1-st practical exercise
Digital signal receiving with additive white gaussian noise

The purpose of the work is to show the very basic principles of creating Matlab simulation model for transmission simulation in channel with noise.

For this 1-st work, there are no individual parameters of system. The report should be prepared and sent to email address: elans.grabs@rtu.lv. The report must include the following:

1. The objective of practical exercise;
2. The full source code of simulation programs;
3. The plots obtained during simulation for different testing scenarios.
4. The conclusion on noise immunity of two compared modulation types.

The tasks to be solved:

1. Choose a random sequence of "1" and "0" binary symbols and display this discrete process as plot;
2. Convert this discrete signal to unipolar pulses with amplitude 1 V, plot the result
3. Introduce additive white Gaussian noise to this signal with 20 dB SNR ratio, plot the result
4. Create a simplified signal detector with a threshold of 0.5 V, plot the detected signal
5. Compare detection result with transmitted data and see if there are any transmission problems.
6. Repeat steps 3. to 5. for different SNR value of 10 dB.
7. Convert signal to NRZ (bipolar pulse signal) and repeat step 6 (with 10 dB SNR value).
8. Make conclusions on noise immunity of these two modulation types.

The guidelines for practical exercise

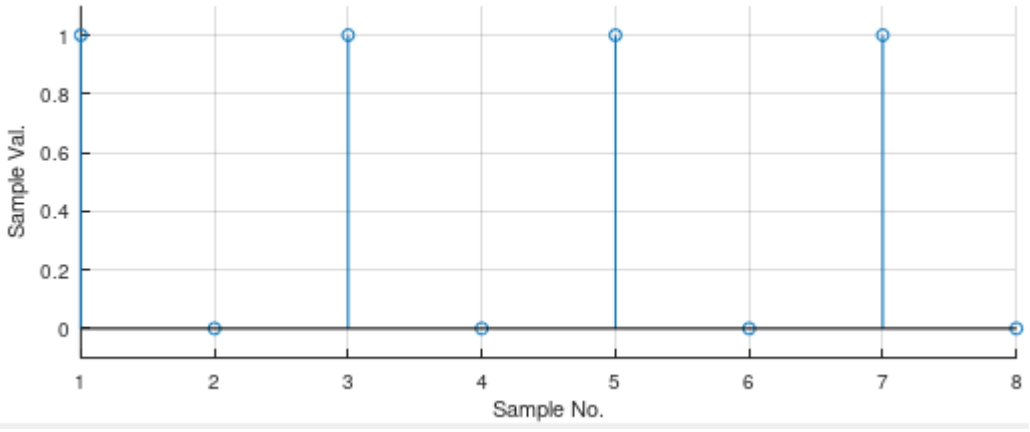
1. The initial data can be manually selected and stored in a vector (pick a total of 10 values):

```
s = [1, 0, 1, 1, 0, 1, 0, 0, …]; % Samples of digital signal
```

To visualize data, different plotting functions can be used. The most appropriate one for discrete process is *stem( )* function, which can be used as follows:

```
stem(s);                % Plot discrete signal
```

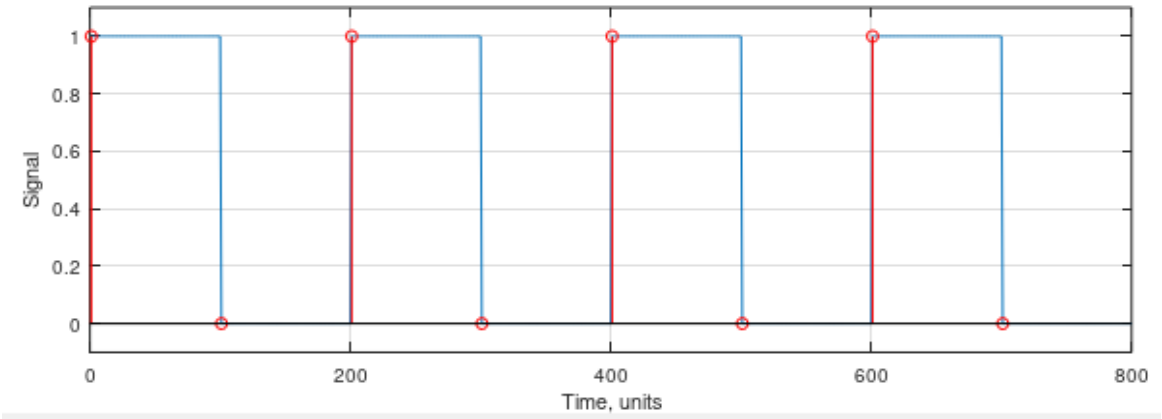
Note: Use "%" symbol to write comments.
The resulting plot should be similar with the figure below:



2. In order to transmit the signal over communications line, it must be converted into analog signal (modulated). In the first scenario, we use unipolar rectangular pulses to send data with amplitude 1:

- 1V voltage corresponds to binary “1”;
- 0V voltage corresponds to binary “0”.

Note, that Matlab is a software running in a computer – a digital device. So, we need a “representation” of analog signal for our model. This can be achieved by repeating each symbol, for example, 100 times to make an effect of signal continuity, as it is shown in figure below.



There are multiple ways to achieve this, the most straight-forward would be use of Kroneker multiplication:

```
SC = ones(1,100);      % A pattern, representing constant level of
                       % analog signal
sTx = kron(s,SC);      % Use Kroneker multiplication to perform DAC
                       % "conversion"
```

Afterwards, the signal can be displayed by using *plot( )* function, which is used similarly to *stem( )*:

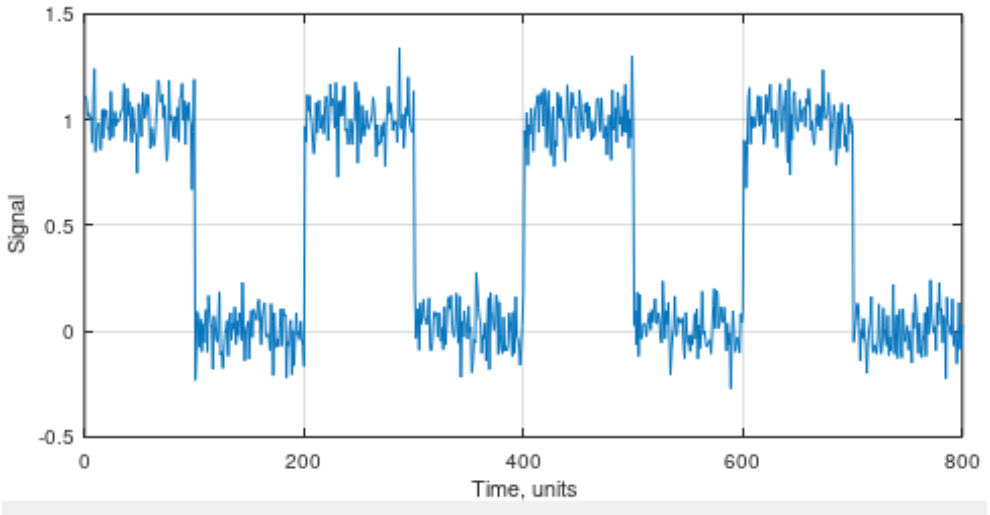
3. The noise can be added by using built-in function. For octave users, it is necessary to load corresponding package after the program startup with command:

```
pkg load communications
```

Afterwards, built-in function *awgn( )* can be used with specified SNR value:

```
SNR = 20;                  % Specify signal to noise ratio in dB
sRx = awgn(sTx,SNR);       % Add white noise with specified power
                           % to a signal
```

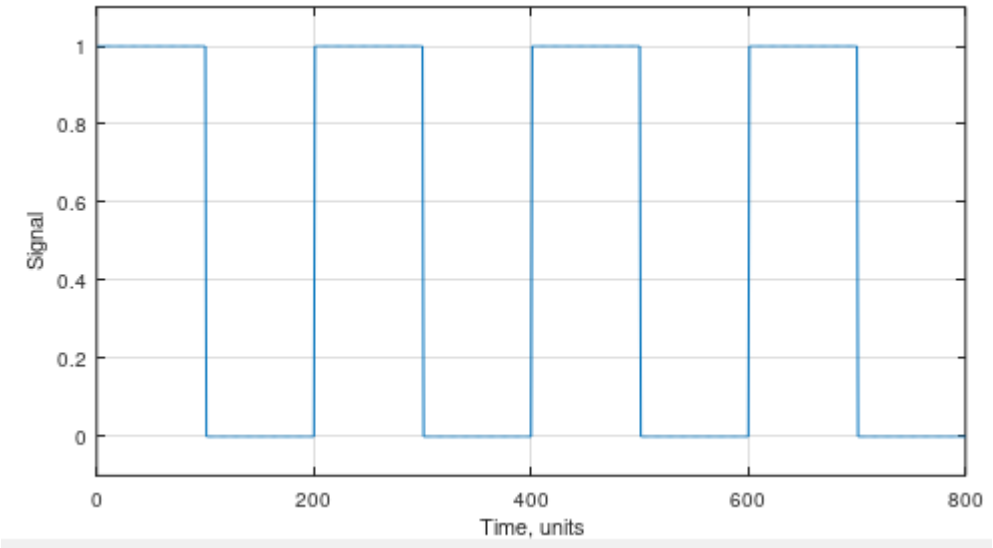
Then plot the received signal in a similar way, as it has been described in 2-nd guideline. The resulting plot should look similar to displayed below:



4. The detection of a signal is performed by comparing noisy value with a threshold of 0.5, which is roughly the mid-point between minimum and maximum values of un-noisy signal. This can be achieved in multiple ways, the simplest in Matlab is comparison operation:

```
% If a signal level >0.5 it is "1", else it is "0" binary symbol
sDt = sRx > 0.5;
```

Then, after plotting the signal the expected result should be similar with a figure below:



5. You can freely change SNR values and test the result, by executing program again.
In order to create NRZ signal, some mathematics can be performed to convert data values as follows:

- "1" → -1 level
- "0" → +1 level

The most straight-forward way to do this is as follows:

```
% Convert to NRZ code. "1" matches -1, "0" matches +1 voltage level
% sTx = -2*sTx + 1;
```

Note, that this must be performed **before adding noise** with *awgn()* function. Also, note that this will change a threshold to 0 and comparison sign:

```
% If a signal level <0 it is "1", else it is "0" binary symbol
sDt = sRx > 0.5;
```