LAMPIRAN

FUNGSI M-FILE MATLAB

```
%-------------------------------------------------------------------%
%-------------------------- algoritma BCJR -------------------------%
%-------------------------------------------------------------------%

function [aposteriori_uncoded_llrs, aposteriori_encoded1_llrs,
aposteriori_encoded2_llrs] = bcjr_decoder(apriori_uncoded_llrs,
apriori_encoded1_llrs, apriori_encoded2_llrs)



if(length(apriori_uncoded_llrs) ~= length(apriori_encoded1_llrs) ||
length(apriori_encoded1_llrs) ~= length(apriori_encoded2_llrs))
    error('LLR sequences must have the same length');
end;


% Matrik untuk menggambarkan trellis %
% Tiap baris menggambarkan suatu transisi dari trelis %
% Tiap state dialokasikan pada indeks 1,2,3 ... %
%             FromState, ToState, UncodedBit, Encoded1Bit,Ecoded2Bit
transitions = [1,           1,           0,           0,      0;
               1,           2,           1,           1,      1;
               2,           1,           1,           1,      0;
               2,           2,           0,           0,      1];

% Cari State terbesar pada matrik transisi %
state_count = max(max(transitions(:,1)),max(transitions(:,2)));

% Hitung apriori transisi log-confidences dengan menjumlahkan
% log-confidences yang disatukan dengan tiap nilai bit yang
dituliskan %

gammas=zeros(size(transitions,1),length(apriori_uncoded_llrs));

for bit_index = 1:length(apriori_uncoded_llrs)
    for transition_index = 1:size(transitions,1)
        if transitions(transition_index, 3)==0
            gammas(transition_index, bit_index) =
gammas(transition_index, bit_index) +
apriori_uncoded_llrs(bit_index)/2;

            else
            gammas(transition_index, bit_index) =
gammas(transition_index, bit_index) -
apriori_uncoded_llrs(bit_index)/2;
        end;
        if transitions(transition_index, 4)==0
```

```
            gammas(transition_index, bit_index) =
gammas(transition_index, bit_index) +
apriori_encoded1_llrs(bit_index)/2;
        else
            gammas(transition_index, bit_index) =
gammas(transition_index, bit_index) -
apriori_encoded1_llrs(bit_index)/2;
        end;
        if transitions(transition_index, 5)==0
            gammas(transition_index, bit_index) =
gammas(transition_index, bit_index) +
apriori_encoded2_llrs(bit_index)/2;
        else
            gammas(transition_index, bit_index) =
gammas(transition_index, bit_index) -
apriori_encoded2_llrs(bit_index)/2;
        end;
    end;
end;

% bagian Forward recursive %
alphas=zeros(state_count,length(apriori_uncoded_llrs));
alphas=alphas-inf;
alphas(1,1)=0; % merupakan awal state %

for state_index = 2:state_count
    alphas(state_index,1)=-inf; % merupakan state selanjutnya %
end;

for bit_index = 2:length(apriori_uncoded_llrs)
    for transition_index = 1:size(transitions,1)
        alphas(transitions(transition_index,2),bit_index) =
jac(alphas(transitions(transition_index,2),bit_index),alphas(transit
ions(transition_index,1),bit_index-1) + gammas(transition_index,
bit_index-1));
    end;
end;

% Bagian Backwards recursive %
betas=zeros(state_count,length(apriori_uncoded_llrs));
betas=betas-inf;

for state_index = 1:state_count
    betas(state_index,length(apriori_uncoded_llrs))=0; % bagian
State Terakhir %
end;

for bit_index = length(apriori_uncoded_llrs)-1:-1:1
    for transition_index = 1:size(transitions,1)
        betas(transitions(transition_index,1),bit_index) =
jac(betas(transitions(transition_index,1),bit_index),betas(transitio
```

```
ns(transition_index,2),bit_index+1) + gammas(transition_index,
bit_index+1));
    end;
end;

% Menghitung aposteriori llrs transisi  %
deltas=zeros(size(transitions,1),length(apriori_uncoded_llrs));

for bit_index = 1:length(apriori_uncoded_llrs)
    for transition_index = 1:size(transitions,1)
        deltas(transition_index, bit_index) =
alphas(transitions(transition_index,1),bit_index) +
gammas(transition_index, bit_index) +
betas(transitions(transition_index,2),bit_index);
    end;
end;


aposteriori_uncoded_llrs = zeros(1,length(apriori_uncoded_llrs));

for bit_index = 1:length(apriori_uncoded_llrs)
    prob0=-inf;
    prob1=-inf;
    for transition_index = 1:size(transitions,1)
        if transitions(transition_index,3)==0
            prob0 = jac(prob0, deltas(transition_index,bit_index));
        else
            prob1 = jac(prob1, deltas(transition_index,bit_index));
        end;
    end;
    aposteriori_uncoded_llrs(bit_index) = prob0-prob1;
end;

aposteriori_encoded1_llrs = zeros(1,length(apriori_uncoded_llrs));

for bit_index = 1:length(apriori_uncoded_llrs)
    prob0=-inf;
    prob1=-inf;
    for transition_index = 1:size(transitions,1)
        if transitions(transition_index,4)==0
            prob0 = jac(prob0, deltas(transition_index,bit_index));
        else
            prob1 = jac(prob1, deltas(transition_index,bit_index));
        end;
    end;
    aposteriori_encoded1_llrs(bit_index) = prob0-prob1;
end;

aposteriori_encoded2_llrs = zeros(1,length(apriori_uncoded_llrs));

for bit_index = 1:length(apriori_uncoded_llrs)
```

```
    prob0=-inf;
    prob1=-inf;
    for transition_index = 1:size(transitions,1)
        if transitions(transition_index,5)==0
            prob0 = jac(prob0, deltas(transition_index,bit_index));
        else
            prob1 = jac(prob1, deltas(transition_index,bit_index));
        end;
    end;
    aposteriori_encoded2_llrs(bit_index) = prob0-prob1;
end;
```

```
%-------------------------------------------------------------------------%
%------------------- mengkodekan dengan Konvolusi ------------------------%
%-------------------------------------------------------------------------%


function [encoded1_bits, encoded2_bits] =
convolutional_encoder(uncoded_bits)

    % Bit Sistematik %
    encoded1_bits = uncoded_bits;

    % Bit Parity %
    encoded2_bits=zeros(1,length(uncoded_bits));
    encoded2_bits(1) = uncoded_bits(1);
    for i = 2:length(uncoded_bits)
        encoded2_bits(i) = mod(encoded2_bits(i-
1)+uncoded_bits(i),2);
    end
```

```
%-------------------------------------------------------------------%
%---------------- Membangkitkan LogLikelihood Ratio ----------------%
%-------------------------------------------------------------------%


function llrs = generate_llrs(bits, mutual_information)
    if(mutual_information < 0 || mutual_information >= 1)
        error('mutual_information must be in the range [0,1)');
    end




%-------------------------------------------------------------------%
%----------------- Memastikan Mutual informasi Rata-rata ----------%
%-------------------------------------------------------------------%



function mutual_information =
measure_mutual_information_averaging(llrs)
    P0 = exp(llrs)./(1+exp(llrs));
    P1 = 1-P0;
    entropies = -P0.*log2(P0)-P1.*log2(P1);
    mutual_information = 1-
sum(entropies(~isnan(entropies)))/length(entropies);




%-------------------------------------------------------------------%
%-----------------Memastikan histogram mutual informasi------------%
%-------------------------------------------------------------------%


function mutual_information =
measure_mutual_information_histogram(llrs, bits)


    if(length(llrs) ~= length(bits))
        error('Must have same number of llrs and bits!');
    end

        if(llr_0_noninfinite_count > 0 && llr_1_noninfinite_count >
0 && llr_0_min <= llr_1_max && llr_1_min <= llr_0_max)
            llr_0_mean = 0.0;
            llr_1_mean = 0.0;
            for bit_index = 1:length(bits)
```

```
                if(llrs(bit_index) ~= -Inf && llrs(bit_index) ~=
Inf)
                    if(bits(bit_index) == 0)
                        llr_0_mean = llr_0_mean+llrs(bit_index);
                    else
                        llr_1_mean = llr_1_mean+llrs(bit_index);
                    end
                end

            end
            llr_0_mean = llr_0_mean/llr_0_noninfinite_count;
            llr_1_mean = llr_1_mean/llr_1_noninfinite_count;

            llr_0_variance = 0.0;
            llr_1_variance = 0.0;
            for bit_index = 1:length(bits)
                if(llrs(bit_index) ~= -Inf && llrs(bit_index) ~=
Inf)

                    if(bits(bit_index) == 0)

                        llr_0_variance = llr_0_variance +
(llrs(bit_index) - llr_0_mean)^2;
                    else

                        llr_1_variance = llr_1_variance +
(llrs(bit_index) - llr_1_mean)^2;
                    end
                end
            end
            llr_0_variance = llr_0_variance/llr_0_noninfinite_count;
            llr_1_variance = llr_1_variance/llr_1_noninfinite_count;

            bin_width =
0.5*(3.49*sqrt(llr_0_variance)*(llr_0_noninfinite_count^(-1.0/3.0))
+ 3.49*sqrt(llr_1_variance)*(llr_1_noninfinite_count^(-1.0/3.0)));
            if(bin_width > 0.0)

                bin_offset = floor(min(llr_0_min,
llr_1_min)/bin_width)-1;
                temp = max(llr_0_max, llr_1_max)/bin_width-
bin_offset+1;
                bin_count = ceil(temp);
                if(bin_count == temp)
                    bin_count = bin_count+1;
                end

            else

                bin_offset = -1;
                bin_count = 3;
```

```
            end
            lots_of_bins = true;

        else
            lots_of_bins = false;
            bin_count = 4;
        end

        histogram = zeros(2,bin_count);

        for bit_index = 1:length(bits)
            if(llrs(bit_index) == -Inf)
                histogram(bits(bit_index)+1,1) =
histogram(bits(bit_index)+1,1)+1;
            elseif(llrs(bit_index) == Inf)
                histogram(bits(bit_index)+1,bin_count) =
histogram(bits(bit_index)+1,bin_count)+1;
            else
                if(lots_of_bins == true)
                    if(bin_width > 0.0)

histogram(bits(bit_index)+1,floor(llrs(bit_index)/bin_width)-
bin_offset+1) =
histogram(bits(bit_index)+1,floor(llrs(bit_index)/bin_width)-
bin_offset+1)+1;
                    else
                        histogram(bits(bit_index)+1,2) =
histogram(bits(bit_index)+1,2)+1;
                    end
                else
                    histogram(bits(bit_index)+1,bits(bit_index)+2) =
histogram(bits(bit_index)+1,bits(bit_index)+2)+1;
                end
            end
        end

        pdf = zeros(2,bin_count);
        pdf(1,:) = histogram(1,:)/bit_0_count;
        pdf(2,:) = histogram(2,:)/bit_1_count;

        mutual_information = 0.0;
        for bit_value = 0:1
            for bin_index = 1:bin_count
                if(pdf(bit_value+1,bin_index) > 0.0)
                    mutual_information = mutual_information +
0.5*pdf(bit_value+1,bin_index)*log2(2.0*pdf(bit_value+1,bin_index)/(
pdf(1,bin_index) + pdf(2,bin_index)));
                end
            end
        end
    end
end
```

```
%-------------------------------------------------------------------------%
%-------------------------Program Inner Kode------------------------------%
%-------------------------------------------------------------------------%


% Menentukan banyak bit untuk dikodekan %
bit_count=10000;

% Jumlah dari apriori mutual informasi yang dipertimbangkan %
IA_count=11;


% Apakah menggunakan metode histogram untuk memastikan mutual
informasinya%
histogram=0;

% Kanal Sinyal Noise Ratio dalam dB %
SNR = -6;


% Membangkitkan beberapa bit secara acak %
uncoded_bits  = round(rand(1,bit_count));

% Mengkodekan menggunakan kode konvolusi rekursiv yang sistematik R
=1/2 %
[encoded1_bits, encoded2_bits] =
convolutional_encoder(uncoded_bits);

% modulasi dengan BPSK %
tx1 = -2*(encoded1_bits-0.5);
tx2 = -2*(encoded2_bits-0.5);


% demodulasi dengan BPSK %
apriori_encoded1_llrs = (abs(rx1+1).^2-abs(rx1-1).^2)/N0;
apriori_encoded2_llrs = (abs(rx2+1).^2-abs(rx2-1).^2)/N0;

% Mempertimbangkan apriori mutual informasi %
```

```
requested_IA = 0.999*(0:1/(IA_count-1):1);


% Mempertimbangkan tiap apriori mutual informasi %
for IA_index = 1:IA_count

    % Membangkitkan apriori LLRs yang memiliki informasi a priori
mutual yang dipertimbangkan %
    apriori_uncoded_llrs = generate_llrs(uncoded_bits,
requested_IA(IA_index));

    % Memastikan mutual informasi dari LLRs yang dibangkitkan %
    if histogram
        IA(IA_index) =
measure_mutual_information_histogram(apriori_uncoded_llrs,
uncoded_bits);
    else
        IA(IA_index) =
measure_mutual_information_averaging(apriori_uncoded_llrs);
    end

    % Proses algoritma BCJR %
    [aposteriori_uncoded_llrs, aposteriori_encoded1_llrs,
aposteriori_encoded2_llrs] = bcjr_decoder(apriori_uncoded_llrs,
apriori_encoded1_llrs, apriori_encoded2_llrs);



    % Memastikan mutual informasi dari extrinsic LLRs
    if histogram
        IE(IA_index) =
measure_mutual_information_histogram(extrinsic_uncoded_llrs,
uncoded_bits);
    else
        IE(IA_index) =
measure_mutual_information_averaging(extrinsic_uncoded_llrs);
    end



% Hasil %
figure
plot(IA,IE);
xlim([0 1]);
ylim([0 1]);
xlabel('Mutual Informasi I_A pada Input dari enkoder');
ylabel('Mutual Informasi I_E pada Output dari enkoder');
title(['Funsi Inner pada SNR = ', num2str(SNR), ' dB']);

% Menampilkan area bawah dari fungsi inner %
annotation('textbox','String',{['Area = ',
num2str(area)]},'LineStyle','none','Position',[0.7 0.1 0.2 0.1]);
```

```
%-------------------------------------------------------------------------%
%--------------------------Program Kode Outer-----------------------------%
%-------------------------------------------------------------------------%

% Menentukan banyak bit untuk dikodekan %
bit_count=100;

% Jumlah dari apriori mutual informasi yang dipertimbangkan %
IA_count=11;

% Apakah menggunakan metode histogram untuk memastikan mutual
informasinya%
histogram=0;

% Mempertimbangkan tiap-tiap apriori mutual informasi
for IA_index = 1:IA_count

    % Membangkitkan apriori LLRs yang memiliki informasi a priori
mutual
    % yang dipertimbangkan %
    apriori_encoded1_llrs = generate_llrs(encoded1_bits,
requested_IA(IA_index));
    apriori_encoded2_llrs = generate_llrs(encoded2_bits,
requested_IA(IA_index));

    % Tidak ada apriori informasi untuk bit yang tidak dikodekan
saat proses pada bagian outer %
    apriori_uncoded_llrs = zeros(1,length(uncoded_bits));

    % Memastikan mutual informasi dari LLRs yang dibangkitkan %
    if histogram
        IA(IA_index) =
(measure_mutual_information_histogram(apriori_encoded1_llrs,
encoded1_bits) +
measure_mutual_information_histogram(apriori_encoded2_llrs,
encoded2_bits))/2;
    else
        IA(IA_index) =
(measure_mutual_information_averaging(apriori_encoded1_llrs) +
measure_mutual_information_averaging(apriori_encoded2_llrs))/2;
```

```
    end;

    % Proses algoritma BCJR %
    [aposteriori_uncoded_llrs, aposteriori_encoded1_llrs,
aposteriori_encoded2_llrs] = bcjr_decoder(apriori_uncoded_llrs,
apriori_encoded1_llrs, apriori_encoded2_llrs);


    % Memastikan mutual informasi dari extrinsic LLRs %
    if histogram
        IE(IA_index) =
(measure_mutual_information_histogram(extrinsic_encoded1_llrs,
encoded1_bits) +
measure_mutual_information_histogram(extrinsic_encoded2_llrs,
encoded2_bits))/2;
    else
        IE(IA_index) =
(measure_mutual_information_averaging(extrinsic_encoded1_llrs) +
measure_mutual_information_averaging(extrinsic_encoded2_llrs))/2;
    end;


    % Memperbaharui daerah bawah dari fungsi kode outer %
    if(IA_index > 1)
       area = area + (IE(IA_index)+IE(IA_index-1))*(IA(IA_index)-
IA(IA_index-1))/2;
    end

end

% menampilkan Hasil %
figure
semilogy(IA,BER);
xlim([0 1]);
ylim([min(100/bit_count,0.1) 1]);
xlabel('I_a');
ylabel('BER');


figure
plot(IE,IA);
xlim([0 1]);
ylim([0 1]);
xlabel('I_e');
ylabel('I_a');

% Menampikan area bawah dari fungsi outer %
annotation('textbox','String',{['Area = ', num2str(1-
area)]},'LineStyle','none','Position',[0.7 0.1 0.2 0.1]);
```