denotes the n-th information symbol, and
  - $p(t)$ denotes a pulse of finite duration.
  - Recall that T is the duration of a symbol.

Linear Modulation



- Note, that the expression

$$s(t) = \sum_{n=0}^{N-1} b_n \cdot p(t - nT)$$

is linear in the symbols b_n.

- Different modulation formats are constructed by choosing appropriate symbol alphabets, e.g.,
  - **BPSK:** $b_n \in \{1, -1\}$
  - **OOK:** $b_n \in \{0, 1\}$
  - **PAM:** $b_n \in \{\pm 1, \ldots, \pm(M-1)\}$.

Linear Modulation in MATLAB

- To simulate a linear modulator in MATLAB, we will need a function with a function header like this:

```
function Signal = LinearModulation( Symbols, Pulse, fsT )
% LinearModulation - linear modulation of symbols with given
%                    pulse shape
%
% A sequence of information symbols is linearly modulated. Pulse
% shaping is performed using the pulse shape passed as input
% parameter Pulse. The integer fsT indicates how many samples
% per symbol period are taken. The length of the Pulse vector may
% be longer than fsT; this corresponds to partial-response signal
%
% Inputs:
%   Symbols - vector of information symbols
%   Pulse   - vector containing the pulse used for shaping
%   fsT     - (integer) number of samples per symbol period
```

Linear Modulation in MATLAB

- In the body of the function, the sum of the pulses is computed.
- There are two issues that require some care:
  - Each pulse must be inserted in the correct position in the output signal.
    - Recall that the expression for the output signal $s(t)$ contains the terms $p(t - nT)$.
    - The term $p(t - nT)$ reflects pulses delayed by nT.
  - Pulses may overlap.
    - If the duration of a pulse is longer than T, then pulses overlap.
    - Such overlapping pulses are added.
    - This situation is called partial response signaling.

Body of Function `LinearModulation`

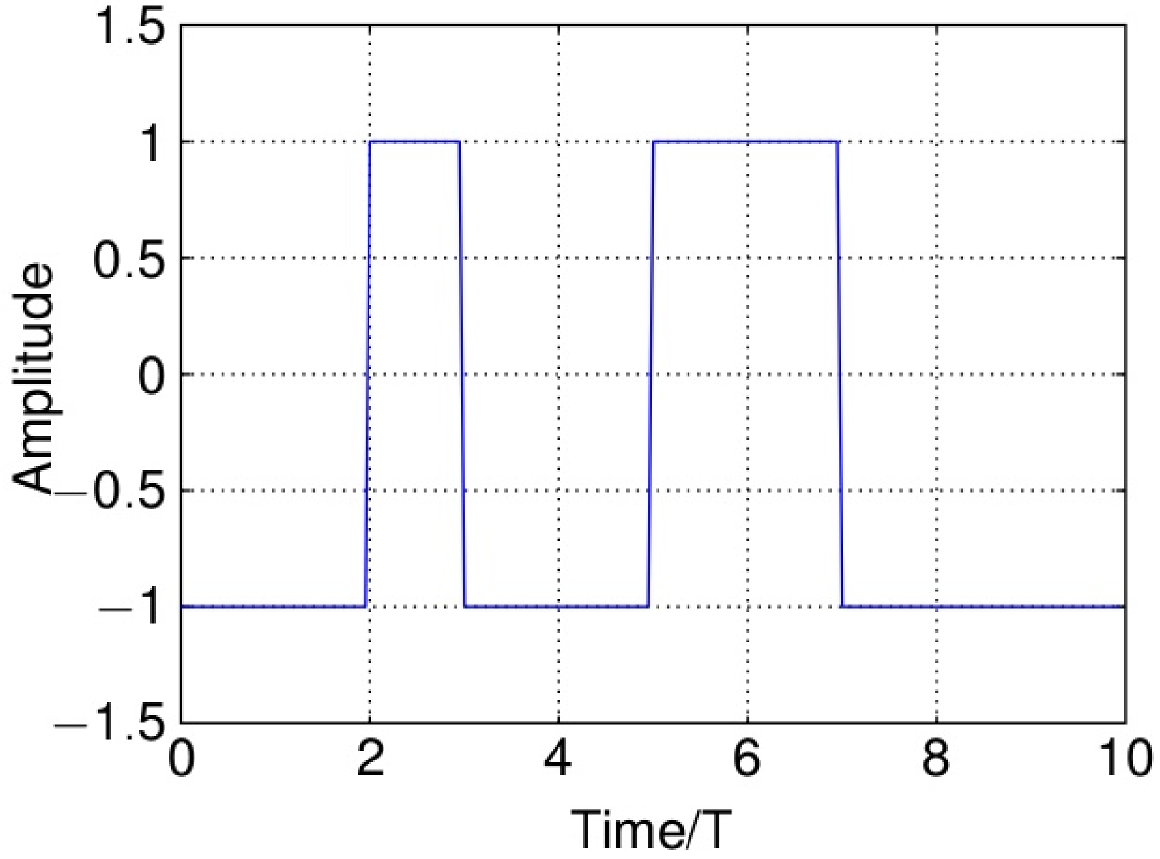
```
% initialize storage for Signal
LenSignal = length(Symbols)*fsT + (length(Pulse))-fsT;
Signal    = zeros( 1, LenSignal );

% loop over symbols and insert corresponding segment into Signal
for kk = 1:length(Symbols)
    ind_start = (kk-1)*fsT + 1;
    ind_end   = (kk-1)*fsT + length(Pulse);

    Signal(ind_start:ind_end) = Signal(ind_start:ind_end) + ...
                                Symbols(kk) * Pulse;
end
```

Testing Function `LinearModulation`

Listing : plot_LinearModRect.m

```
%% Parameters:
fsT       = 20;
Alphabet = [1,-1];
Priors    = 0.5*[1 1];
Pulse     = ones(1,fsT);    % rectangular pulse

%% symbols and Signal using our functions
Symbols = RandomSymbols(10, Alphabet, Priors);
Signal  = LinearModulation(Symbols,Pulse,fsT);
%% plot
tt = (0 : length(Signal)-1 )/fsT;
plot(tt, Signal)
axis([0 length(Signal)/fsT -1.5 1.5])
grid
xlabel('Time/T')
ylabel('Amplitude')
```

Linear Modulation with Rectangular Pulses



Linear Modulation with sinc-Pulses

- More interesting and practical waveforms arise when smoother pulses are used.
- A good example are truncated sinc functions.
  - The sinc function is defined as:

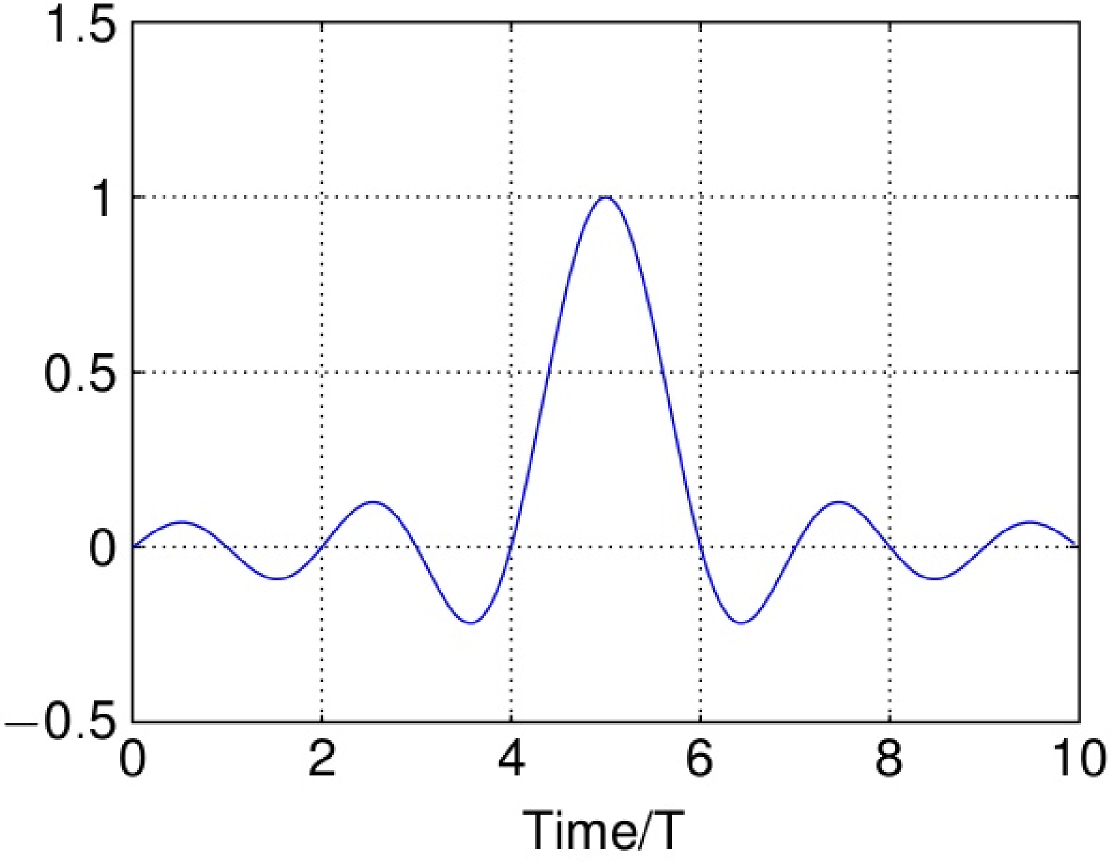
$$\text{sinc}(x) = \frac{\sin(x)}{x}, \text{ with } \text{sinc}(0) = 1.$$

- Specifically, we will use pulses defined by

$$p(t) = \text{sinc}(\pi t/T) = \frac{\sin(\pi t/T)}{\pi t/T};$$

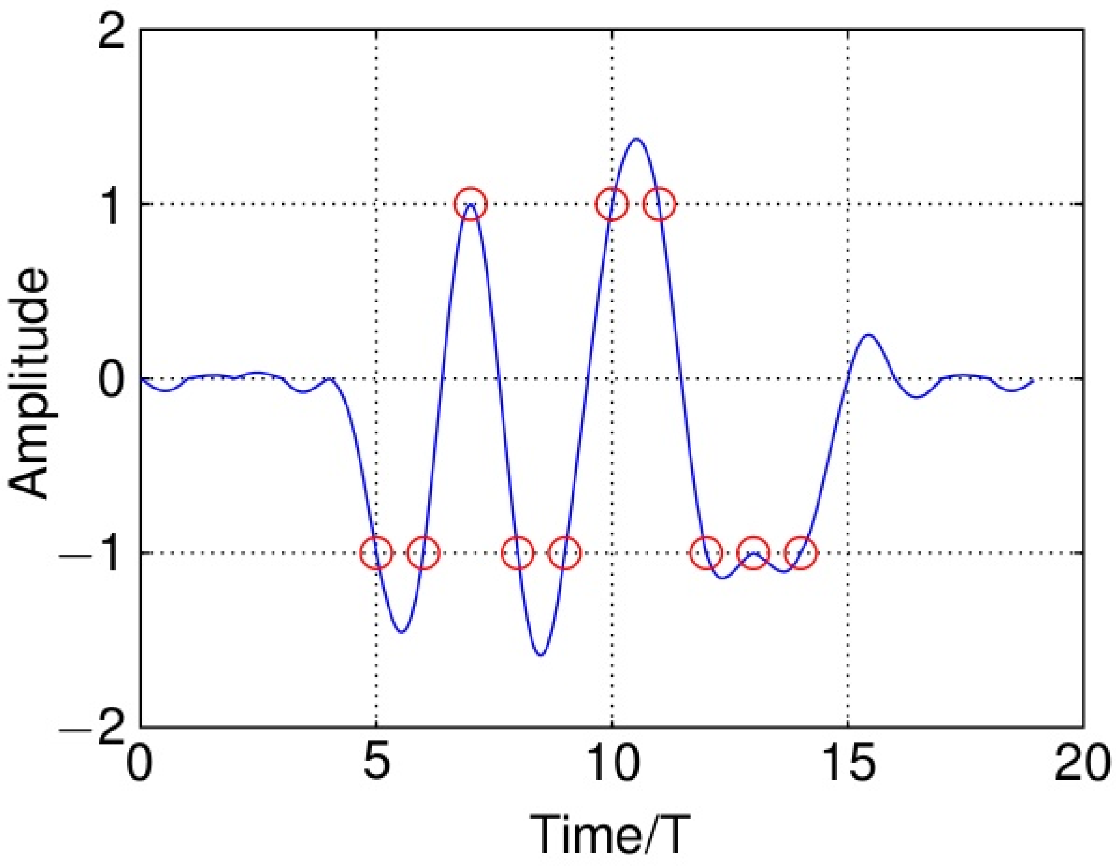
  - pulses are truncated to span L symbol periods, and
  - delayed to be causal.
- Toolbox contains function `Sinc( L, fsT )`.

A Truncated Sinc Pulse



- Pulse is very smooth,
- spans ten symbol periods,
- is zero at location of other symbols.
  - Nyquist pulse.

Linear Modulation with Sinc Pulses

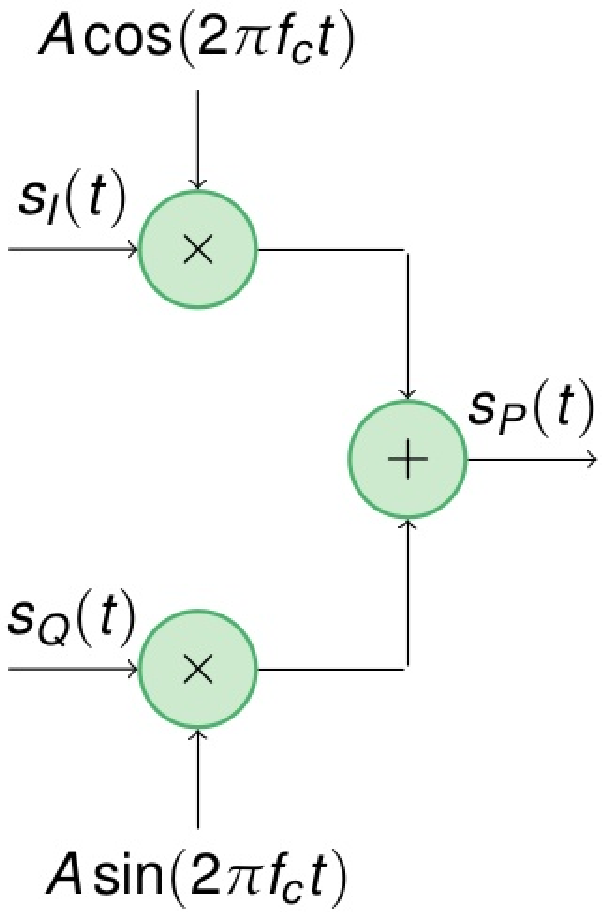


- Resulting waveform is also very smooth; expect good spectral properties.
- Symbols are harder to discern; partial response signaling induces "controlled" ISI.
  - But, there is no ISI at symbol locations.
- Transients at beginning and end.

Passband Signals

- So far, all modulated signals we considered are baseband signals.
  - Baseband signals have frequency spectra concentrated near zero frequency.
- However, for wireless communications passband signals must be used.
  - Passband signals have frequency spectra concentrated around a carrier frequency f_c.
- Baseband signals can be converted to passband signals through up-conversion.
- Passband signals can be converted to baseband signals through down-conversion.

Up-Conversion



- The passband signal $s_P(t)$ is constructed from two (digitally modulated) baseband signals, $s_I(t)$ and $s_Q(t)$.
  - Note that two signals can be carried simultaneously!
  - This is a consequence of $\cos(2\pi f_c t)$ and $\sin(2\pi f_c t)$ being orthogonal.

Baseband Equivalent Signals

- The passband signal $s_P(t)$ can be written as

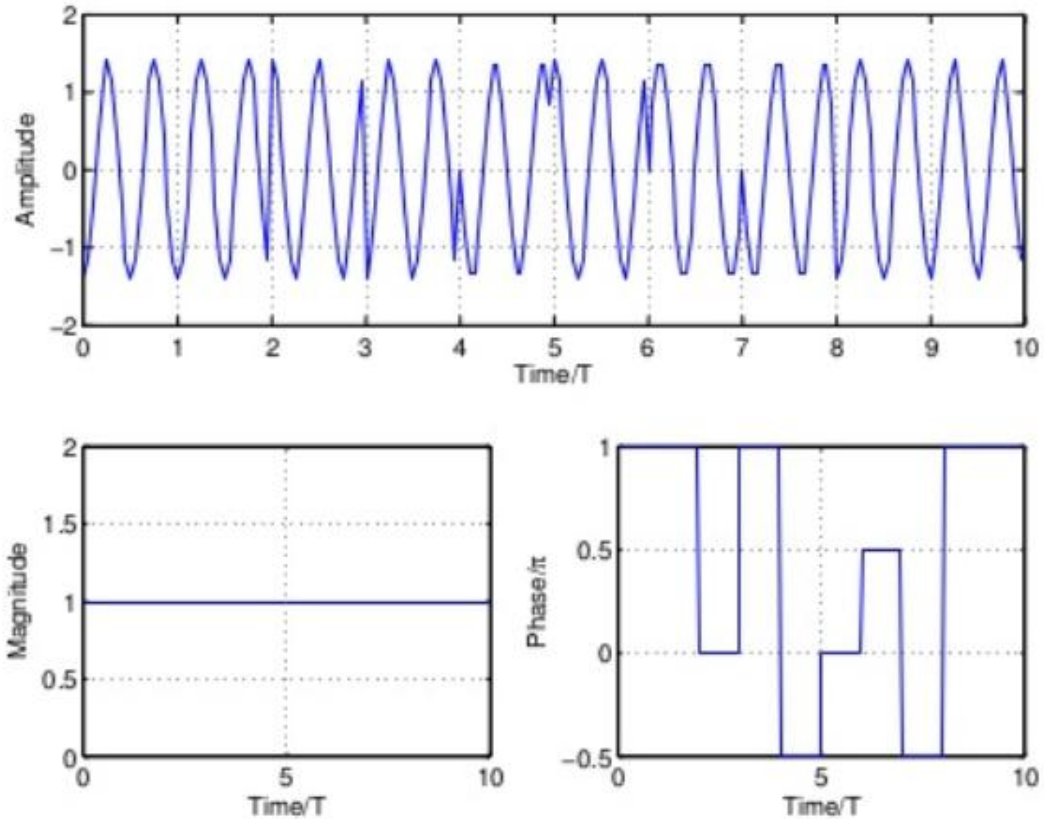
$$s_P(t) = \sqrt{2} \cdot A s_I(t) \cdot \cos(2\pi f_c t) + \sqrt{2} \cdot A s_Q(t) \cdot \sin(2\pi f_c t).$$

- If we define $s(t) = s_I(t) - j \cdot s_Q(t)$, then $s_P(t)$ can also be expressed as

$$s_P(t) = \sqrt{2} \cdot A \cdot \Re\{s(t) \cdot \exp(j2\pi f_c t)\}.$$

- The signal $s(t)$:
  - is called the baseband equivalent or the complex envelope of the passband signal $s_P(t)$.
  - It contains the same information as $s_P(t)$.
  - Note that $s(t)$ is complex-valued.

Illustration: QPSK with $f_c = 2/T$



- Passband signal (top): segments of sinusoids with different phases.
  - Phase changes occur at multiples of T.
- Baseband signal (bottom) is complex valued; magnitude and phase are plotted.
  - Magnitude is constant (rectangular pulses).

MATLAB Code for QPSK Illustration

Listing : plot_LinearModQPSK.m

```
%% Parameters:
fsT      = 20;
L        = 10;
fc       =  2;                % carrier frequency
Alphabet = [1, j, -j, -1];% QPSK
Priors   = 0.25*[1 1 1 1];
Pulse    = ones(1,fsT);    % rectangular pulse

%% symbols and Signal using our functions
Symbols   = RandomSymbols(10, Alphabet, Priors);
Signal    = LinearModulation(Symbols,Pulse,fsT);
%% passband signal
tt = (0 : length(Signal)-1 )/fsT;
Signal_PB = sqrt(2)*real( Signal .* exp(-j*2*pi*fc*tt) );
```

MATLAB Code for QPSK Illustration

Listing : plot_LinearModQPSK.m

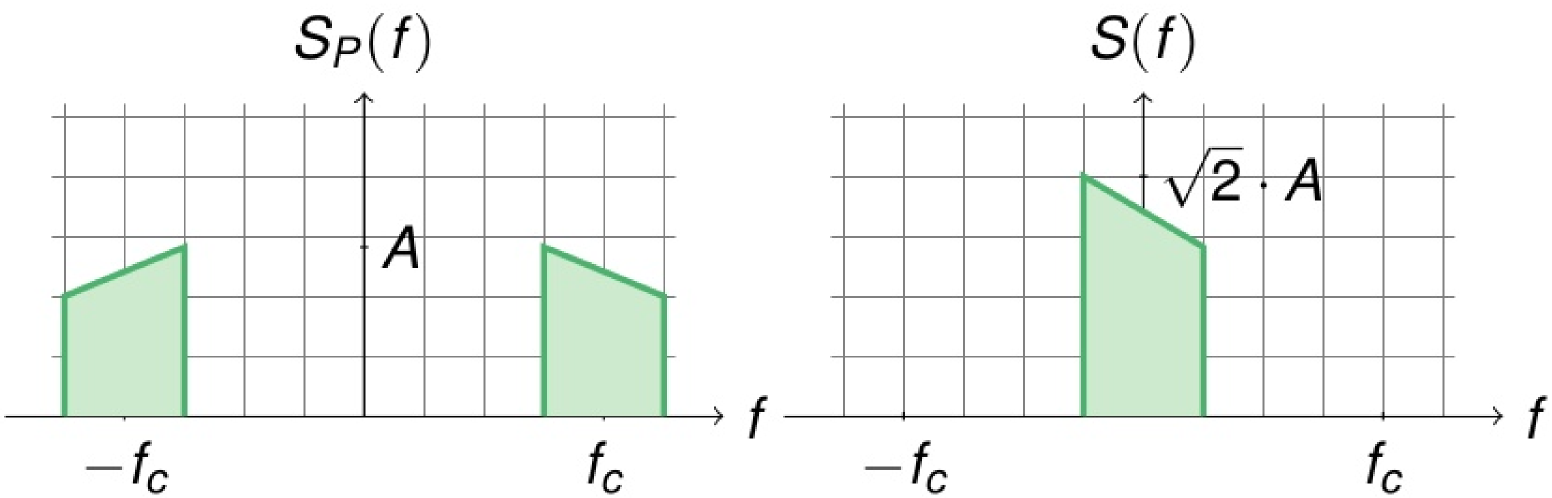
```
subplot(2,1,1)
plot( tt, Signal_PB )
grid
xlabel('Time/T')
ylabel('Amplitude')

subplot(2,2,3)
plot( tt, abs( Signal ) )
grid
xlabel('Time/T')
ylabel('Magnitude')

subplot(2,2,4)
plot( tt, angle( Signal )/pi )
grid
xlabel('Time/T')
ylabel('Phase/\pi')
```

Frequency Domain Perspective

- In the frequency domain:

$$S(f) = \begin{cases} \sqrt{2} \cdot S_P(f + f_c) & \text{for } f + f_c > 0 \\ 0 & \text{else.} \end{cases}$$

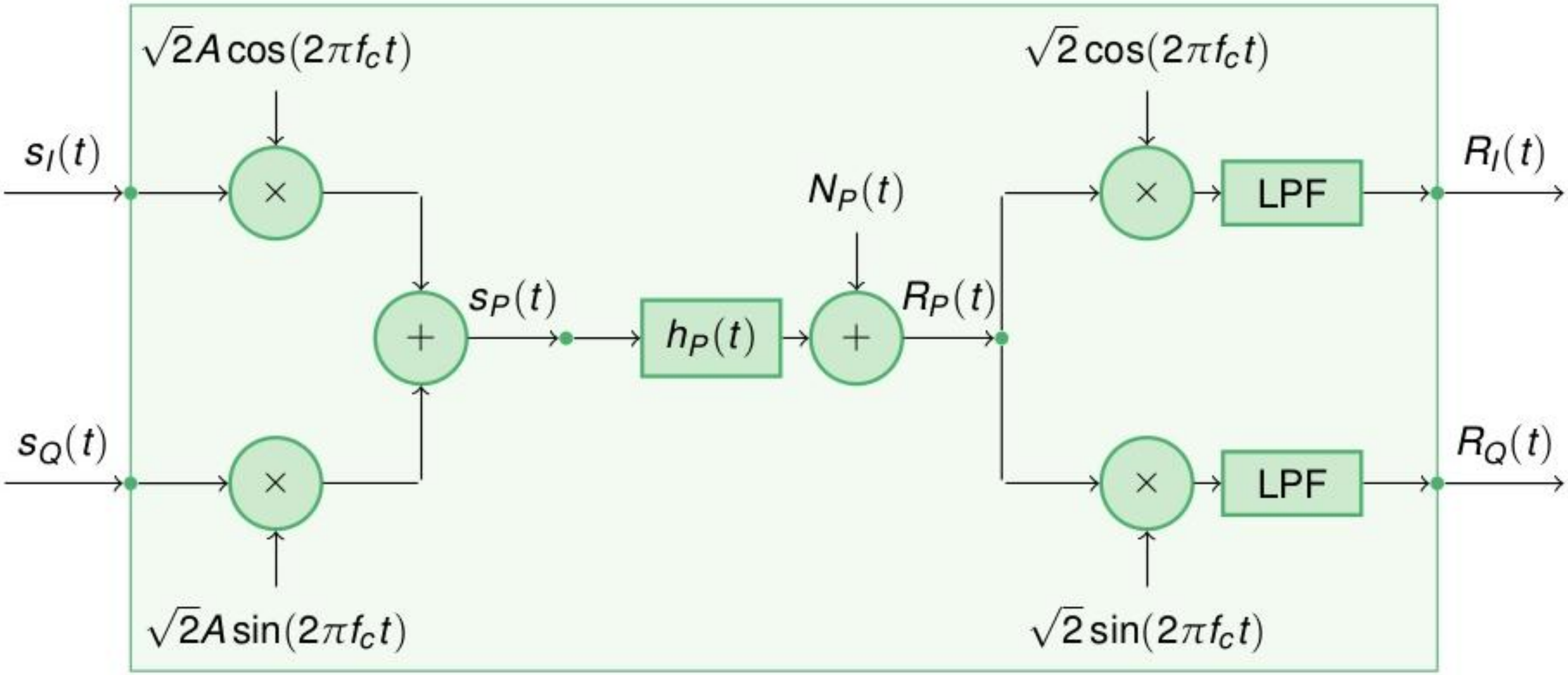
  - Factor $\sqrt{2}$ ensures both signals have the same power.



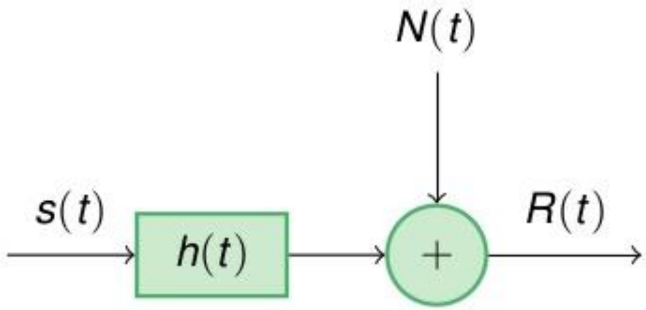
Baseband Equivalent System

- The baseband description of the transmitted signal is very convenient:
  - it is more compact than the passband signal as it does not include the carrier component,
  - while retaining all relevant information.
- However, we are also concerned what happens to the signal as it propagates to the receiver.
  - **Question:** Do baseband techniques extend to other parts of a passband communications system?

Passband System



Baseband Equivalent System



- The passband system can be interpreted as follows to yield an equivalent system that employs only baseband signals:
  - baseband equivalent transmitted signal: $s(t) = s_I(t) - j \cdot s_Q(t)$.
  - baseband equivalent channel with complex valued impulse response: $h(t)$.
  - baseband equivalent received signal: $R(t) = R_I(t) - j \cdot R_Q(t)$.
  - complex valued, additive Gaussian noise: $N(t)$

Baseband Equivalent Channel

- The baseband equivalent channel is defined by the entire shaded box in the block diagram for the passband system (excluding additive noise).
- The relationship between the passband and baseband equivalent channel is

$$h_P(t) = \Re\{h(t) \cdot \exp(j2\pi f_c t)\}$$

in the time domain.

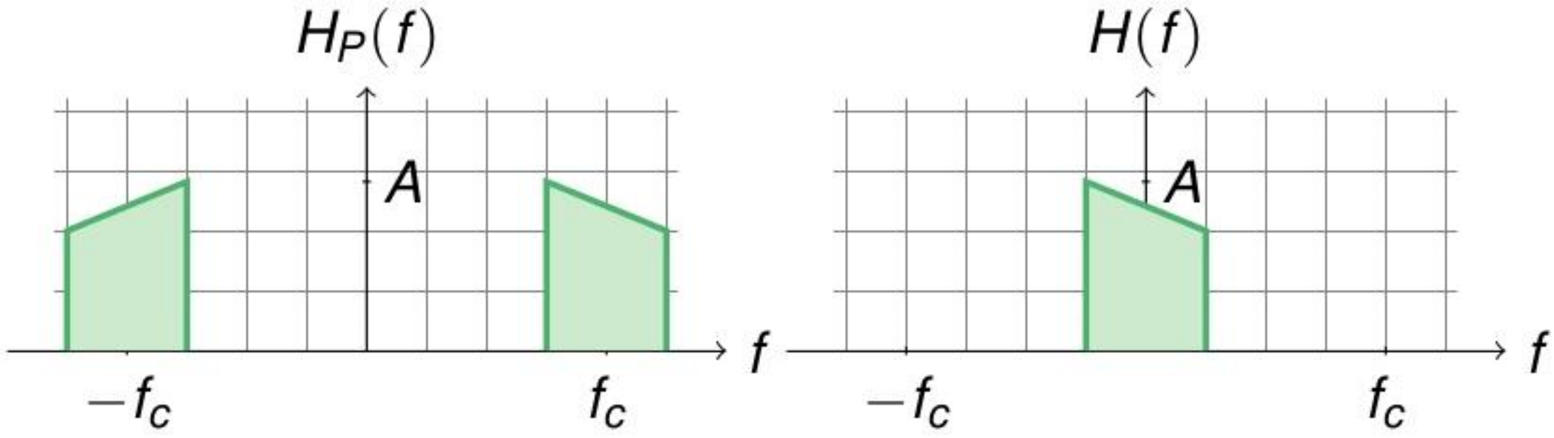
- **Example:**

$$h_P(t) = \sum_k a_k \cdot \delta(t - \tau_k) \Longrightarrow h(t) = \sum_k a_k \cdot e^{-j2\pi f_c \tau_k} \cdot \delta(t - \tau_k).$$

Baseband Equivalent Channel

- In the frequency domain

$$H(f) = \begin{cases} H_P(f + f_c) & \text{for } f + f_c > 0 \\ 0 & \text{else.} \end{cases}$$



Summary

- The baseband equivalent channel is much simpler than the passband model.
  - Up and down conversion are eliminated.
  - Expressions for signals do not contain carrier terms.
- The baseband equivalent signals are easier to represent for simulation.
  - Since they are low-pass signals, they are easily sampled.
- No information is lost when using baseband equivalent signals, instead of passband signals.
- Standard, linear system equations hold:

$$R(t) = s(t) * h(t) + n(t) \text{ and } R(f) = S(f) \cdot H(f) + N(f).$$

- **Conclusion:** Use baseband equivalent signals and systems.