Table of Contents

Example with equalizer

Represent a complete communication system with a not flat channel response, and equalizer filter.

```
clc; clear; close all;
```

Parameters

```
symbol_qtty = 1e5;                  % Number of symbols to transmit.
EsNo_dB = 0:2:30;                   % EsNo.
mod_type = "QAM";                   % Modulation type.
M = 16;                             % Number of symbols.
L = 10;                             % Oversampling factor
beta = 0.8;                         % Roll-off SRRC

%h_c = [0.04 -0.05 0.07 -0.21 -0.5 0.72 0.36 0.21 0.03 0.07]; % Channel A
%h_c=[0.407 0.815 0.407]; % uncomment this for Channel B
h_c=[0.227 0.460 0.688 0.460 0.227]; % uncomment this for Channel C

% Taps for FIR filters. Note: The bigger this value, the closer the
% theoretical and simulated SERs are.
nTaps = 50;

nTaps_equ = 31;                 % Taps for equalizer
```

Calculations

```
ser_no_isi = zeros(1, length(EsNo_dB));            % Symbol error rate
ser_zf_equalizer = zeros(1, length(EsNo_dB));
ser_mmse_equalizer = zeros(1, length(EsNo_dB));

d = randi([0, M-1], 1, symbol_qtty); % Input symbols

% Modulator
[u, constellation] = Modulator.modulate(d, mod_type, M);
v = Modulator.upsample(u, L);
[s, p, delay_tx] = Modulator.pulse_shaping_srrc(v, beta, L, nTaps);

for i=1:1:length(EsNo_dB)
```

NO ISI

Channel

```
    r = Channel.add_awgn_noise(s, EsNo_dB(i), L);

    % Demodulator
    [v_r, g, delay_rx] = Demodulator.pulse_filter_srrc(r, beta, L, nTaps);
    u_r = Demodulator.downsample(v_r, L, delay_rx + delay_tx);
    d_r = Demodulator.demodulate(u_r, mod_type, M, constellation);
    ser_no_isi(i) = sum(d~=d_r)/symbol_qtty;
```

ZF EQUALIZER

Channel

```
    r = Channel.add_awgn_noise(s, EsNo_dB(i), L, h_c);

    % Demodulator
    [a_r, w_zf, n0_zf] = Demodulator.zf_equalizer(r, h_c, nTaps_equ);
    [v_r, g, delay_rx] = Demodulator.pulse_filter_srrc(a_r, beta, L, nTaps);
    u_r = Demodulator.downsample(v_r, L, delay_rx + delay_tx);

    % Hard cap the amount of symbols. A "tail" is added to the vector
    % because of the equalizer, which is a non symetrical FIR filter.
    u_r = u_r(1:1:symbol_qtty);

    d_r = Demodulator.demodulate(u_r, mod_type, M, constellation);
    ser_zf_equalizer(i) = sum(d~=d_r)/symbol_qtty;
```

MMSE EQUALIZER

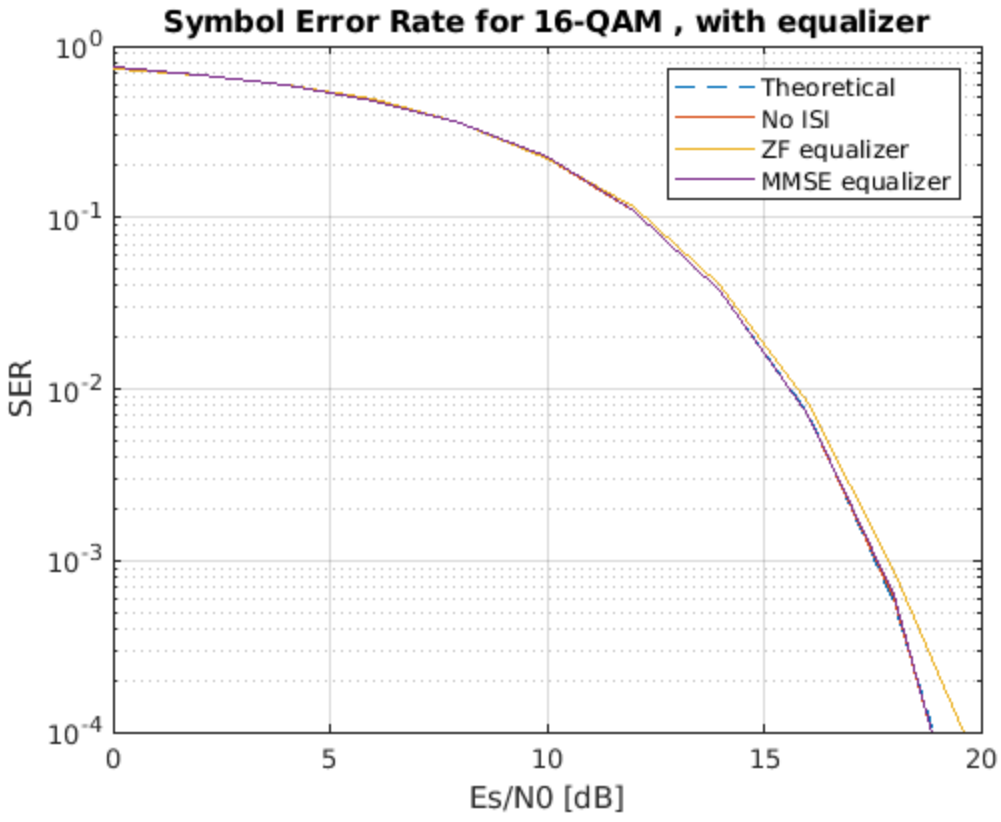
Channel

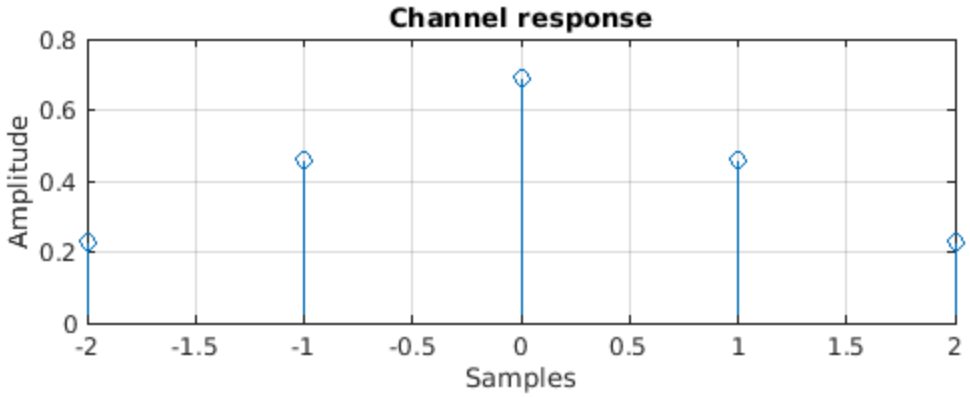
```
    r = Channel.add_awgn_noise(s, EsNo_dB(i), L, h_c);

    % Demodulator
    [a_r, w_mmse, n0_mmse] = Demodulator.mmse_equalizer(r, h_c, nTaps_equ,
 EsNo_dB(i));
    [v_r, g, delay_rx] = Demodulator.pulse_filter_srrc(a_r, beta, L, nTaps);
    u_r = Demodulator.downsample(v_r, L, delay_rx + delay_tx);
    u_r = u_r(1:1:symbol_qtty);

    d_r = Demodulator.demodulate(u_r, mod_type, M, constellation);
    ser_mmse_equalizer(i) = sum(d~=d_r)/symbol_qtty;
end

ser_theory = Theory.ser_AWGN(mod_type, M, EsNo_dB);
```

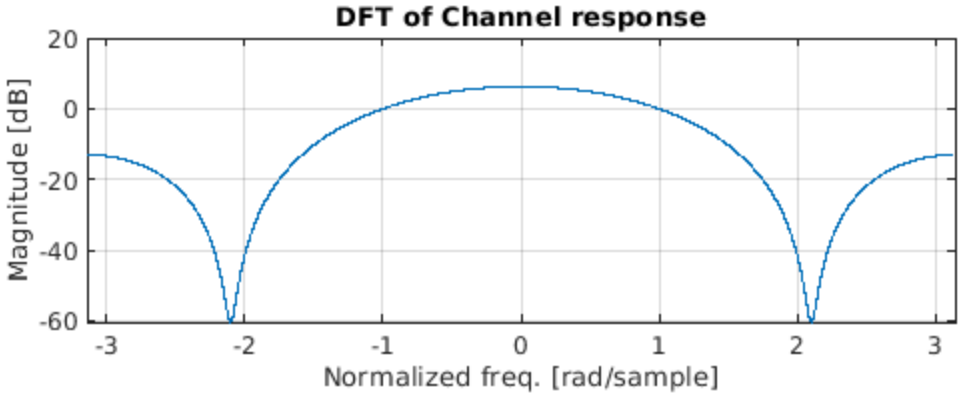
Plotting

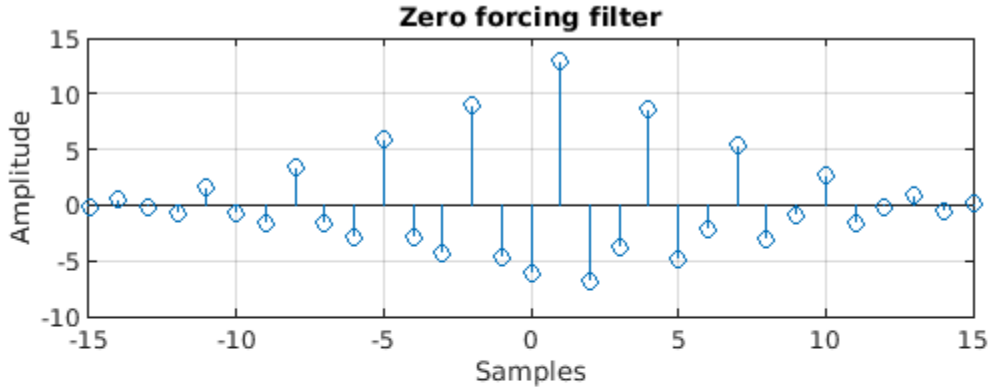
```
semilogy(EsNo_dB, ser_theory, LineStyle="--"); hold on;
```

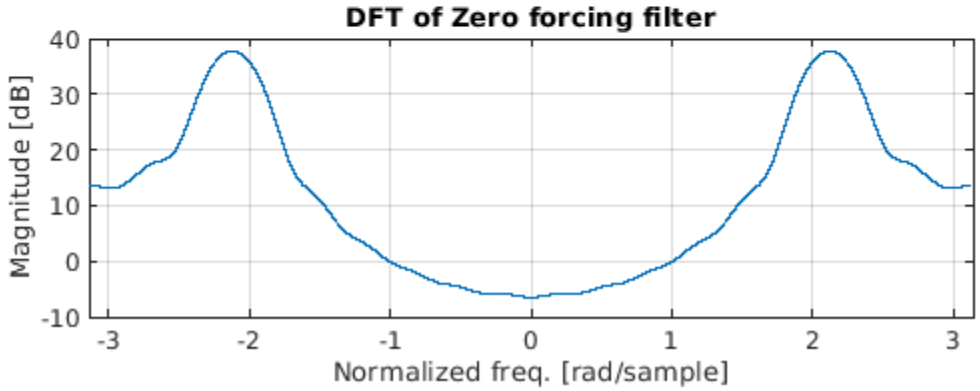
```
semilogy(EsNo_dB, ser_no_isi, LineStyle="-"); hold on;
semilogy(EsNo_dB, ser_zf_equalizer, LineStyle="-"); hold on;
semilogy(EsNo_dB, ser_mmse_equalizer, LineStyle="-"); hold on;

grid on;
legend("Theoretical", "No ISI", "ZF equalizer", "MMSE equalizer");
xlabel("Es/N0 [dB]");
ylabel("SER");
title(strcat("Symbol Error Rate for ", num2str(M), "-", mod_type ," , with
 equalizer"));
ylim([1e-4, 1]);

Scope.plot_fir_filter(h_c, Title="Channel response")
Scope.plot_fir_filter(w_zf, Title="Zero forcing filter");
Scope.plot_fir_filter(w_mmse, Title="MMSE filter");
Scope.plot_IQ(u_r)
```

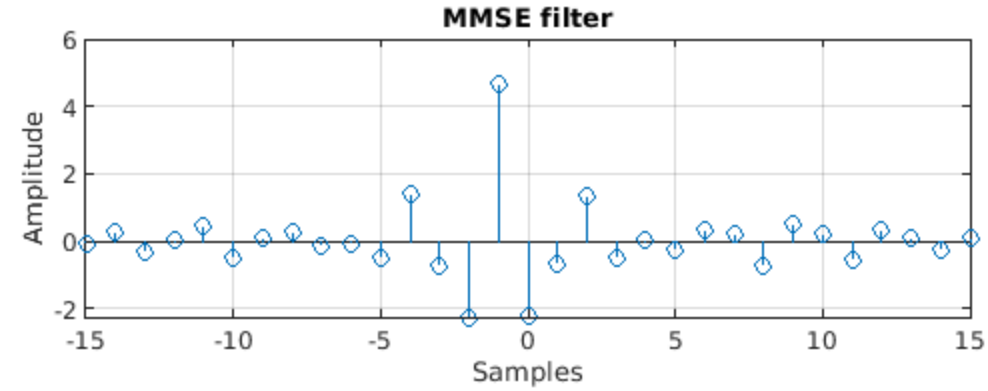


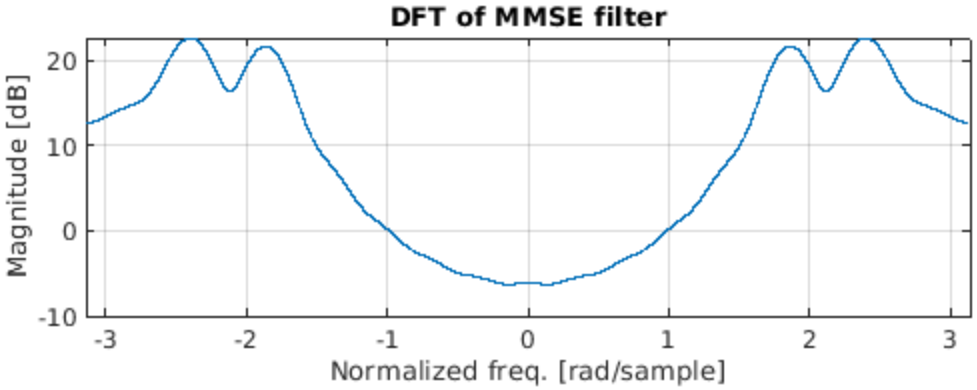

Channel response
Amplitude
Samples

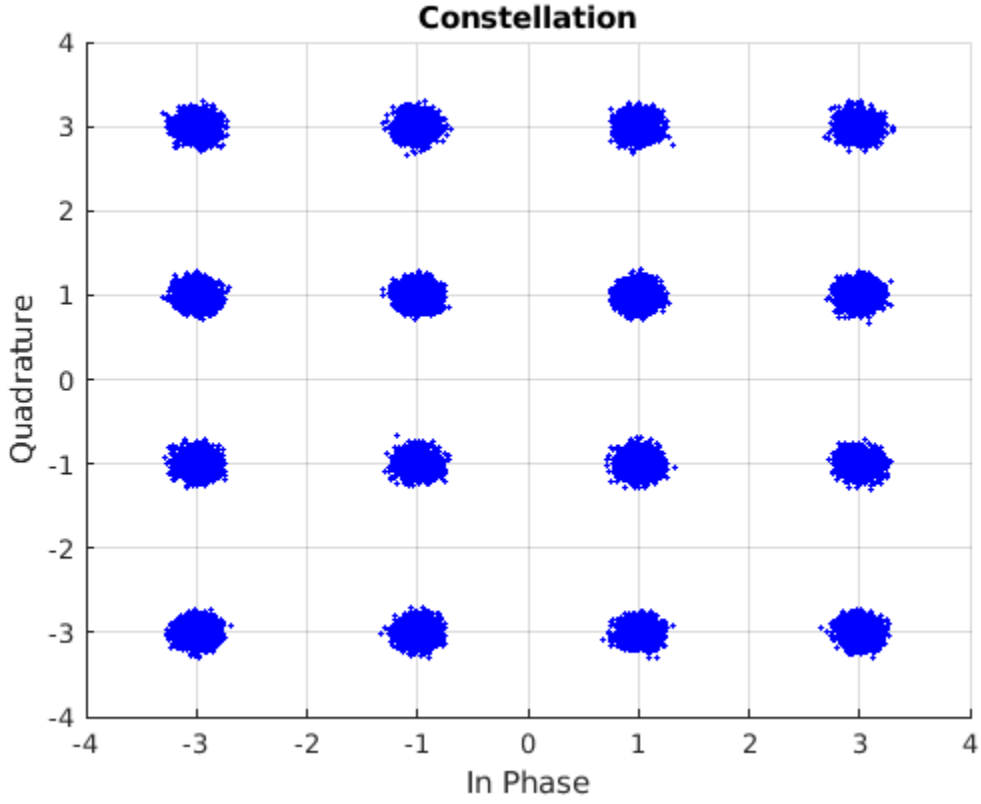

DFT of Channel response
Magnitude [dB]
Normalized freq. [rad/sample]


Zero forcing filter
Amplitude
Samples


DFT of Zero forcing filter
Magnitude [dB]
Normalized freq. [rad/sample]


MMSE filter
Amplitude
Samples
6
4
2
0
-2
-15
-10
-5
0
5
10
15


DFT of MMSE filter
Magnitude [dB]
Normalized freq. [rad/sample]
20
10
0
-10
-3
-2
-1
0
1
2
3


Constellation
Quadrature
In Phase
4
3
2
1
0
-1
-2
-3
-4

Published with MATLAB® R2022b