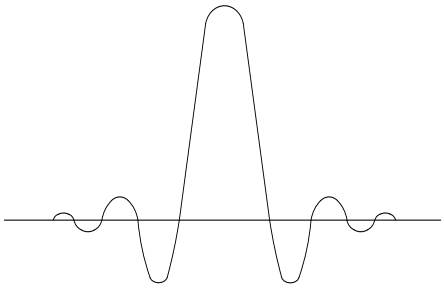
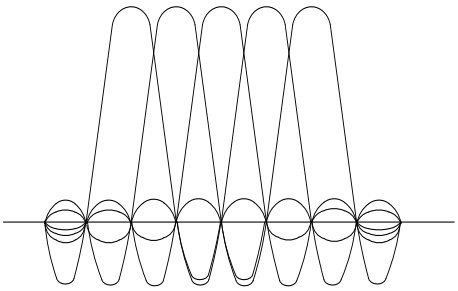
LAMPIRAN

Matematis OFDM

Seperti yang telah dijelaskan sebelumnya, OFDM mentransmisikan data dengan banyak subcarier yang sempit (*narrowband subcarrier*) dan saling overlaping dalam domain frekuensi. Untuk menghindari penggunaan jumlah filter dan modulator pada sisi penerima, maka dibutuhkan teknik pengolahan sinyal digital (DSP) modern yaitu menggunakan *Fast Fourier Transform* (FFT)



Gambar A.1 Spektrum Sinyal OFDM Single Subcarier



Gambar A.2 Spektrum Sinyal OFDM Lima Subcarier

Secara matematis, sinyal setiap carier dapat direpresentasikan :

$$Sc(t) = A0(t)e^{j[\omega nt + \phi_0(t)]} \qquad (1)$$

Bagian riil dari Sc(t) adalah sinyal itu sendiri. Ao(t) adalah amplitudo sinyal carier dan ϕo(t) adalah fasa dari sinyal carier dimana keduanya dapat bervariasi dalam suatu simbol. Nilai parameternya tetap sepanjang durasi waktu τ .

OFDM terdiri dari banyak carier. Sinyal kompleks Ss(t) direpresentasikan :

$$Ss(t) = \frac{1}{N}\sum_{n=0}^{N-1} An(t)e^{j[\omega_n t] + \phi_n(t)]} \qquad (2)$$

Dengan $\omega n = \omega_o + n\Delta\omega$

Hal ini berarti sinyal kompleks merupakan sinyal kontinyu. Jika bentuk gelombang setiap komponen sinyal dalam periode satu simbol, maka variabel Ao(t) dan ϕo(t) merupakan nilai yang tetap yang tergantung pada frekuensi carier tertentu sehingga bisa ditulis :

$\phi n(t) = \phi n$

$An(t) = An$

Jika sinyal disampling frekuensi 1/T, maka akan menghasilkan sinyal :

$$Ss(kT) = \frac{1}{N} \sum_{n=0}^{N-1} A_N e^{j(\omega 0 + n\Delta\omega)kT + \phi n]} \qquad (3)$$

Dari persamaan Ss(kT), maka pembatasan sinyal yang dianalisis sampai pada sinyal ke N, sehingga didapat :

$$\tau = NT$$

Bila persamaan (3) disederhanakan, tanpa *loss*, $\omega_0 = 0$, maka :

$$Ss(kT) = \frac{1}{N} \sum_{n=0}^{N-1} Ane^{j\phi n} e^{j(n\Delta\omega)kT} \qquad (4)$$

Persamaan (4) dapat dibandingkan dengan persamaan umum *Inverse Fourier Transform* (IFT) :

$$g(kT) = \frac{1}{N} \sum_{n=0}^{N-1} G\left(\frac{n}{NT}\right) e^{j2\pi nk/N} \qquad (5)$$

Persamaan (4) dan (5) ekuivalen bila :

$$\Delta f = \frac{\Delta\omega}{2\pi} = \frac{1}{NT} = \frac{1}{\tau} \qquad (6)$$

Kondisi yang sama inilah yang diperlukan untuk sifat orthogonalitas sinyal. Jadi, suatu sinyal OFDM dapat direpresentasikan dengan menggunakan prosedur transformasi Fourier. Pada transmiter dengan menggunakan prosedur *Inverse Fast Fourier Transform* (IFFT), sedangkan pada receiver menggunakan prosedur *Fast Fourier Transform* (FFT).

LAMPIRAN
ZERO FORCING ALGORITHM

Zero Forcing (ZF) dirancang untuk mendeteksi sinyal terima pada sistem multi antena *single carrier*, proses ZF seperti dijelaskan pada Bab 2, dilakukan pada masing-masing *subcarrier*.

$\hat{\underline{s}}_{sc-i} = \underline{\underline{H}}^{+}_{sc-i}.\underline{y}_{sc-i}$, $i = subcarrier$ ke-i

$$\hat{\underline{s}} = (\underline{\underline{H}}^{*}\,\underline{\underline{H}})^{-1}\underline{\underline{H}}^{*}.\underline{y} = \underline{\underline{H}}^{+}.\underline{y}$$

Dimana * *transpose conjugate*, + *pseudo inverse*. Agar *pseudo inverse* ada, maka N_t (jumlah antenna tranmit) harus lebih kecil atau sama dengan N_r (jumlah antenna receive). Jika tidak, proses matematis (H*H) bila di inverse tidak akan bisa.

$$H = \begin{bmatrix} -0.7i & 0.3-0.3i & -0.5-0.4i \\ 0.8-0.6i & 0.7-1.1i & -0.8-1.1i \\ -0.8 & 0.2+0.3i & 0.2i \\ -0.1-0.2i & 1.2-0.3i & -1.7-0.6i \end{bmatrix}$$

H merupakan matrik kanal berdimensi $N_t x N_r$, jika sinyal kirim x = [1+i -1-i 1+3i]^T dan vector noise n = [0.6+0.4i 0.4-0.1i 0.7+0.5i 0.2-0.2i]^T maka :

$$y = Hx + n = \begin{bmatrix} 1.4-2.2i \\ 2.5-3i \\ -0.6-0.6i \\ -1.1-7.1i \end{bmatrix}$$

Penghitungan H^+ :

$$H^{+} = \begin{bmatrix} -0.27+0.7i & -0.17+0.06i & -0.85+0.41i & 0.24-0.24i \\ 0.46+0.18i & 0.003+0.68i & 0.54+0.29i & 0.14-0.47i \\ 0.41-0.11i & 0.28+0.4i & 0.5+0.01i & -0.66-0.14i \end{bmatrix}$$

Dan matrik H^+H=I , matrik identitas.

$$H^{+}H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Penghitungan estimasi ZF adalah:

$$\underline{\hat{s}} = H^{+} y = \begin{bmatrix} -0.31+1.12i \\ -0.56-0.07i \\ 1.65+3.6i \end{bmatrix}$$

LAMPIRAN

Program

```
% INI PROGRAM UNTUK MIMO AJA !!!
clear all,clc

tic;
%
================================================================================
code_rate       = 3/4; % 1/2 , 2/3, 3/4
modulasi        = 64;  % [16 atau 64] QAM
interlv         = 2;   % [1]=Conv. Int, [2]=Block Int

Ntx=2; Nrx=2;
panjang_GI = 1/16;
Nfft=64;
jml_subcar=48;
EbNo_max = 20;
EbNo=0:2:EbNo_max;

% ***************************** Pengirim ******************************
% Data
if modulasi == 16
    if code_rate == 1/2
        jml_itr = 520;
        data=randint(1,3828,2);
    elseif code_rate == 2/3
        jml_itr = 400;
        data=randint(1,5104,2);
    elseif code_rate == 3/4
        jml_itr = 350;
        data=randint(1,5742,2);
    end
elseif modulasi == 64
    if code_rate == 1/2
        jml_itr = 520;
        data=randint(1,5748,2);
    elseif code_rate == 2/3
        jml_itr = 400;
        data=randint(1,7664,2);
    elseif code_rate == 3/4
        jml_itr = 350;
        data=randint(1,8622,2);
    end
end
jml_itr = 2*jml_itr;

% S to P untuk MIMO
pecah = length(data)/Ntx;
for mimo = 0:Ntx - 1
    datane = data(mimo*pecah+1:(1+mimo)*pecah);

% CC Encoder
    if code_rate == 1/2
        % datane = 1914 - 2874
        gen_pol=poly2trellis(7,[171 133]);
        codeword=convenc(datane,gen_pol); % 3828 - 5748
    elseif code_rate == 2/3
        % datane = 2552 - 3832
        gen_pol=poly2trellis(7,[171 133]);
        codeword=convenc(datane,gen_pol); % 5104 - 7664
        codeword(4:4:end)=[]; % Puncture by removing every third value. %
3828 - 5748
    elseif code_rate == 3/4
        % datane = 2871 - 4311
        gen_pol=poly2trellis(7,[171 133]);
        codeword=convenc(datane,gen_pol); % 5742 - 8622
```

```
        codeword(3:3:end)=[]; % Puncture by removing every third value. %
3828 - 5748
    end

    % Interleaver
    if interlv == 1 % Convolutional
        nrows = 3; slope = 2; % Interleaver parameters
        D = nrows*(nrows-1)*slope; % Delay of interleaver/deinterleaver
pair
        codeword_padded = [codeword.'; zeros(D,1)]; % Pad x at the end
before interleaving.
        codeword2 = convintrlv(codeword_padded,nrows,slope); % Interleave
padded data.
        out_int = [-2*codeword2 + 1].'; % 3840 - 5760
    elseif interlv == 2 % Block
        codeword2 = [codeword zeros(1,12)];
        A = 16;
        B = length(codeword2)/A;
        for n=0:length(codeword2)/B - 1
            int_sym(n+1,1:B)=codeword2(1,n*B+1:(n+1)*B);
            n=n+1;
        end
        [p q]=size(int_sym);
        out_int = -2*reshape(int_sym,1,p*q) + 1; % 3840 - 5760
    end
    out_int_tot(mimo+1,:)=out_int;

    if modulasi == 16
        % Modulasi 16QAM
        jml_bit_simbol = 4;
        pjg_data=length(out_int);
        for p = 1:pjg_data/jml_bit_simbol;
            if     out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 -1 -1]
                in_map(p) = 0;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 -1 1]
                in_map(p) = 1;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 1 -1]
                in_map(p) = 2;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 1 1]
                in_map(p) = 3;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 1 -1 -1]
                in_map(p) = 4;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 1 -1 1]
                in_map(p) = 5;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 1 1 -1]
                in_map(p) = 6;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 1 1 1]
                in_map(p) = 7;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[1
-1 -1 -1]
                in_map(p) = 8;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[1
-1 -1 1]
                in_map(p) = 9;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[1
-1 1 -1]
```

```
                in_map(p) = 10;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[1
-1 1 1]
                in_map(p) = 11;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[1
1 -1 -1]
                in_map(p) = 12;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[1
1 -1 1]
                in_map(p) = 13;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[1
1 1 -1]
                in_map(p) = 14;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[1
1 1 1]
                in_map(p) = 15;
            end
        end
        data_mod=qammod(in_map,16); % 960
    elseif modulasi == 64
        % Modulasi 64QAM
        jml_bit_simbol = 6;
        pjg_data=length(out_int);
        for p = 1:pjg_data/jml_bit_simbol;
            if     out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 -1 -1 -1 -1]
                in_map(p) = 0;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 -1 -1 -1  1]
                in_map(p) = 1;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 -1 -1  1 -1]
                in_map(p) = 2;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 -1 -1  1  1]
                in_map(p) = 3;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 -1  1 -1 -1]
                in_map(p) = 4;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 -1  1 -1  1]
                in_map(p) = 5;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 -1  1  1 -1]
                in_map(p) = 6;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1 -1  1  1  1]
                in_map(p) = 7;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1  1 -1 -1 -1]
                in_map(p) = 8;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1  1 -1 -1  1]
                in_map(p) = 9;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1  1 -1  1 -1]
                in_map(p) = 10;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1  1 -1  1  1]
                in_map(p) = 11;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1  1  1 -1 -1]
                in_map(p) = 12;
```

```
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1  1  1 -1  1]
                in_map(p) = 13;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1  1  1  1 -1]
                in_map(p) = 14;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1 -1  1  1  1  1]
                in_map(p) = 15;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1 -1 -1 -1 -1]
                in_map(p) = 16;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1 -1 -1 -1  1]
                in_map(p) = 17;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1 -1 -1  1 -1]
                in_map(p) = 18;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1 -1 -1  1  1]
                in_map(p) = 19;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1 -1  1 -1 -1]
                in_map(p) = 20;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1 -1  1 -1  1]
                in_map(p) = 21;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1 -1  1  1 -1]
                in_map(p) = 22;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1 -1  1  1  1]
                in_map(p) = 23;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1  1 -1 -1 -1]
                in_map(p) = 24;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1  1 -1 -1  1]
                in_map(p) = 25;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1  1 -1  1 -1]
                in_map(p) = 26;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1  1 -1  1  1]
                in_map(p) = 27;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1  1  1 -1 -1]
                in_map(p) = 28;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1  1  1 -1  1]
                in_map(p) = 29;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1  1  1  1 -1]
                in_map(p) = 30;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[-
1  1  1  1  1  1]
                in_map(p) = 31;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1 -1 -1 -1 -1]
                in_map(p) = 32;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1 -1 -1 -1  1]
                in_map(p) = 33;
```

```
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1 -1 -1  1 -1]
                in_map(p) = 34;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1 -1 -1  1  1]
                in_map(p) = 35;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1 -1  1 -1 -1]
                in_map(p) = 36;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1 -1  1 -1  1]
                in_map(p) = 37;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1 -1  1  1 -1]
                in_map(p) = 38;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1 -1  1  1  1]
                in_map(p) = 39;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1  1 -1 -1 -1]
                in_map(p) = 40;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1  1 -1 -1  1]
                in_map(p) = 41;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1  1 -1  1 -1]
                in_map(p) = 42;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1  1 -1  1  1]
                in_map(p) = 43;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1  1  1 -1 -1]
                in_map(p) = 44;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1  1  1 -1  1]
                in_map(p) = 45;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1  1  1  1 -1]
                in_map(p) = 46;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1 -1  1  1  1  1]
                in_map(p) = 47;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1 -1 -1 -1 -1]
                in_map(p) = 48;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1 -1 -1 -1  1]
                in_map(p) = 49;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1 -1 -1  1 -1]
                in_map(p) = 50;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1 -1 -1  1  1]
                in_map(p) = 51;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1 -1  1 -1 -1]
                in_map(p) = 52;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1 -1  1 -1  1]
                in_map(p) = 53;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1 -1  1  1 -1]
                in_map(p) = 54;
```

```
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1 -1  1  1  1]
                in_map(p) = 55;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1  1 -1 -1 -1]
                in_map(p) = 56;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1  1 -1 -1  1]
                in_map(p) = 57;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1  1 -1  1 -1]
                in_map(p) = 58;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1  1 -1  1  1]
                in_map(p) = 59;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1  1  1 -1 -1]
                in_map(p) = 60;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1  1  1 -1  1]
                in_map(p) = 61;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1  1  1  1 -1]
                in_map(p) = 62;
            elseif out_int(1,jml_bit_simbol*(p-1)+1:jml_bit_simbol*p)==[
1  1  1  1  1  1]
                in_map(p) = 63;
            end
        end
        data_mod=qammod(in_map,64); % 960
    end

    % S to P untuk OFDM
    a = length(data_mod);
    b = jml_subcar;
    c = a/b;
    for p = 0:b-1
        out_sp(p+1,:)=data_mod(p*c+1:c*(p+1));
    end

    % Penambahan Pilot dan Zero Padding
    dataout_pilot_zp=zeros(Nfft,c); % 12 Zero Padding ada di sini
    dataout_pilot_zp(2:7,:)   = out_sp(1:6,:);
    dataout_pilot_zp(8,:)     = repmat(1,1,c);
    dataout_pilot_zp(9:21,:)  = out_sp(7:19,:);
    dataout_pilot_zp(22,:)    = repmat(-1,1,c);
    dataout_pilot_zp(23:27,:) = out_sp(20:24,:);
    dataout_pilot_zp(39:42,:) = out_sp(25:28,:);
    dataout_pilot_zp(43,:)    = repmat(1,1,c);
    dataout_pilot_zp(44:56,:) = out_sp(29:41,:);
    dataout_pilot_zp(57,:)    = repmat(-1,1,c);
    dataout_pilot_zp(58:64,:) = out_sp(42:48,:);

    % IFFT
    data_ifft=ifft(dataout_pilot_zp,64);

    % Penambahan Guard Interval
    [g,h]=size(data_ifft);
    simbol_GI=floor(panjang_GI*h);
    data_ifft_insert_GI=[data_ifft(:,h-(simbol_GI-1):end) data_ifft];
    [g,ukuran_frame]=size(data_ifft_insert_GI);

    % Paralel to Serial
    out_ifft_serial=[];
```

```
    for k=1:ukuran_frame
        data_x=data_ifft_insert_GI(:,k).';
        out_ifft_serial=[out_ifft_serial data_x];
    end
    input_kanal(mimo+1,:)=out_ifft_serial;
end

input_kanal_gsr=in_kanal_proses(input_kanal); % Pendelayan sinyal
% Pembentukan matrik "input_kanal"
sinyal_ant_1 = input_kanal_gsr(:,:,1);
sinyal_ant_2 = input_kanal_gsr(:,:,2);
for P = 1:length(input_kanal_gsr)
    input_kanal_gab(:,P)=[sinyal_ant_1(:,P); sinyal_ant_2(:,P)];
end

% Sinyal Training
train = repmat([1+i -1+i; 1-i -1-i],1,length(input_kanal_gab));

% Iterasinya
for jum=1:jml_itr

    H=kanal_transmisi(data_ifft_insert_GI,Ntx*Nrx); % Pembangkitan
koefisien kanal
    % Pembentukan matrik kanal MIMO
    H_1 = H(:,:,1); H_3 = H(:,:,3);
    H_2 = H(:,:,2); H_4 = H(:,:,4);
    for Q = 1:length(H)
        H_gab(:,:,Q)=[H_1(Q,:) H_3(Q,:);...
                      H_2(Q,:) H_4(Q,:)];
        H_(:,:,Q)=[sum(H_1(Q,:)) sum(H_3(Q,:));...
                      sum(H_2(Q,:)) sum(H_4(Q,:))];
    end

    % Perkalian "train_gab" dan "H_gab"
    for Z = 0:length(train)/2 - 1
        HStrain(:,:,Z+1)=H_(:,:,Z+1)*train(:,Z*2+1:(Z+1)*2);
    end

    % Perkalian "input_kanal_gab" dengan "H_gab"
    for Z = 1:length(input_kanal_gab)
        HS(:,Z)=H_gab(:,:,Z)*input_kanal_gab(:,Z);
    end

    % EbNo
    for energi = 1:length(EbNo)

        % Terkena AWGN
        for D = 1:length(HS)
            HS_train=HStrain(:,:,D);
            No = noisenya(modulasi,EbNo(energi)); rata =
mean(mean(abs(HS_train)));
            noise = rata*(randn(size(HS_train,1),size(HS_train,2)) +
i*randn(size(HS_train,1),size(HS_train,2)))*No;
            sinyal_train_terima(:,:,D) = HS_train + noise;
            %
sinyal_train_terima(:,:,D)=awgn(HStrain(:,:,D),EbNo(energi),'measured');
% Noise Addition
        end
        % for D = 1:length(HS)
        %
sinyal_train_terima(:,:,D)=awgn(HStrain(:,:,D),EbNo(energi),'measured');
% Noise Addition
        % end
        No = noisenya(modulasi,EbNo(energi)); rata = mean(mean(abs(HS)));
```

```
        noise = rata*[randn(size(HS,1),size(HS,2)) +
i*randn(size(HS,1),size(HS,2))]*No;
        sinyal_terima1 = HS + noise;
        % sinyal_terima1=awgn(HS,EbNo(energi),'measured'); % Noise
Addition

        % Estimasi Kanal
        for p = 0:length(HS)-1

H_det(:,:,p+1)=sinyal_train_terima(:,:,p+1)*inv(train(:,p*2+1:(p+1)*2));
        end

        % MIMO Detection [ML]
        for q = 1:length(sinyal_terima1)
            out_det(:,q)=pinv(H_det(:,:,q))*sinyal_terima1(:,q);
        end

        % ************************* Penerima ***************************
        for mimo = 1:Ntx
            sinyal_terima2=out_det(mimo,:);

            % serial to paralel OFDM
            for p = 0:size(data_ifft_insert_GI,2)-1
                sinyal_terima3(p+1,:) =
sinyal_terima2(1,64*p+1:64*(p+1));
            end
            sinyal_terima=sinyal_terima3.';

            % Remove Guard Interval
            sinyal_terima(:,1:simbol_GI)=[];

            % FFT
            data_fft=fft(sinyal_terima,64);

            % Remove Pilot dan Zero Padding
            rem_pilotzp(1:6,:) = data_fft(2:7,:);
            rem_pilotzp(7:19,:) = data_fft(9:21,:);
            rem_pilotzp(20:24,:) = data_fft(23:27,:);
            rem_pilotzp(25:28,:) = data_fft(39:42,:);
            rem_pilotzp(29:41,:) = data_fft(44:56,:);
            rem_pilotzp(42:48,:) = data_fft(58:64,:);

            % P to S untuk de-OFDM
            for r = 0:jml_subcar-1

out_ps(r*size(data_ifft,2)+1:size(data_ifft,2)*(r+1))=rem_pilotzp(r+1,:);
            end
            out_ps2 = out_ps.';

            if modulasi == 16
                % Demodulasi 16QAM
                out_demap = qamdemod(out_ps2,16);
                for q = 1:length(out_demap)
                    if     out_demap(q) == 0
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [-1 -1 -1 -1];
                    elseif out_demap(q) == 1
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [-1 -1 -1 1];
                    elseif out_demap(q) == 2
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [-1 -1 1 -1];
                    elseif out_demap(q) == 3
```

```
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [-1 -1 1 1];
                    elseif out_demap(q) == 4
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [-1 1 -1 -1];
                    elseif out_demap(q) == 5
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [-1 1 -1 1];
                    elseif out_demap(q) == 6
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [-1 1 1 -1];
                    elseif out_demap(q) == 7
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [-1 1 1 1];
                    elseif out_demap(q) == 8
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [1 -1 -1 -1];
                    elseif out_demap(q) == 9
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [1 -1 -1 1];
                    elseif out_demap(q) == 10
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [1 -1 1 -1];
                    elseif out_demap(q) == 11
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [1 -1 1 1];
                    elseif out_demap(q) == 12
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [1 1 -1 -1];
                    elseif out_demap(q) == 13
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [1 1 -1 1];
                    elseif out_demap(q) == 14
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [1 1 1 -1];
                    elseif out_demap(q) == 15
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)= [1 1 1 1];
                    end
                end
            elseif modulasi == 64
                % Demodulasi 64QAM
                out_demap = qamdemod(out_ps2,64);
                for q = 1:length(out_demap)
                    if     out_demap(q) == 0
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1 -1 -1 -1 -1];
                    elseif out_demap(q) == 1
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1 -1 -1 -1  1];
                    elseif out_demap(q) == 2
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1 -1 -1  1 -1];
                    elseif out_demap(q) == 3
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1 -1 -1  1  1];
                    elseif out_demap(q) == 4
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1 -1  1 -1 -1];
                    elseif out_demap(q) == 5
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1 -1  1 -1  1];
                    elseif out_demap(q) == 6
```

```
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1 -1  1  1 -1];
                    elseif out_demap(q) == 7
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1 -1  1  1  1];
                    elseif out_demap(q) == 8
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1  1 -1 -1 -1];
                    elseif out_demap(q) == 9
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1  1 -1 -1  1];
                    elseif out_demap(q) == 10
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1  1 -1  1 -1];
                    elseif out_demap(q) == 11
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1  1 -1  1  1];
                    elseif out_demap(q) == 12
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1  1  1 -1 -1];
                    elseif out_demap(q) == 13
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1  1  1 -1  1];
                    elseif out_demap(q) == 14
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1  1  1  1 -1];
                    elseif out_demap(q) == 15
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1 -1  1  1  1  1];
                    elseif out_demap(q) == 16
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1 -1 -1 -1 -1];
                    elseif out_demap(q) == 17
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1 -1 -1 -1  1];
                    elseif out_demap(q) == 18
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1 -1 -1  1 -1];
                    elseif out_demap(q) == 19
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1 -1 -1  1  1];
                    elseif out_demap(q) == 20
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1 -1  1 -1 -1];
                    elseif out_demap(q) == 21
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1 -1  1 -1  1];
                    elseif out_demap(q) == 22
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1 -1  1  1 -1];
                    elseif out_demap(q) == 23
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1 -1  1  1  1];
                    elseif out_demap(q) == 24
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1  1 -1 -1 -1];
                    elseif out_demap(q) == 25
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1  1 -1 -1  1];
                    elseif out_demap(q) == 26
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1  1 -1  1 -1];
                    elseif out_demap(q) == 27
```

```
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1  1 -1  1  1];
                    elseif out_demap(q) == 28
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1  1  1 -1 -1];
                    elseif out_demap(q) == 29
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1  1  1 -1  1];
                    elseif out_demap(q) == 30
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1  1  1  1 -1];
                    elseif out_demap(q) == 31
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[-1  1  1  1  1  1];
                    elseif out_demap(q) == 32
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1 -1 -1 -1 -1];
                    elseif out_demap(q) == 33
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1 -1 -1 -1  1];
                    elseif out_demap(q) == 34
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1 -1 -1  1 -1];
                    elseif out_demap(q) == 35
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1 -1 -1  1  1];
                    elseif out_demap(q) == 36
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1 -1  1 -1 -1];
                    elseif out_demap(q) == 37
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1 -1  1 -1  1];
                    elseif out_demap(q) == 38
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1 -1  1  1 -1];
                    elseif out_demap(q) == 39
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1 -1  1  1  1];
                    elseif out_demap(q) == 40
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1  1 -1 -1 -1];
                    elseif out_demap(q) == 41
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1  1 -1 -1  1];
                    elseif out_demap(q) == 42
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1  1 -1  1 -1];
                    elseif out_demap(q) == 43
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1  1 -1  1  1];
                    elseif out_demap(q) == 44
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1  1  1 -1 -1];
                    elseif out_demap(q) == 45
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1  1  1 -1  1];
                    elseif out_demap(q) == 46
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1  1  1  1 -1];
                    elseif out_demap(q) == 47
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1 -1  1  1  1  1];
                    elseif out_demap(q) == 48
```

```
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1 -1 -1 -1 -1];
                    elseif out_demap(q) == 49
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1 -1 -1 -1  1];
                    elseif out_demap(q) == 50
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1 -1 -1  1 -1];
                    elseif out_demap(q) == 51
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1 -1 -1  1  1];
                    elseif out_demap(q) == 52
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1 -1  1 -1 -1];
                    elseif out_demap(q) == 53
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1 -1  1 -1  1];
                    elseif out_demap(q) == 54
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1 -1  1  1 -1];
                    elseif out_demap(q) == 55
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1 -1  1  1  1];
                    elseif out_demap(q) == 56
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1  1 -1 -1 -1];
                    elseif out_demap(q) == 57
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1  1 -1 -1  1];
                    elseif out_demap(q) == 58
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1  1 -1  1 -1];
                    elseif out_demap(q) == 59
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1  1 -1  1  1];
                    elseif out_demap(q) == 60
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1  1  1 -1 -1];
                    elseif out_demap(q) == 61
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1  1  1 -1  1];
                    elseif out_demap(q) == 62
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1  1  1  1 -1];
                    elseif out_demap(q) == 63
                        out_demod(1,jml_bit_simbol*(q-
1)+1:jml_bit_simbol*q)=[ 1  1  1  1  1  1];
                    end
                end
            end

            % DeInterleaver
            if interlv == 1 % Convolutional
                P = out_demod.';
                de_int = convdeintrlv(P,nrows,slope);
                out_de_int = de_int(D+1:end); % Remove first D symbols.
                out_de_int = out_de_int.';
            elseif interlv == 2 % Block
                P = out_demod;
                for n=0:length(P)/A - 1
                    de_int(n+1,1:A)=P(1,n*A+1:(n+1)*A);
                    n=n+1;
                end
                [t u]=size(de_int);
```

```
                out_deint = reshape(de_int,1,t*u);
                out_de_int=out_deint(1:end-12);
            end

            % CC Decoder
            if code_rate == 1/2
                %out_de_int = (out_de_int-1)/(-2);
                datane_topi(mimo,:) =
vitdec(out_de_int,gen_pol,96,'trunc','unquant'); % Decode.
            elseif code_rate == 2/3
                out_decod_conv = zeros(1,2*length(datane)); % Zeros
represent inserted data.
                out_decod_conv(1:4:end) = out_de_int(1:3:end); % Write
actual data.
                out_decod_conv(2:4:end) = out_de_int(2:3:end); % Write
actual data.
                out_decod_conv(3:4:end) = out_de_int(3:3:end); % Write
actual data.
                datane_topi(mimo,:)=
vitdec(out_decod_conv,gen_pol,96,'trunc','unquant'); % Decode.
            elseif code_rate == 3/4
                out_decod_conv = zeros(1,2*length(datane)); % Zeros
represent inserted data.
                out_decod_conv(1:3:end) = out_de_int(1:2:end); % Write
actual data.
                out_decod_conv(2:3:end) = out_de_int(2:2:end); % Write
actual data.
                datane_topi(mimo,:)=
vitdec(out_decod_conv,gen_pol,96,'trunc','unquant'); % Decode.
            end
            out_demod_tot(mimo,:)=out_demod;
        end

        % P to S
        data_topi = [datane_topi(1,:) datane_topi(2,:)];

        % Perhitungan Bit Salah
        bit_error(1,energi) = sum(data~=data_topi);
        cw_error(1,energi)  = sum(sum(out_int_tot~=out_demod_tot));

        Looping_EbNo_BitError = [jum EbNo(energi) bit_error(1,energi)
cw_error(1,energi)]
        if EbNo(energi) == max(EbNo)
            display('*****************************');
            display('*****************************');
        end

    end

    total_bit_error(jum,:) = bit_error;
    total_cw_error(jum,:)  = cw_error;

end

pe_bit = sum(total_bit_error)/(length(data)*jml_itr);
pe_cw  = sum(total_cw_error)/(length(out_int)*jml_itr);
figure;
semilogy(EbNo,pe_bit,'b:*'); hold on
semilogy(EbNo,pe_cw,'r:*'); grid
toc
waktu_jam=toc/3600;

save Rayl_2x2_64_3per4 pe_bit pe_cw EbNo waktu_jam
```