تمرین شماره۸

علیرضا حسینی

شماره دانشجویی : ۸۱۰۱۰۱۱۴۲

مخابرات پیشرفته

دکتر الفت

بهار 1403

فهرست مطالب

فهرست اشکال

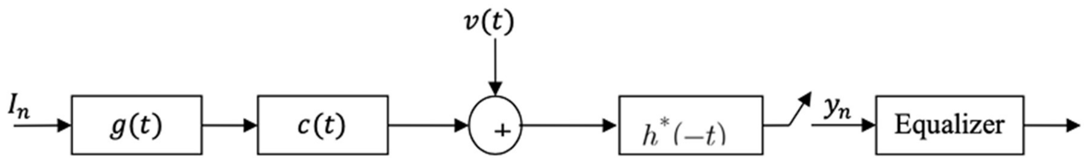
۱-۱- مقدمه

در سیستم‌های ارتباطی دیجیتال مدرن، انتقال داده قابل اعتماد از طریق کانال‌های نویزی بسیار مهم است. این امر مستلزم طرح های مدولاسیون قوی، فیلتر کردن، و تکنیک هایی برای کاهش اثرات نامطلوب نویز و تداخل بین نمادی (ISI) است.BPSK یک تکنیک مدولاسیون ساده و در عین حال مؤثر است که معمولاً در چنین سیستم هایی استفاده می شود. این گزارش عملکرد مدولاسیون BPSK را در ارتباط با استراتژی های مختلف بررسی می کند. به طور خاص، ما اکولایزرهای Zero Forcing (ZF)، حداقل میانگین مربع خطا (MMSE) و Decision Feedback (DFE) را تحلیل و مقایسه می کنیم. علاوه بر این، عملکرد این اکولایزرها با یک گیرنده بهینه Viterbi در کنار هم قرار می گیرد تا بینش جامعی در مورد کارایی نسبی آنها ارائه دهد. از طریق این تجزیه و تحلیل سیستماتیک، هدف ما روشن کردن استراتژی یکسان سازی بهینه برای سیستم های BPSK تحت شرایط کانال های مختلف است.

در ادامه این گزارش ابتدا مدل گسسته و محاسبات آن آورده شده است و در بخش بعدی محاسبات مربوط به هر equalizer و نوشتن function های هر کدام از آن ها آورده شده است و در ادامه کد اصلی و تولید داده ها و شبیه سازی اصلی توضیح داده میشود و در بخش بعدی نتایج هر بخش به طور کامل آورده شده است و با تمامی حالت های تئوری و بدون ISI و با Viterbi مقایسه جامع و کامل انجام شده است.

۲-۱- مدل گسسته سیستم و محاسبات آن

مدل داده در شکل زیر نمایش داده شده است.



شکل (۱-۱) مدل داده شده در صورت سوال

با توجه به معادله g(t) و c(t) داده شده به صورت زیر :

$$g(t) = \begin{cases} \frac{1}{\sqrt{T}} & 0 \le t \le T \\ 0 & \text{otherwise} \end{cases}$$

که به ازای T=1 داریم :

$$g(t) = \begin{cases} 1 & 0 \le t \le 1 \\ 0 & \text{otherwise} \end{cases}$$

و برای کانال داریم :

$$c(t) = \begin{cases} \sqrt{\frac{3}{2T}}\left(1 - \frac{t}{2T}\right) & 0 \le t \le 2T \\ 0 & \text{otherwise} \end{cases}$$

که به ازای T=1 میشود.

$$c(t) = \begin{cases} \sqrt{\frac{3}{2}}\left(1 - \frac{t}{2}\right) & 0 \le t \le 2 \\ 0 & \text{otherwise} \end{cases}$$

باید کانولوشن $h(t) = g(t) * c(t)$ را پیدا کنیم که به صورت زیر محاسبه میشود.

$$h(t) = \int_{-\infty}^{\infty} g(\tau)c(t-\tau)d\tau$$

محاسبات را به صورت زیر در 3 ناحیه انجام میدهیم.

1. For $0 \le t \le 1$:

$$h(t) = \int_0^t g(\tau)c(t-\tau)d\tau = \int_0^t 1 \cdot \sqrt{\frac{3}{2}}\left(1 - \frac{\tau}{2}\right)d\tau = \sqrt{\frac{3}{2}}\int_0^t \left(1 - \frac{t}{2} + \frac{\tau}{2}\right)d\tau$$

$$h(t) = \sqrt{\frac{3}{2}}\left[\tau - \frac{t}{2}\tau + \frac{\tau^2}{4}\right]_0^t = \sqrt{\frac{3}{2}}\left[t - \frac{t^2}{2} + \frac{t^2}{4}\right] = \sqrt{\frac{3}{2}}\left[t - \frac{t^2}{4}\right] = \sqrt{\frac{3}{2}}t\left(1 - \frac{t}{4}\right)$$

2. For $1 \le t \le 2$:

$$h(t) = \int_0^1 g(\tau)c(t-\tau)d\tau = \int_0^1 1 \cdot \sqrt{\frac{3}{2}}\left(1 - \frac{t-\tau}{2}\right)d\tau = \sqrt{\frac{3}{2}}\int_0^1 \left(1 - \frac{t}{2} + \frac{\tau}{2}\right)d\tau$$

$$h(t) = \sqrt{\frac{3}{2}}\left[\tau - \frac{t}{2}\tau + \frac{\tau^2}{4}\right]_0^1 = \sqrt{\frac{3}{2}}\left[1 - \frac{t}{2} + \frac{1}{4}\right] = \sqrt{\frac{3}{2}}\left(\frac{5}{4} - \frac{t}{2}\right)$$

3. For $2 \le t \le 3$:

$$h(t) = \int_{t-2}^1 g(\tau)c(t-\tau)d\tau = \int_{t-2}^1 1 \cdot \sqrt{\frac{3}{2}}\left(1 - \frac{t-\tau}{2}\right)d\tau = \sqrt{\frac{3}{2}}\int_{t-2}^1 \left(1 - \frac{t}{2} + \frac{\tau}{2}\right)d\tau$$

$$h(t) = \sqrt{\frac{3}{2}}\left(\frac{9}{4} - \frac{3t}{2} + \frac{t^2}{4}\right)$$

با توجه به اینکه h در بازه 0 تا 3 مقدار دارد پس x که $h(t) * h * (-t)$ به ازای 2, 1, 0 , 1- , 2- مقدار

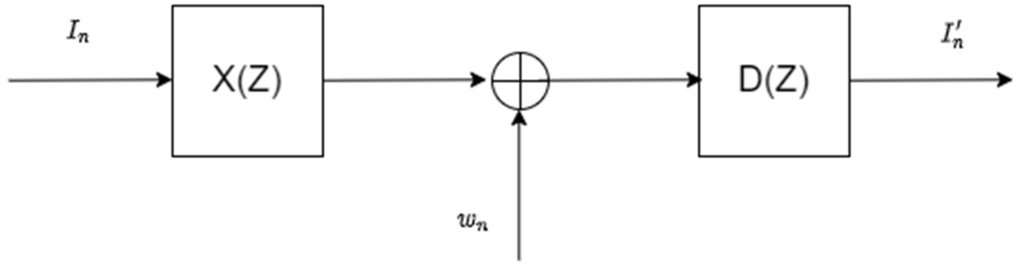
خواهد داشت که پس از محاسبات آن در متلب مقادیر آن به صورت زیر میشود.

$$x[n] = \begin{cases} 0.75 & n = 0 \\ 0.3437 & n = 1, -1 \\ 0.0281 & n = 2, -2 \\ 0 & \text{otherwise} \end{cases}$$

بدین ترتیب داریم :

$$X(z) = 0.75 + 1.1\,(Z^{-1} + Z^{1}) + 0.1219(Z^{-2} + Z^{2})$$

و مدل به صورت زیر میشود.



شکل (۲-۱) مدل نهایی با اکوآلایزر

که در مدل شکل 2 داریم :

$$S_w(z) = 2N_0X(z)$$

۳-۱- طراحی و محاسبات equalizer ها و سایر الگوریتم ها

1-3-1- Zero-Forcing (ZF)

هدف اصلی یک اکوالایزر صفرکننده (ZF) این است که پاسخ ضربه سیستم معادل را به گونه‌ای تنظیم کند که تداخل بین‌نمادی به حداقل برسد ولی به نوی توجهی ندارد

برای این منظور، q_0 باید برابر با 1 و سایر q ها برابر با 0 باشند. q_n به صورت $d_n * x_n$ تعریف می‌شود. که بسته به تعداد tap ها معادله و مجهولاتی دارد که پس از حل آن ها d[n] تعیین میشود.

برای حل آن به صورت زیر عمل میکنیم.

```
% ZF Equalizer design
function d = zf_equalizer(K, x)
    A = zeros(2*K+1);
```

```
    for i = 1:2*K+1
        A(i,i) = x(3);
        if i+1 <= 2*K+1, A(i,i+1) = x(4); end
        if i+2 <= 2*K+1, A(i,i+2) = x(5); end
        if i-1 >= 1, A(i,i-1) = x(2); end
        if i-2 >= 1, A(i,i-2) = x(1); end
    end
    b = zeros(2*K+1, 1);
    b(K+1) = 1;
    d = pinv(A) * b;
end
```

که در کد فوق،

تابع `zf_equalizer` با دریافت دو پارامتر k و x که k تعداد تپ‌های اکوالایزر و x پاسخ ضربه سیستم است، اکوالایزر صفرکننده را طراحی می‌کند.

- ابتدا، ماتریس A با ابعاد 2k+1 * 2k+1 ایجاد می‌شود و مقادیر اولیه آن صفر تنظیم می‌شود.

- سپس، حلقه‌ای برای پر کردن ماتریس A با مقادیر مناسب از x اجرا می‌شود. در این حلقه، مقدار A(i,i) برابر با x(3) قرار داده می‌شود. همچنین، مقادیر A(i,i+1) و A(i,i+2) با x(4) و x(5) و مقادیر A(i,i-1) و A(i,i-2) با x(2) و x(1) پر می‌شوند. که x به صورت زیر میباشد.

```
x = [0.0281 0.3437 0.75 0.3437 0.0281]
```

- بردار b با ابعاد 1 * (2K+1) ایجاد شده و مقدار وسط آن برابر با 1 قرار داده می‌شود

- در نهایت، از تابع `pinv` برای محاسبه شبه‌معکوس ماتریس A و ضرب آن در بردار b استفاده می‌شود تا مقادیر d به دست آیند.

با این روش، اکوالایزر صفرکننده به گونه‌ای طراحی می‌شود که پاسخ ضربه سیستم را بهبود دهد و تداخل بین‌نمادی را به حداقل برساند.

MMSE -1-3-2

اکوالایزرهای MMSE (حداقل مربعات میانگین) برای کاهش خطای مجموع مربعات میانگین بین سیگنال ورودی و خروجی طراحی می‌شوند. این اکوالایزرها در حضور نویز و تداخل بین‌نمادی (ISI) عملکرد بهتری نسبت به اکوالایزرهای صفرکننده دارند.

بر اساس آموزش‌های کلاس، برای محاسبه اکوالایزر MMSE باید از ماتریس‌های خودهمبستگی R_y و R_{IY} استفاده کنیم. در نظر داشته باشید که

$$E\{|I_n|^2\} = \frac{1}{2}\times 1 + \frac{1}{2}\times 1 = 1$$

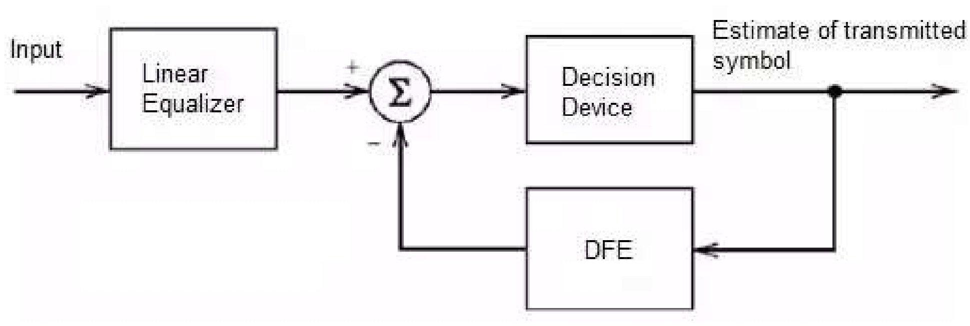
```
% MMSE Equalizer design
function d = mmse_equalizer(R_y, R_IY, K)
    A = zeros(2*K+1);
    for i = 1:2*K+1
        A(i,i) = R_y(5);
        if i+1 <= 2*K+1, A(i,i+1) = R_y(4); end
        if i+2 <= 2*K+1, A(i,i+2) = R_y(3); end
        if i+3 <= 2*K+1, A(i,i+3) = R_y(2); end
        if i+4 <= 2*K+1, A(i,i+4) = R_y(1); end
        if i-1 >= 1, A(i,i-1) = R_y(6); end
        if i-2 >= 1, A(i,i-2) = R_y(7); end
        if i-3 >= 1, A(i,i-3) = R_y(8); end
        if i-4 >= 1, A(i,i-4) = R_y(9); end
    end
    b = zeros(2*K+1, 1);
    b(K+1-floor(length(R_IY)/2):K+1+floor(length(R_IY)/2)) = R_IY;
    d = pinv(A) * b;
end
```

در کد فوق،

در این کد، تابع `mmse_equalizer` با دریافت سه پارامتر R_y و R_{IY} و k که k تعداد تپ‌های اکوالایزر، است، اکوالایزر حداقل مربعات میانگین را طراحی می‌کند.

DFE -1-3-3

اکوالایزر DFE شکل آن در زیر آمده است.



شکل (۳-۱) اکوالایزر DFE

با توجه به توضیحات درس هدف حل معادله زیر میباشد.

$$\begin{bmatrix} R_y & & R_{IY} \\ - & & - \\ R_{IY} & & R_I \end{bmatrix} \begin{bmatrix} d_{K1} \\ \vdots \\ d_0 \\ \vdots \\ d_{K2} \end{bmatrix} = \begin{bmatrix} R_{IY}(-K1) \\ \vdots \\ R_{IY}(0) \\ R_I(1) \\ \vdots \\ R_I(K2) \end{bmatrix}$$

که برای اینکار از کد زیر استفاده میشود.

```
% DFE Equalizer design
function d = dfe_equalizer(R_y, R_IY, K2)
    A1 = zeros(K2, K2);
    A2 = zeros(K2, K2);
    A3 = diag(ones(K2, 1));
    for i = 1:K2
        A1(i,i) = R_y(5);
        if i+1 <= K2, A1(i,i+1) = R_y(4); end
        if i+2 <= K2, A1(i,i+2) = R_y(3); end
        if i+3 <= K2, A1(i,i+3) = R_y(2); end
        if i+4 <= K2, A1(i,i+4) = R_y(1); end
        if i-1 >= 1, A1(i,i-1) = R_y(6); end
        if i-2 >= 1, A1(i,i-2) = R_y(7); end
        if i-3 >= 1, A1(i,i-3) = R_y(8); end
        if i-4 >= 1, A1(i,i-4) = R_y(9); end
    end
    A2(K2, 1) = R_IY(2);
    A2(K2-1, 1) = R_IY(1);
    A2(K2, 2) = R_IY(1);
```

```
    b = zeros(2*K2, 1);
    b(K2) = R_IY(3); b(K2-1) = R_IY(2); b(K2-2) = R_IY(1);
    d = pinv([A1 A2; A2' A3]) * b;
end
```

Viterbi -1-3-4

برای مقایسه دقیق تر الگوریتم Viterbi که در تمرین قبل استفاده شد نیز در اینجا استفاده شده است.

```
% Viterbi Functions
function error_probabilities = simulate_viterbi_bpsk(SNR_dB_range, N, L)
    error_probabilities = zeros(size(SNR_dB_range));
    for i = 1:length(SNR_dB_range)
        SNR_dB = SNR_dB_range(i);
        total_errors = 0;
        total_bits = 0;

        while total_bits < N
            I_n = generate_bpsk_sequence(N);
            y_n = add_noise_bpsk(I_n, SNR_dB);
            decoded = viterbi_algorithm_bpsk(y_n, L);
            errors = sum(decoded ~= I_n);
            total_errors = total_errors + errors;
            total_bits = total_bits + length(I_n);
        end

        ber = total_errors / total_bits;
        error_probabilities(i) = ber;
        fprintf('BPSK SNR: %d dB, BER: %f\n', SNR_dB, ber);
    end
end
```

که برای Viterbi جداگانه مراحل تولید دیتا و نویزی کردن آن و دیکود کردن آن به صورت زیر انجام میشود (توضیحات بیشتر در گزارش تمرین 7 آمده است)

```
function I_n = generate_bpsk_sequence(N)
    % Generate random BPSK symbols
    I_n = randsrc(1, N, [1, -1]);
end

function y_n = add_noise_bpsk(I_n, SNR_dB)
    SNR_linear = 10^(SNR_dB / 10);
    sigma_n = sqrt(1 / SNR_linear);
    noise = sigma_n * randn(size(I_n));
    y_n = I_n + noise;
end

function decoded = viterbi_algorithm_bpsk(y_n, L)
```

```
    N = length(y_n);
    trellis = inf(2, N+1);
    trellis(:, 1) = 0; % Starting state with zero path metric
    path = zeros(2, N);

    states = [1, -1];
    for i = 2:N+1
        for curr_state = 1:2
            for prev_state = 1:2
                path_metric = trellis(prev_state, i-1) + (y_n(i-1) -
states(curr_state))^2;
                if path_metric < trellis(curr_state, i)
                    trellis(curr_state, i) = path_metric;
                    path(curr_state, i-1) = prev_state;
                end
            end
        end
    end

    decoded = zeros(1, N);
    [~, state] = min(trellis(:, N+1));
    for i = N:-1:1
        decoded(i) = states(state);
        state = path(state, i);
    end
end
```

5-3-1- حالت theorical

برای مقایسه بهتر یک احتمال خطای دیگر نیز داریم برای حالتی که اصلا ISI رخ ندهد که احتمال خطای آن به صورت زیر میشود.

```
% Theoretical BER for BPSK in AWGN
theoretical_ber = qfunc(sqrt(2 * SNR_lin));
```

۴-۱- کد اصلی

در کد اصلی ابتدا مقدار دهی اولیه داده میشود و ماتریس ها توابع خود همبستگی مورد نیاز ساخته شده و در ادمه محاسبات مربوط به الگوریتم های Viterbi و سایر اکوالایزر ها و .. انجام میشود و پس از به دست آوردن d هر کدام و سایر موارد مورد نیاز احتمال خطای هر یک و همچنین SINR بر حسب SNR به کمک توابع از قبل نوشته شده رسم میشود که کد نهایی به صورت زیر میباشد.

```
clc; clear; close all;

% Define Parameters
SNR_db = 0 : 20;                      % SNR in dB
SNR_lin = 10.^(SNR_db/10);            % Linear SNR
N = 1e5;                              % N
x = [0.0281 0.3437 0.75 0.3437 0.0281]'; % x(t)
xx = conv(x, x);                       % Auto-correlation of x(t)

% Theoretical BER for BPSK in AWGN
theoretical_ber = qfunc(sqrt(2 * SNR_lin));

% Preallocate BER and SINR arrays
ber_zf = zeros(3, length(SNR_lin));
sinr_zf = zeros(3, length(SNR_lin));
ber_mmse = zeros(3, length(SNR_lin));
sinr_mmse = zeros(3, length(SNR_lin));
ber_dfe = zeros(3, length(SNR_lin));
sinr_dfe = zeros(3, length(SNR_lin));

% Define Equalizer Lengths
K_values = [2, 4, 6];
K2_values = [3, 5, 7];

% Calculate Viterbi BER
error_probabilities_bpsk = simulate_viterbi_bpsk(SNR_db, 1e6, 6);

% Run Simulations for ZF, MMSE, and DFE Equalizers
for snr_idx = 1:length(SNR_lin)
    for k_idx = 1:length(K_values)
        % ZF Equalizer
        zf_eq = zf_equalizer(K_values(k_idx), x);
        [ber_zf(k_idx, snr_idx), sinr_zf(k_idx, snr_idx)] =
simulate_ber(SNR_lin(snr_idx), N, x, zf_eq);

        % MMSE Equalizer
        mmse_eq = mmse_equalizer(xx + [zeros(floor(length(x)/2), 1); x;
zeros(floor(length(x)/2), 1)]/SNR_lin(snr_idx), x, K_values(k_idx));
        [ber_mmse(k_idx, snr_idx), sinr_mmse(k_idx, snr_idx)] =
simulate_ber(SNR_lin(snr_idx), N, x, mmse_eq);

        % DFE Equalizer
```

```
        dfe_eq = dfe_equalizer(xx + [zeros(floor(length(x)/2), 1); x;
zeros(floor(length(x)/2), 1)]/SNR_lin(snr_idx), x, K2_values(k_idx));
        [ber_dfe(k_idx, snr_idx), sinr_dfe(k_idx, snr_idx)] =
simulate_ber_feedback(SNR_lin(snr_idx), N, x, dfe_eq);
    end
end

% Plot Results for ZF Equalizer
figure;
subplot(1, 2, 1);
hold on; grid on;
semilogy(SNR_db, ber_zf(1,:), '-o', 'DisplayName', 'ZF 5');
semilogy(SNR_db, ber_zf(2,:), '-x', 'DisplayName', 'ZF 9');
semilogy(SNR_db, ber_zf(3,:), '-s', 'DisplayName', 'ZF 13');
semilogy(SNR_db, theoretical_ber, '-.', 'DisplayName', 'Theoretical BER');
semilogy(SNR_db, error_probabilities_bpsk, 'o-', 'DisplayName', 'Viterbi BPSK');
xlabel('SNR (dB)');
ylabel('BER');
legend('show');
title('BER vs. SNR for ZF Equalizer');

subplot(1, 2, 2);
hold on; grid on;
semilogy(SNR_db, sinr_zf(1,:), '-o', 'DisplayName', 'ZF 5');
semilogy(SNR_db, sinr_zf(2,:), '-x', 'DisplayName', 'ZF 9');
semilogy(SNR_db, sinr_zf(3,:), '-s', 'DisplayName', 'ZF 13');
xlabel('SNR (dB)');
ylabel('SINR');
legend('show');
title('SINR vs. SNR for ZF Equalizer');

% Plot Results for MMSE Equalizer
figure;
subplot(1, 2, 1);
hold on; grid on;
semilogy(SNR_db, ber_mmse(1,:), '--o', 'DisplayName', 'MMSE 5');
semilogy(SNR_db, ber_mmse(2,:), '--x', 'DisplayName', 'MMSE 9');
semilogy(SNR_db, ber_mmse(3,:), '--s', 'DisplayName', 'MMSE 13');
semilogy(SNR_db, theoretical_ber, '-.', 'DisplayName', 'Theoretical BER');
semilogy(SNR_db, error_probabilities_bpsk, 'o-', 'DisplayName', 'Viterbi BPSK');
xlabel('SNR (dB)');
ylabel('BER');
legend('show');
title('BER vs. SNR for MMSE Equalizer');

subplot(1, 2, 2);
hold on; grid on;
semilogy(SNR_db, sinr_mmse(1,:), '--o', 'DisplayName', 'MMSE 5');
semilogy(SNR_db, sinr_mmse(2,:), '--x', 'DisplayName', 'MMSE 9');
semilogy(SNR_db, sinr_mmse(3,:), '--s', 'DisplayName', 'MMSE 13');
xlabel('SNR (dB)');
ylabel('SINR');
legend('show');
title('SINR vs. SNR for MMSE Equalizer');

% Plot Results for DFE Equalizer
figure;
subplot(1, 2, 1);
hold on; grid on;
```

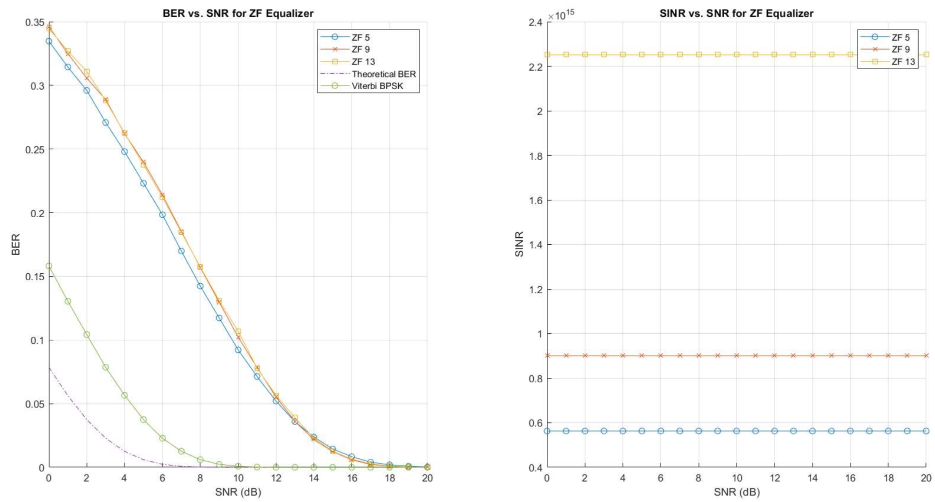
```
semilogy(SNR_db, ber_dfe(1,:), ':o', 'DisplayName', 'DFE 5');
semilogy(SNR_db, ber_dfe(2,:), ':x', 'DisplayName', 'DFE 9');
semilogy(SNR_db, ber_dfe(3,:), ':s', 'DisplayName', 'DFE 13');
semilogy(SNR_db, theoretical_ber, '-.', 'DisplayName', 'Theoretical BER');
semilogy(SNR_db, error_probabilities_bpsk, 'o-', 'DisplayName', 'Viterbi BPSK');
xlabel('SNR (dB)');
ylabel('BER');
legend('show');
title('BER vs. SNR for DFE Equalizer');

subplot(1, 2, 2);
hold on; grid on;
semilogy(SNR_db, sinr_dfe(1,:), ':o', 'DisplayName', 'DFE 5');
semilogy(SNR_db, sinr_dfe(2,:), ':x', 'DisplayName', 'DFE 9');
semilogy(SNR_db, sinr_dfe(3,:), ':s', 'DisplayName', 'DFE 13');
xlabel('SNR (dB)');
ylabel('SINR');
legend('show');
title('SINR vs. SNR for DFE Equalizer');

% Combined BER Plot for All Equalizers
figure;
hold on; grid on;
semilogy(SNR_db, ber_zf(2,:), '-x', 'DisplayName', 'ZF 9');
semilogy(SNR_db, ber_mmse(2,:), '--x', 'DisplayName', 'MMSE 9');
semilogy(SNR_db, ber_dfe(2,:), ':x', 'DisplayName', 'DFE 9');
semilogy(SNR_db, theoretical_ber, '-.', 'DisplayName', 'Theoretical BER');
semilogy(SNR_db, error_probabilities_bpsk, 'o-', 'DisplayName', 'Viterbi BPSK');
xlabel('SNR (dB)');
ylabel('BER');
legend('show');
title('BER vs. SNR for All Equalizers');

% Combined SINR Plot for All Equalizers
figure;
hold on; grid on;
semilogy(SNR_db, sinr_zf(2,:), '-x', 'DisplayName', 'ZF 9');
semilogy(SNR_db, sinr_mmse(2,:), '--x', 'DisplayName', 'MMSE 9');
semilogy(SNR_db, sinr_dfe(2,:), ':x', 'DisplayName', 'DFE 9');
xlabel('SNR (dB)');
ylabel('SINR');
legend('show');
title('SINR vs. SNR for All Equalizers');
```

در تمامی موارد برای مقایسه بهتر Viterbi و حالت بدون ISI در منحنی احتمال خطا ها میباشد و در انتها برای مقایسه نهایی همه احتمال خطا ها برای تمامی الگوریتم ها در یک منحنی نمایش داده میشود.

۵-۱- نتایج شبیه سازی

Zero-Forcing -1-5-1

نتایج به صورت زیر میباشد.



شکل (۴-۱) منحنیSINR و BER برای اکوالایزر ZF

همانطور که مشاهده میشود احتمال خطا بیشتر از حالت Viterbi و حالت theorical میباشد.

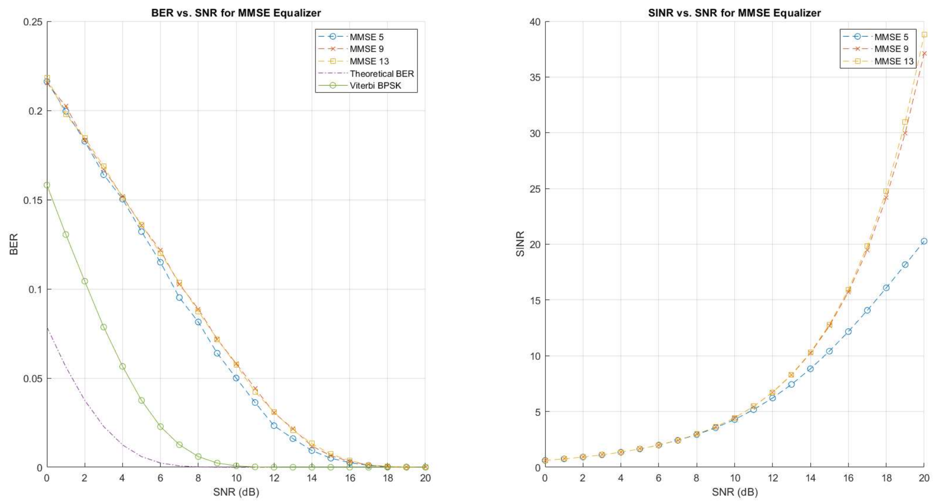
همچنین همانطور که از قبل میدانستیم ZF توجهی به نویز ندارد و به ازای هر SNR ای SINR آن ثابت میباشد.

MMSE -1-5-2

نتایج آن به صورت زیر میباشد.

همانطور که مشاهده میشود در SNR های بالا اوضاع بهتر میشود در SINR و همچنین احتمال خطا از ZF

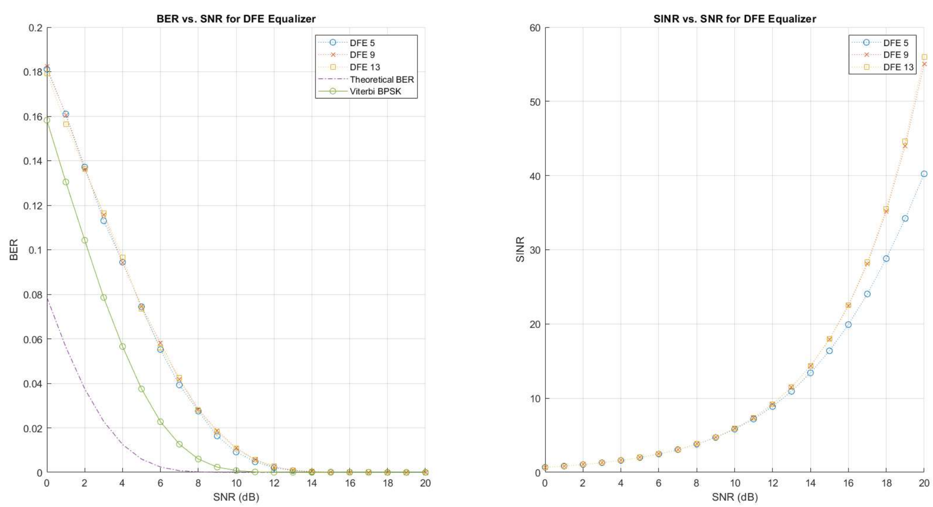
اندکی بهتر است (به خصوص در SNR های بالا) ولی همچنان بهینه ترین حالت همان Viterbi میباشد.



شکل (۵-۱) منحنیSINR و BER برای اکوالایزر MMSE

DFE -1-5-3

نتیج در شکل زیر قابل مشاهده میباشد.

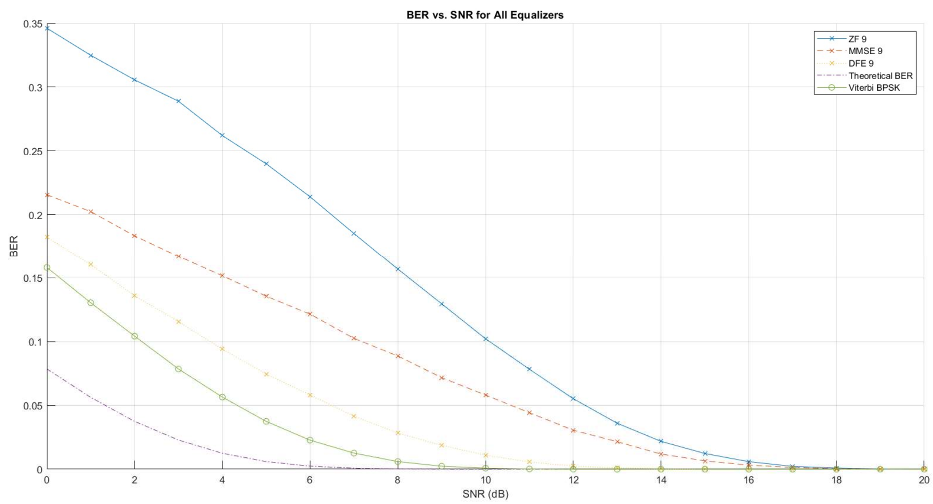


شکل (۶-۱) منحنی SINR و BER برای اکوالایزر DFE

همانطور که در شکل فوق میبینید و انتظار میرفت اوضاع بهتر است و احتمال خطا بهتر شده و نزدیک تر به Viterbi که حالت بهینه برای ISI میباشد است. همچنین SINR نیز در SNR های بالا ، بالا تر از حالت MMSE میباشد.

4-5-1- مقایسه همه حالت ها در کنار هم

شکل زیر BER بر حیب SNR همه حالت ها را در کنار هم نمایش میدهد.



شکل (۷-۱) منحنی BER بر حسب SNR برای همه حالت ها

همانطور که مشاهده میشود. به ترتیب بهترین عملکرد برای vitrbi سپس DFE و سپس MMSE و در نهایت ZF میباشد.

همچنین در منحنی های قبلی مشاهده شد که DFE بهترین عملکرد را در SINR دارد.

۶-۱- منابع

MATLAB/Simulink for Digital Communication by Won Young Yang et.al

https://github.com/himanshu-jaiswal/ZERO-FORCING-EQUALIZER

https://github.com/hodiepanh/Equalizer

https://github.com/momaee/advanced-communication