COMM.SYS.300 COMMUNICATION THEORY

Project Work:
Experimenting an Elementary Single-Carrier M-QAM-based Digital Communication Chain

1 Assumed System Model and Parameters

1.1 System model

In this project work, we experiment a single-carrier quadrature amplitude modulation (QAM)-based digital communication system including the basic transmitter (TX) and receiver (RX) processing principles, as well as the impacts of a frequency-selective multipath channel and RX frequency oscillator phase noise. The basic system model is shown in Fig. 1 below, where baseband equivalent approach is taken (i.e. I/Q modulation and I/Q demodulation are not explicitly considered).

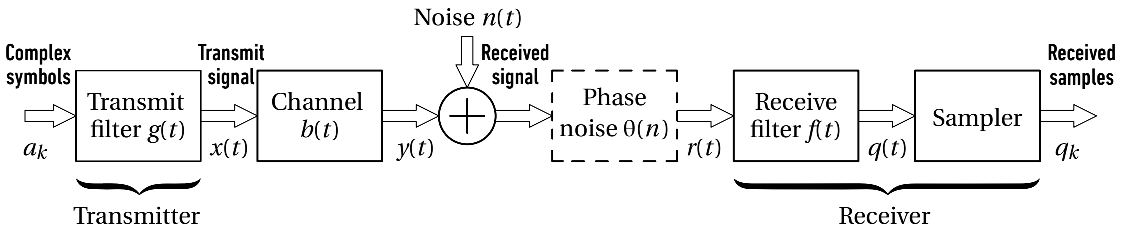

Figure 1: Baseband equivalent system structure with transmit pulse-shape filtering, noisy multipath channel, receiver filtering, and non-ideal receiver oscillator model with phase noise included. Channel equalization and symbol detection functionalities not shown/considered at RX side.

As you can observe, when it comes to the RX side, only the basic processing (RX filtering and sampling) are considered, while a true receiver would have channel equalization and detector functionalities included as well. Those aspects are excluded here, for simplicity, and thus we only experiment the received signal quality using the unequalized received samples and try to understand how a frequency-selective multipath channel and noise impact them, particularly from the TX signal spectrum point of view. We also experiment the RX signal characteristics and the impact of the RX oscillator phase noise.

1.2 Basic system parameters

- First let's define some general system parameters as follows:

```
clear all % Remove all variables from memory
R = 10e6; % This defines the symbol rate in Hz or [sym/sec]
T = 1/R; % Corresponding symbol interval [s]
r = 4; % Oversampling factor (r samples per pulse)
```

- Based on above we can define sampling frequency and sampling time interval as

```
Fs = r/T; % Sampling frequency
Ts = 1/Fs; % Sampling time interval
```

2 Basic Transmitter Processing

2.1 Generation of QAM symbols

- First we define the number of symbols to be transmitted:

```
N_symbols = 10000; % Number of symbols to be transmitted
```

- Then, decide the modulation order M, let's use 16-QAM for now

```
M = 16; % Modulation order
```

- Then generate a random sequence of symbols
  - First, create M-QAM constellation
  - Then draw the specified number (`N_symbols`) of random symbols from the constellation
  - Plot the transmitted symbols in complex plane, for visualization

```
% Here qam_axis presents the symbol values per real/imaginary axis.
% Below code is valid for values of M of the form 4, 16, 64, 256, 1024, ...
qam_axis = -sqrt(M)+1:2:sqrt(M)-1;

% For example, the above results in
% qam_axis = [-1 1]; % for QPSK
% qam_axis = [-3 -1 1 3]; % for 16-QAM
% qam_axis = [-7 -5 -3 -1 1 3 5 7]; % for 64-QAM
% Generation of the complex constellation:
alphabet = bsxfun(@plus,qam_axis',1j*qam_axis); % see help bsxfun

% This is equivalent to (alternative way to do the same thing)
% alphabet = repmat(qam_axis', 1, sqrt(M)) + repmat(1j*qam_axis, sqrt(M), 1);

alphabet = alphabet(:).'; % Finally represent alphabet symbols as a row vector

% Then draw a random vector of numbers between 1...M
symbol_ind = randi(length(alphabet),1,N_symbols);

% Then "index" the alphabet vector using the above sequence to effectively
% draw random symbols from the constellation
symbols = alphabet(symbol_ind); % Random symbol sequence to be transmitted

% Plot the symbols
figure(1)
plot(symbols,'ro', 'MarkerFaceColor','r')
axis equal
xlabel('Real part')
ylabel('Imaginary part')
title('Transmitted symbols')
grid on

% You may wish to use the command axis([XMIN XMAX YMIN YMAX]) to
% better adjust the horizontal and vertical axis scaling. Alternative
% way is to use commands xlim and ylim
```

2.2 Transmitter pulse shaping filtering

By following the TX structure in Fig. 1, we generate a continuous-time baseband QAM signal by adopting upsampling and root-raised cosine (RRC) filtering.

- First define some basic parameters:

```
N_symbols_per_pulse = 40; % Duration of TX pulse-shape filter in symbols
alpha = 0.20; % Roll-off factor (excess bandwidth)

% Comment: we are using quite long pulses (longer than what is done in
% practice commonly. This is because Matlab's rcosdesign function that we use
% below is not very well optimized to design RRC pulses
```

- Implement then the transmit RRC filter and plot the pulse shape

```
% Filter generation
gt = rcosdesign(alpha,N_symbols_per_pulse,r,'sqrt'); % see help rcosdesign
```

```
% Plot the pulse shape of the transmit/receive filter
figure(2)
plot(-N_symbols_per_pulse*r/2*Ts:Ts:N_symbols_per_pulse*r/2*Ts,gt,'b')
xlabel('Time [s]')
ylabel('Value')
title('Transmit RRC filter (pulse shape)')
grid on
```

- Filter the transmitted symbol sequence. We have to first upsample the symbol sequence rate to match with sampling rate of the filter/pulse:

```
% Zero vector initilized for up-sampled symbol sequence
symbols_upsampled = zeros(size(1:r*N_symbols));

% Symbol insertion
symbols_upsampled(1:r:r*N_symbols) = symbols;

% Now the up-sampled sequence looks like {a1 0 0... a2 0 0... a3 0 0...}

% Alternatively, the upsampling by a factor of r can be carried out using
% the ready "upsample" function as follows:
% symbols_upsampled = upsample(symbols, r);

xt = filter(gt,1,symbols_upsampled); % Transmit pulse-shape filtering
xt = xt(1+(length(gt)-1)/2:end); % Correct for the filter delay/transient
```

- Plot the spectrum of the generated signal

```
NFFT = 16384; % FFT size for spectrum analysis, here 2^14
f = -Fs/2:1/(NFFT*Ts):Fs/2-1/(NFFT*Ts); % Frequency vector

% Calculating and plotting the spectrum, in absolute scale and in dB
figure(3)
subplot(2,1,1);
plot(f/1e6, fftshift(abs(fft(xt, NFFT)))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude')
title('TX signal amplitude spectrum')
subplot(2,1,2);
plot(f/1e6, fftshift(20*log10(abs(fft(xt, NFFT))))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude [dB]')
title('TX signal amplitude spectrum')
```

Tasks 1

- Plot the relevant responses and explain what you observe.

3 Noisy Multipath Channel Model

Next, we model the channel between the TX and RX. In all electrical and/or electromagnetic communication systems, there is always additive noise, hence that is one central ingredient in our channel modeling part as shown in Fig. 1. Furthermore, in many systems, there is also linear distortion, hence we take that also into account and it is modeled by a linear filter $b(t)$ shown also in Fig. 1. Concrete example of linear distortion is multipath propagation in wireless communications.

▶ First we experiment the linear distortion/multi-path propagation by generating three example impulse responses of a multipath channel:

```
b1 = 1; % means no multipath at all
b2 = [0.9-0.15j 0 -0.2-0.44j 0 0.1+0.36j]; % some multipath components already
b3 = [0.8+0.2j 0 0 0 -0.3+0.68j 0 0 0 0 0.4-0.6j]/1.5; % more harsh multipaths
% These examples are "from the hat" but b2 and b3 anyway reflect scenarios where
% there are two additional paths on top of the most direct path. (But exact
% multipath weights are from the hat)
```

▶ Decide which channel profile to use and apply the channel model to the signal:

```
b = b2;
yt = filter(b,1,xt); % Applying the multipath channel model to the signal
```

▶ Then let's plot the spectrum of the signal where multipath effects are included.

```
figure(4)
subplot(2,2,1);
plot(f/1e6, fftshift(abs(fft(xt, NFFT)))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude')
title('TX signal amplitude spectrum')
subplot(2,2,3);
plot(f/1e6, fftshift(20*log10(abs(fft(xt, NFFT))))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude [dB]')
title('TX signal amplitude spectrum ')
subplot(2,2,2);
plot(f/1e6, fftshift(abs(fft(yt, NFFT)))); grid on;
```

```
xlabel('Frequency [MHz]')
ylabel('Relative amplitude')
title('RX signal amplitude spectrum, with multipath')
subplot(2,2,4);
plot(f/1e6, fftshift(20*log10(abs(fft(yt, NFFT))))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude [dB]')
title('RX signal amplitude spectrum, with multipath')
```

Next, we add white Gaussian noise to the signal. We first create a noise sequence with unit variance (power) and then scale it properly such that a given inband signal-to-noise ratio (SNR) is obtained, when the noise is superimposed with the signal $y(t)$.

- Set SNR in dB and generate a properly scaled noise sequence

```
SNRdB = 15; % Experimented signal-to-noise ratio [dB]

% Complex white Gaussian noise sequence, first of unit power/variance
n = (1/sqrt(2))*(randn(size(yt)) + 1j*randn(size(yt)));

P_y = var(yt); % Signal power
P_n = var(n); % Noise power
% Defining noise scaling factor based on the SNR level:
noise_scaling_factor = sqrt(P_y/P_n./10.^(SNRdB./10)*(r/(1+alpha)));

% perhaps a little cryptic where the exact expression of the scaling factor
% comes from --can you see it ? Keep in mind that noise has wider bandwidth
% than the useful signal, this is where the final r/(1+alpha) factor comes
% from
```

- Add noise on top of the signal $y(t)$

```
rt = yt + noise_scaling_factor*n;
```

- Then let's plot again the amplitude spectra of signals.

```
figure(5)
subplot(2,2,1);
plot(f/1e6, fftshift(abs(fft(xt, NFFT)))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude')
title('TX signal amplitude spectrum')
subplot(2,2,3);
plot(f/1e6, fftshift(20*log10(abs(fft(xt, NFFT))))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude [dB]')
```

```
title('TX signal amplitude spectrum')
subplot(2,2,2);
plot(f/1e6, fftshift(abs(fft(rt, NFFT)))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude')
title('RX signal amplitude spectrum, with multipath+noise')
subplot(2,2,4);
plot(f/1e6, fftshift(20*log10(abs(fft(rt, NFFT))))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude [dB]')
title('RX signal amplitude spectrum, with multipath+noise')
```

Tasks 2

- Vary the `SNRdB` value e.g. few values between 0 . . . 50, and see how that impacts the RX signal spectrum. Provide relevant spectral examples and explain what you observe.
- Explain also the effects of multipath, why does the RX signal spectrum have clear fading notches inside the passband ? To address the issue, plot the amplitude response of the multipath channel on a separate figure. You can do it as

  ```
  figure(99)
  subplot(211)
  plot(f/1e6, fftshift(abs(fft(b, NFFT)))); grid on;
  subplot(212)
  plot(f/1e6, fftshift(20*log10(abs(fft(b, NFFT))))); grid on;
  ```

  When interpreting the results remember that the transmit signal is bandlimited.
- Vary also the multipath channel profile between the channels b1, b2, b3 and explain what you observe (in terms of the RX signal spectrum).

4 Basic Receiver Processing (Filtering and Sampling)

Next we move on to the RX side, where two fundamental tasks are signal filtering (to remove noise outside the signal band) and sampling the signal

4.1 RX filtering and sampling

Filter the received signal $r(t)$ with the receive filter. We assume that the RX filter is also an RRC filter similar to the TX side, and sample the resulting continuous time signal $q(t)$ in order to obtain the discrete time sample sequence $q(k)$.

```
% Creating the receive filter f(t) (it is here the same as in the transmitter)
ft = gt;

% Then filtering the received noisy signal with the chosen RX filter

qt = filter(ft,1,rt); % Receiver filtering
qt = qt(1+(length(ft)-1)/2:end); % Discard filter delay/transient
```

- Plotting the amplitude spectra of the RX filter input and output signals

```
figure(6)
subplot(2,2,1);
plot(f/1e6, fftshift(abs(fft(rt, NFFT)))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude')
title('RX signal amplitude spectrum, before filtering')
subplot(2,2,3);
plot(f/1e6, fftshift(20*log10(abs(fft(rt, NFFT))))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude [dB]')
title(' RX signal amplitude spectrum, before filtering ')
subplot(2,2,2);
plot(f/1e6, fftshift(abs(fft(qt, NFFT)))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude')
title(' RX signal amplitude spectrum, after filtering ')
subplot(2,2,4);
plot(f/1e6, fftshift(20*log10(abs(fft(qt, NFFT))))); grid on;
xlabel('Frequency [MHz]')
ylabel('Relative amplitude [dB]')
title(' RX signal amplitude spectrum, after filtering ')
```

- Then we do the “sampling” at symbol rate, see again Fig. 1

```
% Sampling the filtered RX signal at symbol rate. Remember that we used
% oversampling in the TX, by a factor of r, so this is now just picking every
% r-th sample
qk = qt(1:r:end);
```

- Then we plot the symbol rate RX samples in complex plane (RX constellation), with ideal QAM constellation also shown for reference

```
figure(7)
plot(qk,'b*'); hold on; grid on;
plot(alphabet,'ro', 'MarkerFaceColor','r'); hold off
```

```
legend('Received samples', 'Original symbols')
xlabel('Real Part')
ylabel('Imaginary Part')
title('Received symbol-rate samples (RX constellation)')
axis equal

% With noise and multipath on, this looks messy ...
```

Tasks 3

- First momentarily omit the multipath (i.e. use the channel b1) and set SNR to 35 dB. Plot the RX signal constellation and explain what you see.
- Then repeat by changing the SNR to 10 dB and 20 dB and plot and comment again the RX signal constellation. Would the RX still be able to reliably decode/detect the received signal ?
- Then repeat by setting SNR back to 35 dB but now turning on the multipath channel. Experiment with both multipath channels b2 and b3. Plot always the RX signal constellation and try to explain what you see.
- Then lower the SNR down to 10 dB. Again plot the RX signal constellations with all (multipath) channels b1, b2 and b3 and explain what you see.
- Next, change the modulation order to $M = 4$, and repeat the above steps shortly. Comment on the differences.
- Finally, change the modulation order to $M = 64$, and repeat the above steps shortly. Comment on the differences.

4.2 Modeling a phase noise in receiver

Next we incorporate a non-ideal frequency oscillator into the receiver chain. The short term instability of the oscillators appears as phase noise and sampling jitter, and it is very critical for the performance of the system. This is something that we have not discussed much during lectures and is thus complementary offering a chance to learn something new as well. For your convenience, there is ready-made Matlab function called `phasenoise.m` available at the course Moodle site to shortly experiment this issue. Download the m file, and put it into the folder where your actual source code is. This function models a free-running oscillator with 3 dB noise bandwidth determined by parameter `Beta`. Now we experiment the phase-noise effect on the above single-carrier waveform (without any additional channel noise or multipath of the channel, to be able to more easily visually interpret how phase noise is impacting the signal).
▶ Apply the phase noise effect to the ideal transmit waveform and experiment the constellations of received signal.

```
Beta = 100; % Noise bandwidth in Hz
% Generate phase noise model, i.e. complex exponential with random phase jitter
pt = phasenoise(numel(xt), Fs, Beta);

% Omit AWGN noise and multipath and add phase noise:
% The effect on the used waveform is modeled by multiplying the ideal
% waveform samples with generated phase noise
rt = xt.*pt; %

qt = filter(ft, 1, rt); % Receiver filtering
qt = qt(1+(length(ft)-1)/2:end); % Discard filter delay/transient
qk = qt(1:r:end); % Sample at symbol rate

% Plot the constellation for 1024 symbols
figure(8)
plot(qk(1:1024), 'b*'); hold on; grid on;
plot(alphabet,'ro', 'MarkerFaceColor','r'); hold off
legend('Received samples', 'Original symbols')
xlabel('Real part')
ylabel('Imaginary part')
title('Received symbol-rate samples (RX constellation)')
axis equal
```

Tasks 4

- First set `Beta = 100` and plot the RX signal constellation and explain what you see.
- Then repeat by changing the `Beta = 5000` and plot and comment again the RX signal constellation. Would the RX still be able to reliably decode/detect the received signal?

What to return and how:

- Prepare a PDF document where you provide answers and explanations as requested in the "Tasks" parts
- Include also selected graphics (figures) in the document from Matlab to complement your answers and explanations, but exercise some caution such that you choose those figures that you think are relevant.
- Return your finalized PDF document to the project work supervisor Mr. Roman Klus (contact information at the course Moodle site)
- The **final deadline** for returning the complete project work is **Friday 15th of December, 2023**.