Table of Contents

```
clc
clear all
close all
warning('off','all')
tic
```

Enabling Options

```
enable_AFT = true;
enable_OTFS = true;
enable_OTFS_LMMSE = true
```

OTFS parameters

```
% number of symbol
N = 16
% number of subcarriers
M = 16
% size of constellation
M_mod = 64
M_bits = log2(M_mod);
% number of symbols per frame
N_syms_perfram = N*M;
% number of bits per frame
N_bits_perfram = N*M*M_bits;
```

AFT parameters

```
% number of subcarriers
N_AFT = M;
% number of AFT symbol
Num_AFT_sym = N;
% noise poser
```

```
SNR_dB = 0:5:30;
SNR = 10.^(SNR_dB/10);
noise_var_sqrt = sqrt(1./SNR);

% Singal Power--> calculated in iesn0 = 0
sig_energy_OTFS = 0;
sig_energy_AFT = 0;

rng(1)
N_fram = 1000;%10^4;
err_ber_OTFS = zeros(length(SNR_dB),1);
err_ber_AFT = zeros(length(SNR_dB),1);
for iesn0 = 0:length(SNR_dB)
    for ifram = 1:N_fram
```

random input bits generation

```
        data_info_bit = randi([0,1],N_bits_perfram,1);
        data_temp =
 bi2de(reshape(data_info_bit,N_syms_perfram,M_bits));
        x = qammod(data_temp,M_mod,'gray');
        x = reshape(x,N,M);
```

channel generation

```
        [taps,delay_taps,Doppler_taps,chan_coef] =
 OTFS_channel_gen(N,M);
        N_CP = max(delay_taps);
        % for the moment, we assume two-tap delay channel
        if taps == 2
            [c0, c1, c2] = ComputeC0_C1_for2path(Doppler_taps,
 delay_taps);
        end
```

Modulation

```
        if enable_OTFS
            % OTFS modulation
            s_OTFS = OTFS_modulation(N,M,x);
        end
        if enable_AFT
            % AFT modulation
            s_AFT = AFT_modulation(N_AFT,Num_AFT_sym, N_CP, c1, c2,
 x);
        end
```

Calculate the Signal Energy

```
        sig_energy = 0;
        if iesn0 == 0
            if enable_OTFS
```

```
                sig_energy =
OTFS_Sig_energy(N,M,taps,delay_taps,Doppler_taps,chan_coef,s_OTFS);
                sig_energy_OTFS = sig_energy_OTFS + sig_energy;
            end
            if enable_AFT
                % AFT
                sig_energy = AFT_Sig_energy(N_AFT, Num_AFT_sym, taps,
delay_taps, Doppler_taps, chan_coef,s_AFT);
                sig_energy_AFT = sig_energy_AFT + sig_energy;
            end
            continue;
        end
```

channel output

```
        if enable_OTFS
            % OTFS
            % H_OTFS_eq is the equivalent channel matrix which is used
for
            % the MMSE equalizer
            [r_OTFS, H_OTFS_eq] =
OTFS_channel_output(N,M,taps,delay_taps,Doppler_taps,chan_coef,sigma_2_OTFS(iesn0
        end
        if enable_AFT
            % AFT
            r_AFT = AFT_channel_output(N_AFT, Num_AFT_sym, taps,
delay_taps, Doppler_taps, chan_coef,sigma_2_AFT(iesn0),s_AFT); % OTFS
        end
```

OTFS demodulation

```
        if enable_OTFS
            % MMSE
            if enable_OTFS_LMMSE
                r_OTFS = H_OTFS_eq'*(H_OTFS_eq*H_OTFS_eq'
+sigma_2_OTFS(iesn0)/sig_energy_OTFS_sqrt^2*eye(M*N))^(-1)*r_OTFS;
            end
            y_OTFS = OTFS_demodulation(N,M,r_OTFS);
        end
        if enable_AFT
            y_AFT = AFT_demodulation(N_AFT,Num_AFT_sym, c0, c1,
c2,r_AFT);
        end
```

detector

```
        if enable_OTFS
            if enable_OTFS_LMMSE
                x_est_OTFS = y_OTFS;
            else
                x_est_OTFS =
OTFS_mp_detector(N,M,M_mod,taps,delay_taps,Doppler_taps,chan_coef,sigma_2_OTFS(ie
            end
        end
        if enable_AFT
            x_est_AFT = AFT_mp_detector(N_AFT, Num_AFT_sym, c0, c1,
 c2,taps,delay_taps,Doppler_taps,chan_coef, y_AFT);
        end
```

output bits and errors count

```
            if enable_OTFS
                % OTFS
                data_demapping = qamdemod(x_est_OTFS,M_mod,'gray');
                data_info_est =
 reshape(de2bi(data_demapping,M_bits),N_bits_perfram,1);
                errors = sum(xor(data_info_est,data_info_bit));
                err_ber_OTFS(iesn0) = errors + err_ber_OTFS(iesn0);
            end
            if enable_AFT
                % AFT
                x_est_AFT_serial           =
 reshape(transpose(x_est_AFT) ,
[1,size(x_est_AFT,1)*size(x_est_AFT,2)]);
                data_demapping = qamdemod(x_est_AFT_serial, M_mod,'gray');
                data_info_est =
 reshape(de2bi(data_demapping,M_bits),N_bits_perfram,1);
                errors = sum(xor(data_info_est,data_info_bit));
                err_ber_AFT(iesn0) = errors + err_ber_AFT(iesn0);
            end
            if mod(ifram, 100) == 0
                ifram
            end

        end
        if iesn0 ==0
            sig_energy_OTFS_sqrt = sqrt(sig_energy_OTFS/N_fram);
            sig_energy_AFT_sqrt = sqrt(sig_energy_AFT/N_fram);
            sigma_2_OTFS = abs(sig_energy_OTFS_sqrt*noise_var_sqrt).^2;
            sigma_2_AFT = abs(sig_energy_AFT_sqrt*noise_var_sqrt).^2;
        end
    end
    if enable_OTFS
        err_ber_fram_OTFS = err_ber_OTFS/N_bits_perfram./N_fram
        semilogy(SNR_dB, err_ber_fram_OTFS,'-*','LineWidth',2);
        title(sprintf(['N = ' num2str(N) ', M = ' num2str(M) ', '
 num2str(M_mod) 'QAM']))
        ylabel('BER'); xlabel('SNR in dB');grid on
        hold on
    end
    if enable_AFT
        err_ber_fram_AFT = err_ber_AFT/N_bits_perfram./N_fram
        semilogy(SNR_dB, err_ber_fram_AFT,'-*','LineWidth',2);
    end
    if enable_OTFS_LMMSE
        legend('OTFS MMSE', 'AFT');
    else
        legend('OTFS Message Passing', 'AFT');
    end
    toc
```

Published with MATLAB® R2018b