TLT-5400/5406 DIGITAL TRANSMISSION, 2nd Matlab-Exercise

Here we consider complex symbol alphabets, such as QPSK and QAM (Chapter 6.5. in E. A. Lee, D. G. Messerschmitt Digital Communication is very useful as a theoretical framework to this exercise). It is assumed that the oversampling factor $F_s*T=1$, so there is no sense to generate a pulse-shaping filter in this case (it would just act like a delay element).

MAKE an m-file, it helps you to do the exercise!

### Constellation and Symbol Sequence Generation

#### PSK-Signal
- The QPSK-signal consists of symbols $\exp(j*\theta)$, where $\theta = \{\pi/4, 3\pi/4, 5\pi/4, 7\pi/4\}$
- Constellation points are located symmetrically on the unit-circle in the complex-domain

Let’s generate a QPSK-symbol sequence of 20000 symbols and plot the constellation figure!

```matlab
>> konst_qpsk = exp(j*[pi/4 3*pi/4 5*pi/4 7*pi/4].');   % QPSK alphabet
>> qpsk = konst_qpsk(randint(20000,1,4)+1);   % QPSK symbol sequence
>> plot(qpsk,'o'); xlabel('Re');ylabel('Im');   % plot the constellation
>> axis([-2 2 -2 2]);      % Scale the axis of the figure
```

- QPSK is a special case of M-PSK ($M = 4$). In M-PSK, the constellation points are located at regular angular intervals on the unit circle.
- Generate an 8-PSK symbol stream and plot the constellation!

#### QAM-Signal
- The QAM-signal consists of alphabet symbols of $a+j*b$, where $a,b = \{-7,-5,-3,-1,1,3,5,7\}$
- Constellation points are located on a symmetric “grid” in the complex domain. How many amplitude levels are there in this case? How many phases?

Let’s generate a 64-QAM symbol sequence of 20000 symbols and plot the constellation:

```matlab
>> aqam = [-7 -5 -3 -1 1 3 5 7];
>> A = repmat(aqam,8,1);
>> B = flipud(A') ;
>> konst_qam = A+j*B;       % 8x8-Matrix that contains all the constellation points.
>> konst_qam = konst_qam(:);      % Corresponding column-vector, our 64-QAM alphabet.

>> qam = konst_qam(randint(20000,1,64)+1); % 64-QAM symbol sequence
>> plot(qam,'o');        % plot the constellation
>> axis([-8 8 -8 8]);       % Scale the axis of the figure
>> xlabel('Re');ylabel('Im');
```

### Adding Noise to the Signal

Let’s add some white Gaussian noise to the generated complex symbol streams, so that the SNR is 20 dB. The noise must be, like the signal (QPSK or QAM), complex-valued.
- Any communication system always includes some noise, which is usually additive!
- Usually, the noise is assumed to be Gaussian distributed!
- To add some noise in Matlab, the signal-vector and the noise-vector must be vectors of equal size!
- Now, we add some noise to the previously generated signals, so that the SNR (signal-to-noise ratio) becomes 20dB.
- First, we form a complex noise vector as:

```matlab
>> n = randn(size(qam))+j*randn(size(qam));   % randn generates Gaussian distributed noise
```

- Then, we define the variances or powers as:

```matlab
>> svqpsk = std(qpsk)^2;   % QPSK signal variance (power)
>> svqam = std(qam)^2;   % QAM signal variance (power)
>> nv = std(n)^2;   % noise variance (power)
```
SNR is defined as “SNR = 10*\log_{10}(\text{signal\_power/noise\_power}) = 20*\log_{10}(\text{std(signal)/std(noise)})”. Now SNR can be changed by multiplying either the signal or noise vector by a proper constant p. Let’s multiply the noise-vector by variable p => “SNR = 20*\log_{10}(\text{std(signal)/std(p*noise)})”. Now, p can be defined as

```matlab
>> SNR = 20;  % Target SNR = 20 dB

>> p1 = std(qpsk)/(std(n)*10^(SNR/20));  % proper constant p
>> snqpsk = qpsk+n*p1;  % add noise to signal
>> plot(snqpsk,'o');  % plot noisy constellation

>> p2 = std(qam)/(std(n)*10^(SNR/20));
>> snqam = qam+n*p2;  % add noise to signal
>> plot(snqam,'o');  % plot noisy constellation
```

Consider the constellation with SNR = 15, 10, and 5dB, what is the effect of noise on the signal?

We can see, that there are so called Gaussian clouds around the constellation points.

**Signal Detection (symbol by symbol detection)**

In symbol detection, we compare the distance of a received noisy sample to all possible symbol values in the used alphabet. The detected symbol (decision) is the one that minimizes the distance! There are several ways how to do that in Matlab:

**QPSK:**

```matlab
>> sn_block = repmat(snqpsk,1,4);'  % 4x20000-matrix, each line contains the same sn-vector
>> konst_block = repmat(konst_qpsk,1,20000);  % 4x20000-matrix, where each column contains the const.
>> distance = abs(sn_block-konst_block);  % 4x20000-matrix, whose every column contains
% the received symbol distances to all possible symbol
% constellation points
>> [y,ind_1] = min(distance);  % returns the minimum distance y and the corresponding
% constellation index ind_1. Both vectors have the size of 1x20000
>> qpsk_det = konst_qpsk(ind_1);  % using vector ind_1, we can determine the detected symbol vector

Note! ‘ is the non-conjugate (ordinary) transpose while ‘ is the conjugate transpose in Matlab. Using a real valued vector, it doesn’t matter which one is used. But in the case of a complex valued vector, using ‘ changes the sign of imaginary part of signal which is not wanted (usually).

Detection using for-expression (just for reference, you can skip this). Notice! In general, try to avoid using long for-loops because they are not efficient in Matlab.

```matlab
>> for i = 1:20000,
>>      [y,ind_1b] = min(abs(konst_qpsk-snqpsk(i)));
>>      qpsk_det(i) = konst_qpsk(ind_1b);
>> end
```

**QAM:**

```matlab
>> sn_block = repmat(snqam,1,64);'  % 64x20000-matrix, each line contains the same sn-vector
>> konst_block = repmat(konst_qam,1,20000);  % 64x20000-matrix, where each column contains the const.
>> distance = abs(sn_block-konst_block);  % 64x20000-matrix, whose every column contains
% the received symbol distances to all possible symbol
% constellation points
>> [y,ind_2] = min(distance);  % returns the minimum distance y and the corresponding
% constellation index ind_2. Both vectors have the size of 1x20000
>> qam_det = konst_qam(ind_2);  % using vector ind_2, we can determine the detected symbol vector
```
Symbol Error Probability (SEP) / Symbol Error Rate (SER)

Can be calculated both analytically and using computer simulations. Here the idea is to verify the theoretical symbol error probabilities derived during the lectures using simulations.

QPSK:

In QPSK, the theoretical symbol error probability $P$ can be defined as (see Chapter 6.5 or lecture notes)

$$P = 2Q(d/(2\sigma))-Q(d/(2\sigma))^2,$$

where $Q(.)$ is the Q-function, $d$ the minimum distance of two constellation points, and $\sigma^2$ is the variance of the noise (real or imaginary part).

There is no Q-function directly in Matlab, but it can be determined by using erfc-function (see help erfc). Also if the Matlab Communications Toolbox is installed, there is a direct implementation (see help qfunc).

$$Q(x) = 0.5*erfc(x/sqrt(2))$$

Here is the theoretical SEP:

```
>> d = sqrt(2);     % minimum distance d for our QPSK constellation
>> sigma = std(real(n*p1));   % sigma is the deviation of noise (real or imaginary part)
>> Q = 0.5*erfc(d/(sqrt(2)*2*sigma));
>> sep_theo = 2*Q - Q^2    % returns the symbol error probability
```

In computer simulation, we compare transmitted symbols with detected symbols and count how many errors occur. To obtain the symbol error probability, the number of errors is divided by the total number of symbols. The lower probabilities we want to consider the more symbols are needed in the simulation.

```
>> number_of_errors = sum(qpsk ~= qpsk_det);  % The comparison returns value 0 when its false and value 1
>> sep_simu = number_of_errors/20000          % when it’s true.
```

Does the analytical method differ from the computer simulation? Why/why not?

What is the SEP in the case of SNR=0,1,2,3,4,5,...,10dB? Use both the analytical model and computer simulations! Plot the SEP as a function of SNR. **Compare the analytical and computer simulation results - what can be said about the reliability of the simulation? Any idea how to improve it?** (You can use e.g. a for loop to iterate over the given SNR values.)

QAM:

For 64-QAM, the theoretical SEP (see the additional M-QAM symbol error probability material) is of the form

$$P = 3.5Q(d/(2\sigma))-3.0625Q(d/(2\sigma))^2,$$

where $Q(.)$ is Q-function, $d$ the minimum distance of two constellation points, and $\sigma^2$ is the variance of noise real or imaginary part.

```
>> d = 2;     % minimum distance d for our QAM alphabet
>> sigma = std(real(n*p2));   % sigma is the deviation of noise real or imaginary part
>> Q = 0.5*erfc(d/(sqrt(2)*2*sigma));
>> sep_theo = 3.5*Q - 3.0625*Q^2  % returns the symbol error probability
```

Simulated SEP:

```
>> number_of_errors = sum(qam ~= qam_det);
>> sep_simu = number_of_errors/20000   % simulated symbol error probability
```

Again try with different SNRs, e.g., from 15 dB … 25 dB and compare theoretical and simulated SEPs. **In general, what would be the approximate probability of bit-error in the case of using Gray-coding?**