Simulation of Wireless Communication Systems using MATLAB

Dr. B.-P. Paris
Dept. Electrical and Comp. Engineering
George Mason University

Fall 2007

Outline

MATLAB Simulation

Frequency Diversity: Wide-Band Signals

MATLAB Simulation

- **Objective:** Simulate a simple communication system and estimate bit error rate.
- **System Characteristics:**
  - BPSK modulation, $b \in \{1, -1\}$ with equal a priori probabilities,
  - Raised cosine pulses,
  - AWGN channel,
  - oversampled integrate-and-dump receiver front-end,
  - digital matched filter.
- **Measure:** Bit-error rate as a function of E_s/N_0 and oversampling rate.

System to be Simulated

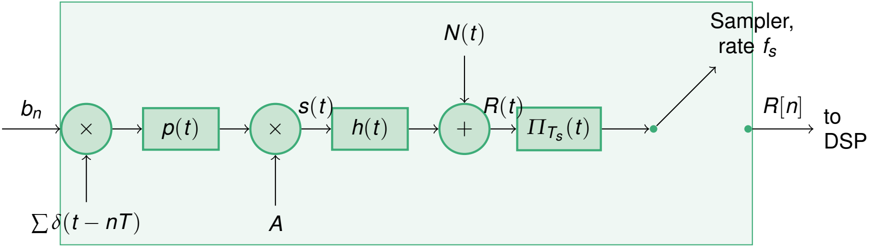


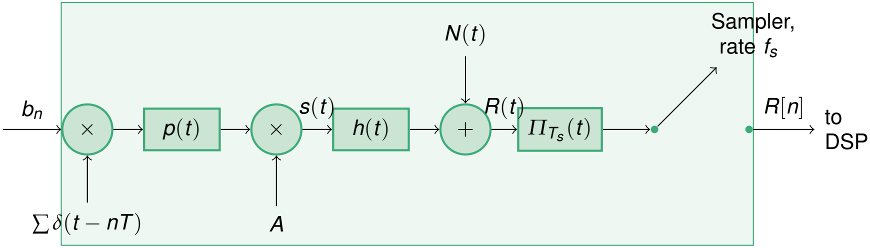
Figure: Baseband Equivalent System to be Simulated.

From Continuous to Discrete Time

- The system in the preceding diagram cannot be simulated immediately.
  - **Main problem:** Most of the signals are continuous-time signals and cannot be represented in MATLAB.
- **Possible Remedies:**
  1. Rely on Sampling Theorem and work with sampled versions of signals.
  2. Consider discrete-time equivalent system.
- The second alternative is preferred and will be pursued below.

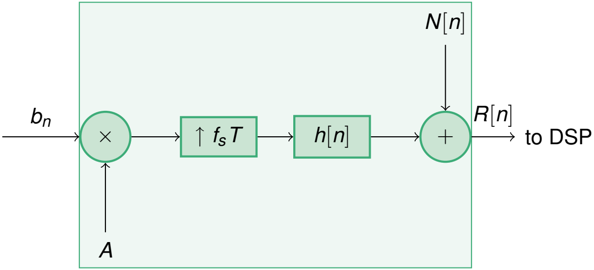
Towards the Discrete-Time Equivalent System

- The shaded portion of the system has a discrete-time input and a discrete-time output.
  - Can be considered as a discrete-time system.
  - **Minor problem:** input and output operate at different rates.



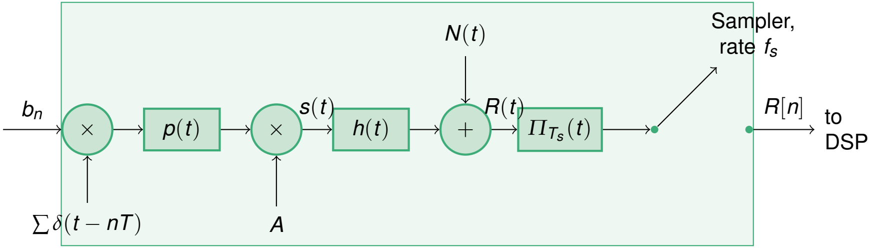
Discrete-Time Equivalent System

- The discrete-time equivalent system
  - is equivalent to the original system, and
  - contains only discrete-time signals and components.
- Input signal is up-sampled by factor $f_s T$ to make input and output rates equal.
  - Insert $f_s T - 1$ zeros between input samples.



Components of Discrete-Time Equivalent System

- **Question:** What is the relationship between the components of the original and discrete-time equivalent system?



Discrete-time Equivalent Impulse Response

- To determine the impulse response $h[n]$ of the discrete-time equivalent system:
  - Set noise signal N_t to zero,
  - set input signal b_n to unit impulse signal $\delta[n]$,
  - output signal is impulse response $h[n]$.
- Procedure yields:

$$h[n] = \frac{1}{T_s} \int_{nT_s}^{(n+1)T_s} p(t) * h(t) \, dt$$

- For high sampling rates ($f_s T \gg 1$), the impulse response is closely approximated by sampling $p(t) * h(t)$:

$$h[n] \approx p(t) * h(t)\big|_{(n+\frac{1}{2})T_s}$$

Discrete-time Equivalent Impulse Response

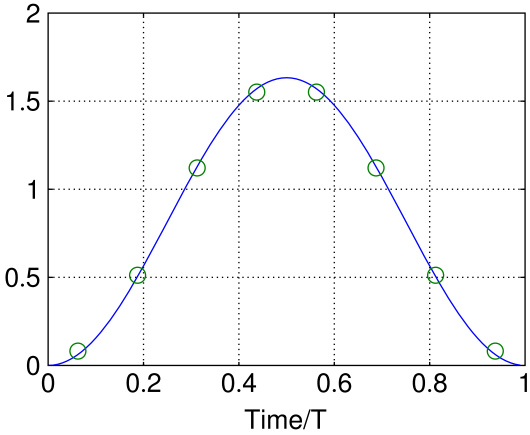


Figure: Discrete-time Equivalent Impulse Response ($f_s T = 8$)

Discrete-Time Equivalent Noise

- To determine the properties of the additive noise $N[n]$ in the discrete-time equivalent system,
  - Set input signal to zero,
  - let continuous-time noise be complex, white, Gaussian with power spectral density N_0,
  - output signal is discrete-time equivalent noise.
- Procedure yields: The noise samples $N[n]$
  - are independent, complex Gaussian random variables, with
  - zero mean, and
  - variance equal to N_0 / T_s.

Received Symbol Energy

- The last entity we will need from the continuous-time system is the received energy per symbol E_s.
  - Note that E_s is controlled by adjusting the gain A at the transmitter.
- To determine E_s,
  - Set noise $N(t)$ to zero,
  - Transmit a single symbol b_n,
  - Compute the energy of the received signal $R(t)$.
- Procedure yields:

$$E_s = \sigma_s^2 \cdot A^2 \int |p(t) * h(t)|^2 \, dt$$

  - Here, σ_s^2 denotes the variance of the source. For BPSK, $\sigma_s^2 = 1$.
  - For the system under consideration, $E_s = A^2 T$.

Simulating Transmission of Symbols

- We are now in position to simulate the transmission of a sequence of symbols.
  - The MATLAB functions previously introduced will be used for that purpose.
- We proceed in three steps:
  1. Establish parameters describing the system,
     - By parameterizing the simulation, other scenarios are easily accommodated.
  2. Simulate discrete-time equivalent system,
  3. Collect statistics from repeated simulation.

Listing : SimpleSetParameters.m

```
% This script sets a structure named Parameters to be used by
% the system simulator.

%% Parameters
% construct structure of parameters to be passed to system simulator
% communications parameters
Parameters.T   = 1/10000;      % symbol period
Parameters.fsT = 8;            % samples per symbol
Parameters.Es  = 1;            % normalize received symbol energy to 1
Parameters.EsOverN0 = 6;       % Signal-to-noise ratio (Es/N0)
Parameters.Alphabet = [1 -1]; % BPSK
Parameters.NSymbols = 1000;    % number of Symbols

% discrete-time equivalent impulse response (raised cosine pulse)
fsT = Parameters.fsT;
tts = ( (0:fsT-1) + 1/2 )/fsT;
Parameters.hh = sqrt(2/3) * ( 1 - cos(2*pi*tts)*sin(pi/fsT)/(pi/fsT));
```

Simulating the Discrete-Time Equivalent System

- The actual system simulation is carried out in MATLAB function `MCSimple` which has the function signature below.
  - The parameters set in the controlling script are passed as inputs.
  - The body of the function simulates the transmission of the signal and subsequent demodulation.
  - The number of incorrect decisions is determined and returned.

```
function [NumErrors, ResultsStruct] = MCSimple( ParametersStruct )
```

Simulating the Discrete-Time Equivalent System

- The simulation of the discrete-time equivalent system uses toolbox functions `RandomSymbols`, `LinearModulation`, and `addNoise`.

```
A        = sqrt(Es/T);      % transmitter gain
N0       = Es/EsOverN0;     % noise PSD (complex noise)
NoiseVar = N0/T*fsT;        % corresponding noise variance N0/Ts
Scale    = A*hh*hh';        % gain through signal chain

%% simulate discrete-time equivalent system
% transmitter and channel via toolbox functions
Symbols = RandomSymbols( NSymbols, Alphabet, Priors );
Signal = A * LinearModulation( Symbols, hh, fsT );
if ( isreal(Signal) )
    Signal = complex(Signal);% ensure  Signal is complex-valued
end
Received = addNoise( Signal, NoiseVar );
```

Digital Matched Filter

- The vector `Received` contains the noisy output samples from the analog front-end.
- In a real system, these samples would be processed by digital hardware to recover the transmitted bits.
  - Such digital hardware may be an ASIC, FPGA, or DSP chip.
- The first function performed there is digital matched filtering.
  - This is a discrete-time implementation of the matched filter discussed before.
  - The matched filter is the best possible processor for enhancing the signal-to-noise ratio of the received signal.

Digital Matched Filter

- In our simulator, the vector `Received` is passed through a discrete-time matched filter and down-sampled to the symbol rate.
  - The impulse response of the matched filter is the conjugate complex of the time-reversed, discrete-time channel response $h[n]$.

$$\xrightarrow{R[n]} \boxed{h^*[-n]} \rightarrow \boxed{\downarrow f_s T} \rightarrow \boxed{\text{Slicer}} \xrightarrow{\hat{b}_n}$$

MATLAB Code for Digital Matched Filter

- The signature line for the MATLAB function implementing the matched filter is:

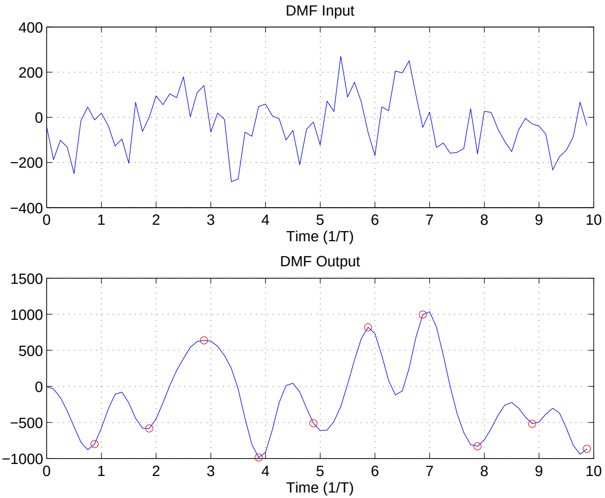
```
function MFOut = DMF( Received, Pulse, fsT )
```

- The body of the function is a direct implementation of the structure in the block diagram above.

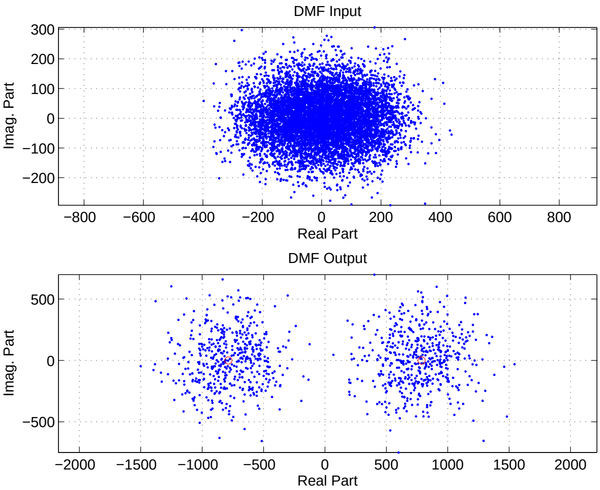
```
% convolve received signal with conjugate complex of
% time-reversed pulse (matched filter)
Temp = conv( Received, conj( fliplr(Pulse) ) );

% down sample, at the end of each pulse period
MFOut = Temp( length(Pulse) : fsT : end );
```

DMF Input and Output Signal



IQ-Scatter Plot of DMF Input and Output



Slicer

- The final operation to be performed by the receiver is deciding which symbol was transmitted.
  - This function is performed by the slicer.
- The operation of the slicer is best understood in terms of the IQ-scatter plot on the previous slide.
  - The red circles in the plot indicate the noise-free signal locations for each of the possibly transmitted signals.
  - For each output from the matched filter, the slicer determines the nearest noise-free signal location.
  - The decision is made in favor of the symbol that corresponds to the noise-free signal nearest the matched filter output.
- Some adjustments to the above procedure are needed when symbols are not equally likely.

MATLAB Function SimpleSlicer

- The procedure above is implemented in a function with signature

```
function [Decisions, MSE] = SimpleSlicer( MFOut, Alphabet, Scale )
```

```
%% Loop over symbols to find symbol closest to MF output
for kk = 1:length( Alphabet )
    % noise-free signal location
    NoisefreeSig = Scale*Alphabet(kk);
    % Euclidean distance between each observation and constellation po
    Dist         = abs( MFOut - NoisefreeSig );
    % find locations for which distance is smaller than previous best
    ChangedDec   = ( Dist < MinDist );

    % store new min distances and update decisions
    MinDist( ChangedDec)     = Dist( ChangedDec );
    Decisions( ChangedDec ) = Alphabet(kk);
end
```

Entire System

- The addition of functions for the digital matched filter completes the simulator for the communication system.
- The functionality of the simulator is encapsulated in a function with signature

```
function [NumErrors, ResultsStruct] = MCSimple( ParametersStruct )
```

  - The function simulates the transmission of a sequence of symbols and determines how many symbol errors occurred.
  - The operation of the simulator is controlled via the parameters passed in the input structure.
  - The body of the function is shown on the next slide; it consists mainly of calls to functions in our toolbox.

Listing : MCSimple.m

```
%% simulate discrete-time equivalent system
% transmitter and channel via toolbox functions
Symbols = RandomSymbols( NSymbols, Alphabet, Priors );
Signal = A * LinearModulation( Symbols, hh, fsT );
if ( isreal(Signal) )
    Signal = complex(Signal);% ensure  Signal is complex-valued
end
Received = addNoise( Signal, NoiseVar );

% digital matched filter and slicer
MFOut     = DMF( Received, hh, fsT );
Decisions = SimpleSlicer( MFOut(1:NSymbols), Alphabet,  Scale );

%% Count errors
NumErrors = sum( Decisions ~= Symbols );
```

Monte Carlo Simulation

- The system simulator will be the work horse of the Monte Carlo simulation.
- The objective of the Monte Carlo simulation is to estimate the symbol error rate our system can achieve.
- The idea behind a Monte Carlo simulation is simple:
  - Simulate the system repeatedly,
  - for each simulation count the number of transmitted symbols and symbol errors,
  - estimate the symbol error rate as the ratio of the total number of observed errors and the total number of transmitted bits.

Monte Carlo Simulation

- The above suggests a relatively simple structure for a Monte Carlo simulator.
- Inside a programming loop:
  - perform a system simulation, and
  - accumulate counts for the quantities of interest

```
while ( ~Done )
    NumErrors(kk)  = NumErrors(kk)  + MCSimple( Parameters );
    NumSymbols(kk) = NumSymbols(kk) + Parameters.NSymbols;

    % compute Stop condition
    Done = NumErrors(kk) > MinErrors || NumSymbols(kk) > MaxSy
end
```

Confidence Intervals

- **Question:** How many times should the loop be executed?
- **Answer:** It depends
  - on the desired level of accuracy (confidence), and
  - (most importantly) on the symbol error rate.
- **Confidence Intervals:**
  - Assume we form an estimate of the symbol error rate P_e as described above.
  - Then, the true error rate $\hat{P}_e$ is (hopefully) close to our estimate.
  - Put differently, we would like to be reasonably sure that the absolute difference $|\hat{P}_e - P_e|$ is small.

Confidence Intervals

- More specifically, we want a high probability p_c (e.g., p_c =95%) that $|\hat{P}_e - P_e| < s_c$.
  - The parameter s_c is called the confidence interval;
  - it depends on the confidence level p_c, the error probability P_e, and the number of transmitted symbols N.
- It can be shown, that

$$s_c = z_c \cdot \sqrt{\frac{P_e(1-P_e)}{N}},$$

where z_c depends on the confidence level p_c.
  - Specifically: $Q(z_c) = (1-p_c)/2$.
  - Example: for p_c =95%, $z_c = 1.96$.
- **Question:** How is the number of simulations determined from the above considerations?

Choosing the Number of Simulations

- For a Monte Carlo simulation, a stop criterion can be formulated from
  - a desired confidence level p_c (and, thus, z_c)
  - an acceptable confidence interval s_c,
  - the error rate P_e.
- Solving the equation for the confidence interval for N, we obtain

$$N = P_e \cdot (1 - P_e) \cdot (z_c / s_c)^2.$$

  - A Monte Carlo simulation can be stopped after simulating N transmissions.
  - **Example:** For p_c =95%, $P_e = 10^{-3}$, and $s_c = 10^{-4}$, we find $N \approx 400{,}000$.

A Better Stop-Criterion

- When simulating communications systems, the error rate is often very small.
- Then, it is desirable to specify the confidence interval as a fraction of the error rate.
  - The confidence interval has the form $s_c = \alpha_c \cdot P_e$ (e.g., $\alpha_c = 0.1$ for a 10% acceptable estimation error).
- Inserting into the expression for N and rearranging terms,

$$P_e \cdot N = (1 - P_e) \cdot (z_c/\alpha_c)^2 \approx (z_c/\alpha_c)^2.$$

  - Recognize that $P_e \cdot N$ is the expected number of errors!
  - **Interpretation:** Stop when the number of errors reaches $(z_c/\alpha_c)^2$.
- **Rule of thumb:** Simulate until 400 errors are found (p_c =95%, α =10%).

Listing : MCSimpleDriver.m

```
% comms parameters delegated to script SimpleSetParameters
SimpleSetParameters;

% simulation parameters
EsOverN0dB = 0:0.5:9; % vary SNR between 0 and 9dB
MaxSymbols = 1e6;     % simulate at most 1000000 symbols

% desired confidence level an size of confidence interval
ConfLevel   = 0.95;
ZValue      = Qinv( ( 1-ConfLevel )/2 );
ConfIntSize = 0.1;  % confidence interval size is 10% of estimate
% For the desired accuracy, we need to find this many errors.
MinErrors  = ( ZValue/ConfIntSize )^2;

Verbose    = true;  % control progress output

%% simulation loops
% initialize loop variables
NumErrors  = zeros( size( EsOverN0dB ) );
NumSymbols = zeros( size( EsOverN0dB ) );
```

Listing : MCSimpleDriver.m

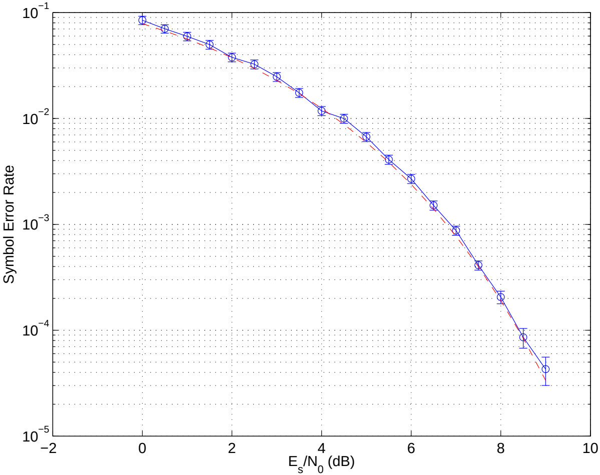
```
for kk = 1:length( EsOverN0dB )
    % set Es/N0 for this iteration
    Parameters.EsOverN0 = dB2lin( EsOverN0dB(kk) );
    % reset stop condition for inner loop
    Done = false;

    % progress output
    if (Verbose)
       disp( sprintf( 'Es/N0:␣%0.3g␣dB',  EsOverN0dB(kk) ) );
    end

    % inner loop iterates until enough errors have been found
    while ( ~Done )
        NumErrors(kk)  = NumErrors(kk)  + MCSimple( Parameters );
        NumSymbols(kk) = NumSymbols(kk) + Parameters.NSymbols;

        % compute Stop condition
        Done = NumErrors(kk) > MinErrors || NumSymbols(kk) > MaxSymbol
    end
```

Simulation Results



Summary

- Introduced discrete-time equivalent systems suitable for simulation in MATLAB.
  - Relationship between original, continuous-time system and discrete-time equivalent was established.
- Digital post-processing: digital matched filter and slicer.
- Monte Carlo simulation of a simple communication system was performed.
  - Close attention was paid to the accuracy of simulation results via confidence levels and intervals.
  - Derived simple rule of thumb for stop-criterion.

Where we are ...

- Laid out a structure for describing and analyzing communication systems in general and wireless systems in particular.
- Saw a lot of MATLAB examples for modeling diverse aspects of such systems.
- Conducted a simulation to estimate the error rate of a communication system and compared to theoretical results.
- To do: consider selected aspects of wireless communication systems in more detail, including:
  - modulation and bandwidth,
  - wireless channels,
  - advanced techniques for wireless communications.

Outline

MATLAB Simulation

Frequency Diversity: Wide-Band Signals

Frequency Diversity through Wide-Band Signals

- We have seen above that narrow-band systems do not have built-in diversity.
  - Narrow-band signals are susceptible to have the entire signal affected by a deep fade.
- In contrast, wide-band signals cover a bandwidth that is wider than the coherence bandwidth.
  - **Benefit:** Only portions of the transmitted signal will be affected by deep fades (frequency-selective fading).
  - **Disadvantage:** Short symbol duration induces ISI; receiver is more complex.
- The benefits, far outweigh the disadvantages and wide-band signaling is used in most modern wireless systems.

Illustration: Built-in Diversity of Wide-band Signals

- We illustrate that wide-band signals do provide diversity by means of a simple thought experiments.
- **Thought experiment:**
  - Recall that in discrete time a multi-path channel can be modeled by an FIR filter.
    - Assume filter operates at symbol rate T_s.
    - The delay spread determines the number of taps L.
  - Our hypothetical system transmits one information symbol in every L-th symbol period and is silent in between.
  - At the receiver, each transmission will produce L non-zero observations.
    - This is due to multi-path.
    - Observation from consecutive symbols don't overlap (no ISI)
  - Thus, for each symbol we have L independent observations, i.e., we have L-fold diversity.

Illustration: Built-in Diversity of Wide-band Signals

- We will demonstrate shortly that it is not necessary to leave gaps in the transmissions.
  - The point was merely to eliminate ISI.
- Two insights from the thought experiment:
  - Wide-band signals provide built-in diversity.
    - The receiver gets to look at multiple versions of the transmitted signal.
  - The order of diversity depends on the ratio of delay spread and symbol duration.
    - Equivalently, on the ratio of signal bandwidth and coherence bandwidth.
- We are looking for receivers that both exploit the built-in diversity and remove ISI.
  - Such receiver elements are called equalizers.

Equalization

- Equalization is obviously a very important and well researched problem.
- Equalizers can be broadly classified into three categories:
  1. **Linear Equalizers:** use an inverse filter to compensate for the variations in the frequency response.
     - Simple, but not very effective with deep fades.
  2. **Decision Feedback Equalizers:** attempt to reconstruct ISI from past symbol decisions.
     - Simple, but have potential for error propagation.
  3. **ML Sequence Estimation:** find the most likely sequence of symbols given the received signal.
     - Most powerful and robust, but computationally complex.

Maximum Likelihood Sequence Estimation

- Maximum Likelihood Sequence Estimation provides the most powerful equalizers.
- Unfortunately, the computational complexity grows exponentially with the ratio of delay spread and symbol duration.
  - I.e., with the number of taps in the discrete-time equivalent FIR channel.

Maximum Likelihood Sequence Estimation

- The principle behind MLSE is simple.
  - Given a received sequence of samples $R[n]$, e.g., matched filter outputs, and
  - a model for the output of the multi-path channel: $\hat{r}[n] = s[n] * h[n]$, where
    - $s[n]$ denotes the symbol sequence, and
    - $h[n]$ denotes the discrete-time channel impulse response, i.e., the channel taps.
  - Find the sequence of information symbol $s[n]$ that minimizes

$$D^2 = \sum_{n}^{N} |r[n] - s[n] * h[n]|^2.$$

Maximum Likelihood Sequence Estimation

- The criterion

$$D^2 = \sum_n^N |r[n] - s[n] * h[n]|^2.$$

  - performs diversity combining (via $s[n] * h[n]$), and
  - removes ISI.
- The minimization of the above metric is difficult because it is a discrete optimization problem.
  - The symbols $s[n]$ are from a discrete alphabet.
- A computationally efficient algorithm exists to solve the minimization problem:
  - The Viterbi Algorithm.
  - The toolbox contains an implementation of the Viterbi Algorithm in function `va`.

MATLAB Simulation

- A Monte Carlo simulation of a wide-band signal with an equalizer is conducted
  - to illustrate that diversity gains are possible, and
  - to measure the symbol error rate.
- As before, the Monte Carlo simulation is broken into
  - set simulation parameter (script `VASetParameters`),
  - simulation control (script `MCVADriver`), and
  - system simulation (function `MCVA`).

MATLAB Simulation: System Parameters

Listing : VASetParameters.m

```
Parameters.T    = 1/1e6;          % symbol period
Parameters.fsT = 8;              % samples per symbol
Parameters.Es   = 1;             % normalize received symbol energy to 1
Parameters.EsOverN0 = 6;         % Signal-to-noise ratio (Es/N0)
Parameters.Alphabet = [1 -1];  % BPSK
Parameters.NSymbols = 500;      % number of Symbols per frame

Parameters.TrainLoc     = floor(Parameters.NSymbols/2); % location of t
Parameters.TrainLength = 40;
Parameters.TrainingSeq = RandomSymbols( Parameters.TrainLength, ...
                                         Parameters.Alphabet, [0.5 0.5]

% channel
Parameters.ChannelParams = tux(); % channel model
Parameters.fd              = 3;      % Doppler
Parameters.L               = 6;      % channel order
```

MATLAB Simulation

- The first step in the system simulation is the simulation of the transmitter functionality.
  - This is identical to the narrow-band case, except that the baud rate is 1 MHz and 500 symbols are transmitted per frame.
  - There are 40 training symbols.

Listing : MCVA.m

```
% transmitter and channel via toolbox functions
InfoSymbols = RandomSymbols( NSymbols, Alphabet, Priors );
% insert training sequence
Symbols = [ InfoSymbols(1:TrainLoc) TrainingSeq ...
    InfoSymbols(TrainLoc+1:end)];
% linear modulation
Signal = A * LinearModulation( Symbols, hh, fsT );
```

MATLAB Simulation

- The channel is simulated without spatial diversity.
  - To focus on the frequency diversity gained by wide-band signaling.
- The channel simulation invokes the time-varying multi-path simulator and the AWGN function.

```
% time-varying multi-path channels and additive noise
Received = SimulateCOSTChannel( Signal, ChannelParams, fs);
Received = addNoise( Received, NoiseVar );
```

MATLAB Simulation

- The receiver proceeds as follows:
    - Digital matched filtering with the pulse shape; followed by down-sampling to 2 samples per symbol.
    - Estimation of the coefficients of the FIR channel model.
    - Equalization with the Viterbi algorithm; followed by removal of the training sequence.

```
MFOut      = DMF( Received, hh, fsT/2 );

% channel estimation
MFOutTraining = MFOut( 2*TrainLoc+1 : 2*(TrainLoc+TrainLength) );
ChannelEst = EstChannel( MFOutTraining, TrainingSeq, L, 2);

% VA over MFOut using ChannelEst
Decisions = va( MFOut, ChannelEst, Alphabet, 2);
% strip training sequence and possible extra symbols
Decisions( TrainLoc+1 : TrainLoc+TrainLength ) = [ ];
```

Channel Estimation

- **Channel Estimate:**

$$\hat{\mathbf{h}} = (\mathbf{S}'\mathbf{S})^{-1} \cdot \mathbf{S}'\mathbf{r},$$

  where
  - **S** is a Toeplitz matrix constructed from the training sequence, and
  - **r** is the corresponding received signal.

```
TrainingSPS                = zeros(1, length(Received) );
TrainingSPS(1:SpS:end) = Training;

% make into a Toepliz matrix, such that T*h is convolution
TrainMatrix = toeplitz( TrainingSPS, [Training(1) zeros(1, Order-1)]);

ChannelEst = Received * conj( TrainMatrix) * ...
    inv(TrainMatrix' * TrainMatrix);
```

Simulated Symbol Error Rate with MLSE Equalizer

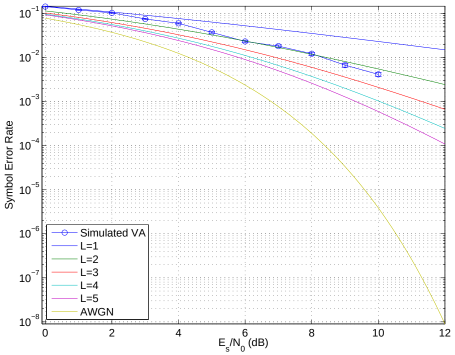


Figure: Symbol Error Rate with Viterbi Equalizer over Multi-path Fading Channel; Rayleigh channels with transmitter diversity shown for comparison. Baud rate 1MHz, Delay spread $\approx 2\mu$s.

Conclusions

- The simulation indicates that the wide-band system with equalizer achieves a diversity gain similar to a system with transmitter diversity of order 2.
  - The ratio of delay spread to symbol rate is 2.
  - comparison to systems with transmitter diversity is appropriate as the total average power in the channel taps is normalized to 1.
  - Performance at very low SNR suffers, probably, from inaccurate estimates.
- Higher gains can be achieved by increasing bandwidth.
  - This incurs more complexity in the equalizer, and
  - potential problems due to a larger number of channel coefficients to be estimated.
- Alternatively, this technique can be combined with additional diversity techniques (e.g., spatial diversity).

More Ways to Create Diversity

- A quick look at three additional ways to create and exploit diversity.
  1. Time diversity.
  2. Frequency Diversity through OFDM.
  3. Multi-antenna systems (MIMO)

Time Diversity

- **Time diversity:** is created by sending information multiple times in different frames.
  - This is often done through *coding* and *interleaving*.
  - This technique relies on the channel to change sufficiently between transmissions.
    - The channel's coherence time should be much smaller than the time between transmissions.
  - If this condition cannot be met (e.g., for slow-moving mobiles), *frequency hopping* can be used to ensure that the channel changes sufficiently.
- The diversity gain is (at most) equal to the number of time-slots used for repeating information.
- Time diversity can be easily combined with frequency diversity as discussed above.
  - The combined diversity gain is the product of the individual diversity gains.

OFDM

- OFDM has received a lot of interest recently.
- OFDM can elegantly combine the benefits of narrow-band signals and wide-band signals.
  - Like for narrow-band signaling, an equalizer is not required; merely the gain for each subcarier is needed.
    - Very low-complexity receivers.
  - OFDM signals are inherently wide-band; frequency diversity is easily achieved by repeating information (really coding and interleaving) on widely separated subcarriers.
    - Bandwidth is not limited by complexity of equalizer;
    - High signal bandwidth to coherence bandwidth is possible; high diversity.

MIMO

- We have already seen that multiple antennas at the receiver can provide both diversity and array gain.
  - The diversity gain ensures that the likelihood that there is no good channel from transmitter to receiver is small.
  - The array gain exploits the benefits from observing the transmitted energy multiple times.
- If the system is equipped with multiple transmitter antennas, then the number of channels equals the product of the number of antennas.
  - Very high diversity.
- Recently, it has been found that multiple streams can be transmitted in parallel to achieve high data rates.
  - Multiplexing gain
- The combination of multi-antenna techniques and OFDM appears particularly promising.

Summary

- A close look at the detrimental effect of typical wireless channels.
  - Narrow-band signals without diversity suffer poor performance (Rayleigh fading).
  - Simulated narrow-band system.
- To remedy this problem, diversity is required.
  - Analyzed systems with antenna diversity at the receiver.
  - Verified analysis through simulation.
- Frequency diversity and equalization.
  - Introduced MLSE and the Viterbi algorithm for equalizing wide-band signals in multi-path channels.
  - Simulated system and verified diversity.
- A brief look at other diversity techniques.