```
clear all; close all; clc;
% parameter
N = 7; % quantity of samples
e = randn(N); % NxN
e = e / max(e); % Nx1 [-1,1]

%e = sin(318.*(1:N)+3)'+e;

lag = 1; % AR(p) p is lag
% (N-lag)>=2 need to be true(==1), '>1' -> lag ,'>2' integral
a = zeros(1,N);
ac = zeros(1,N);
```

設計 a(t) 的部分

```
% time varying a --> design varying a
% TVAR a = sum(a.*h)
% h(t,n,j), r = t./(n+1) h = r.^j
h = @(t,n,j) (t./(n+1)).^j; % Anonymous匿名函數
% a(r) = g_struct.varing_gain( round(r.*(n+1) ) ).*h(r.*(n+1),n,j)
at_gain = @(t,n,j) -sum( (t./n).*h(t,n,j) );% a use g_struct.varing_gain
for i = lag:N
    a(i) = at_gain(i,N,i);
end
```

```
%a = [0.999, -0.0104, -0.0042];
% ac = [0.1, 0.9, -0.9 ,0.3, 0.5]; % constant close to 0, 1...
k = randn(N);
ac = k / max(k);  % random a [-1,1]

x = zeros(1,N);
xc = zeros(1,N); % 初始化矩陣大小 constant

for i =1:lag % lag not calc
    x(i) = e(i); % a(i).*x(j,i)+e(j);
    xc(i) = e(i); % ac(i).*x(j,i)+e(j);
end
Deq = zeros(1,N-lag); Req = Deq;
for i = (1+lag):N % calc start from AR(p->lag) --> lag+1
    ax_poly = zeros(1,lag);
    axc_poly = zeros(1,lag);
    for k =1:lag % ex: a1.*x_{3-lag}+ a2.*x_{3-lag}, lag =2
        ax_poly(k) = a(i-k).*x(i-k);
        axc_poly(k) = ac(i-k).*x(i-k);
    end
    x(i) =sum(ax_poly)+e(i);
    xc(i) = sum(axc_poly)+e(i); % 對照組 random a
    g_struct = struct('lag_p',lag,'samples',N, 'varing_gain',a,...
        'predict',x,'noise',e, 'input',e );
```

```
    [D,R] = rateFcn(g_struct);
    Deq(i) = D; Req(i) = R;
    fprintf('\nseq: %d/%d\n D: %d, R: %d, lag: %d\n',i,N,Deq(i),Req(i),lag);
end

% Plot
% print the distortion rate of x(lag:N) t in [lag, N]
str1 = ['AR', '(', string(lag), ')' ];
t = linspace(1,N,N);
ylimit_const = [-3, 3];
AR = sprintf( str1(1)+str1(2)+str1(3)+str1(4));

figure();
subplot(3, 1, 1);
plot(t(lag:end),e(lag:end));
hold on;
plot(t(lag:end),xc(lag:end));
title('gain of random num');
xlim([lag, N]); %xlim(0, N);
ylim(ylimit_const); %ylim(-2, 2);
xlabel('samples');
ylabel('e, x') ;
% legend(['noise', 'AR(2)'], loc='best');
% tight_layout(pad=0.5, w_pad=0.5, h_pad=1.0);
legend('noise', AR)
subplot(3, 1, 2);
plot(t(lag:end),e(lag:end));
hold on;
plot(t(lag:end),x(lag:end));
title('gain of a = r, r = t/N & distotion rate')
xlim([lag, N]); %xlim(0, N);
ylim(ylimit_const); %ylim(-2, 2);
xlabel('samples');
ylabel('e, x') ;
legend('noise', AR)
figure();
plot(Deq,Req);
title('distortion D/R');
xlabel('D');
ylabel('R') ;
```

```
function [D,R] = rateFcn(g_struct)
% have g_struct generated
% g_struct = struct('lag_p',lag,'samples',N, 'varing_gain',a,...
%         'predict',x,'noise',e, 'input',e );
%function [D,R] = rate(g_struct)
    %D = 0;
    %R = 1;
%end

% initial 宣告
```

```
sigma = zeros(1,g_struct.samples);
% calc the parameters
sigma = var(g_struct.predict);

%theta = max(x) - sigma./4;
theta = 1.0893e-86;

% calcl the g
omega = linspace(-pi,pi,1000);
assignin('base','bug_w',omega);
%find(bug_w>=0.5027,1);%581 找最接近 f= 0.08, => 0.08.*2*pi = 0.5027
%omega = logspace(-pi,pi,1000);
g = zeros(length(omega),g_struct.samples);
% g(i,:) = 1./( (sigma).^2 ).*abs(1+ sum(g_struct.varying_gain .* exp(-j.*m*omega(i)) ) ).^2;
for i = 1:length(omega)
    poly_of_sum = 0;
    for m = 1:g_struct.samples
        poly_of_sum = poly_of_sum + g_struct.varing_gain(m) .* exp(-j.*m.*omega(i));
        g(i,m) = 1./( (sigma).^2 ).*abs(1+ poly_of_sum ).^2;
    end
end

k= 1./g;
k_flat = reshape(k.',1,[]); % flat to 1xN 找N個裏面最小的
%-----------------------------------------------------------------%
```

找所有Omega產生的1/g的最小值

```
%-----------------------------------------------------------------%
% find the index of minumax value 暫時找一個頻率的最大值，這裏先蓋過去
if isempty(min( k_flat(k_flat>theta) ))
    fprintf('\nmin>theta\n');
    index = find( k_flat == min( k_flat ),1,'last');
else
    % index = find( k_flat == min( k_flat(k_flat>theta) ),10,'last');% it find the same iter
    index = find( k_flat == min( k_flat ),20,'last');
end

%iter = mod(index, g_struct.samples)+1 % searching the iteration of min value
% iter must large than p lags of AR(p)

iter = max(mod(index,g_struct.samples)) + (g_struct.lag_p+1); % 10個點找最大的
% iter = max(mod(bug_index,g_struct.samples)) + (g_struct.lag_p+1)
%-----------------------------------------------------------------%
```

發現數值很奇怪，所以只有找一個頻率蓋掉找到的iter，兩個找法都有一些問題，猜是a(t)寫的有點問題。

他們都找一樣的iter帶入去積分

```
%-----------------------------------------------------------------%
```

```
% f= 0.08, => 0.08.*2*pi = 0.5027
freq = 0.08
omg = 2.*pi.*freq
indexOMG = find(omega>=omg,1)
new_k = k(indexOMG,:)
iter = find (new_k==min(new_k))

% 每次不知道?什麼不是最大就是最小，都找到第一項或是最後一項？
% 42    43    44    45    46    47    48    49    50     1

% [i,j] = find(k == k_flat(index),1,'last'); % 從後面找積分項數比較多的一項
%fprintf(' We get the MIN at:');
%fprintf('\n seq = %d\n min(1/g): %d\n theta: %d\n in omega: %d\n',iter,k_flat(index),theta,ome
%figure();stem(g);title('stem g seqs'); figure();stem(1./g); title('stem 1/g seqs');
assignin('base','bug_theta',theta);
assignin('base','bug_k_flat',k_flat);
assignin('base','bug_k',k);
assignin('base','bug_index',index);

% generator integral string of 'iter' seqs

% D_theta = @(r,w) 1./( 1./(sigma(i).^2).* abs(1 + sum(1 + a(r).*exp(-j.*length(s(i,:)).*w) ) )
str1 = '@(r,w) 1./(1./ (';
str2 = string(sigma);
str3 = '.^2).* abs(1+';
% ------------------------------------------------------%
% 定義匿名函數給 -> integral(積分的匿名函數調用,-inf,inf)
% a(r), r = t./(n+1), t = r(n+1)
% h = (t./(n+1) ).^j, r.^j % arrayfun(fun)
h = @(t,n,j) (t./(n+1)).^j; % h(r(n+1),n,j)
% n = g_struct.samples; t = r.*(n+1); j = 1:(r.*(n+1));
% a(t) = -sum(a(j).*h(r.*(n+1),n,j))
% a(r) = g_struct.varing_gain( round(r.*(n+1) ) ).*h(r.*(n+1),n,j)
ar_gain = @(r,n,j,a) sum( a( round(r.*(n+1) ) ).*h(r.*(n+1),n,j) );% a use g_struct.varing_gain
% anonymous function is a short term function so need to --> *.m file
% ------------------------------------------------------%

poly_str4 = string('');
for k = 1:iter % with a(k), g_struct.varing_gain(k)
    % ar(r, g_struct.samples, k ,g_struct.varing_gain) % k is the 公式中的 j
    gain_list_str = '['; %ex: [ 0.6800    0.7000    0.7200    0.740]
    for length_of_a = 1:length(g_struct.varing_gain)
        if length_of_a == length(g_struct.varing_gain)
            gain_list_str = sprintf(gain_list_str+ string(g_struct.varing_gain(length_of_a))+']
        else
            gain_list_str = sprintf(gain_list_str+ string(g_struct.varing_gain(length_of_a))+',
        end
    end
    gain = sprintf('ar_gain(r,'+string(g_struct.samples)+','+string(k)+','+gain_list_str+')');
    %gain = string(g_struct.varing_gain(k)); %-jwm  m is eq to iter
    m = string(k);
    str_temp = '.*exp(-j.*';
    if k==1
        poly_str4 = sprintf(  gain + str_temp + m +'.*w)' );
```

```
    else
        poly_str4 = sprintf( poly_str4 + '+' + gain + str_temp + m +'.*w)' );
    end
end
% for k = 1:iter % with a(k), g_struct.varing_gain(k) this no r in integral
%     gain = string(g_struct.varing_gain(k)); %-jwm  m is eq to iter
%     m = string(k);
%     str_temp = '.*exp(-j.*';
%     if k==1
%         poly_str4 = sprintf(  gain + str_temp + m +'.*w)' );
%     else
%         poly_str4 = sprintf( poly_str4 + '+' + gain + str_temp + m +'.*w)' );
%     end
% end

% str4: a1(r)*exp(-jmw)+a2(r)*exp(exp(-jmw)+...
str4 =sprintf('('+poly_str4+')');% (sum)
str5 = ').^2)';
D_poly = sprintf(str1+str2+str3+str4+str5);
D_fun = str2func(D_poly); % str2func
assignin('base','bug_fcn_d',D_fun);
D = integral2(D_fun, 0,1, -pi, pi);

str1 = sprintf('@(r,w) 1./2.*log10(1./'+string(theta)+'./(1./ ('); % add 1./2log(1./theta./g)to
str5 = ').^2) )';
R_poly = sprintf(str1+str2+str3+str4+str5);
R_fun = str2func(R_poly); % struct()
R = integral2(R_fun, 0,1, -pi, pi);
%fprintf(' D: %d  , R: %d\n',D,R);

% k_flat(iter) % use this minimum value in the final iteration
% R_poly
% D_poly
% D
% R
% calc end %
end
```

a(r) 給積分匿名函數取用

integral2() 丟進來14x14的 r.....

assignin('base','bug_r',r); 可以把**r assign**到**workspace**叫做**bug_r**

帶入**t./(n+1)**好像會超過 N，N=50，會出現**50**

```
function [aa] = ar_gain(r,n,j,a)
% ------------------------------------------------------%
% 定義匿名函數給 -> integral(積分的匿名函數調用,-inf,inf)
% a(r), r = t./(n+1), t = r(n+1)
% h = (t./(n+1) ).^j, r.^j % arrayfun(fun)
h = @(t,n,j) (t./(n+1)).^j; % h(r(n+1),n,j)
```

```
% n = g_struct.samples; t = r.*(n+1); j = 1:(r.*(n+1));
% a(t) = -sum(a(j).*h(r.*(n+1),n,j))
% a(r) = g_struct.varing_gain( round(r.*(n+1) ) ).*h(r.*(n+1),n,j)
%a = @(r,n,j,a) sum( a( round(r.*(n+1) ) ).*h(r.*(n+1),n,j) );% a use g_struct.varing_gain
if r >=0.99
    aa = -sum( a( ceil(r.*(n) ) ).*h(ceil( r.*(n) ) ,n,j) ) ;% r ==1 .* n+1 exceed the samples
else
    %--------------------------------%
    %Array indices must be positive integers or logical values
    % assemble -> workspace -> debuging
%     assignin('base','bug_r',r);
%     assignin('base','bug_n',n);
%     assignin('base','bug_fcn_h',h);
%     assignin('base','bug_j',1:10);
    %g_struct.varing_gain( round(bug_r.*(bug_n+1) ) )
    %bug_fcn_h(round( bug_r.*(bug_n+1) ) ,bug_n,bug_j)
    %bug_a = g_struct.varing_gain( round(bug_r.*(bug_n+1) ) ).*bug_fcn_h(round( bug_r.*(bug_n+1
    %--------------------------------%
%     a( round(r.*(n+1) ) )
%     h(round( r.*(n+1) ) ,n,j)
    aa = -sum( a( ceil(r.*(n) ) ).*h(ceil( r.*(n+1) ) ,n,j) ); % a use g_struct.varing_gain
end
% anonymous function is a short term function so need to --> *.m file
% ------------------------------------------------------%
```

```
The function "rateFcn" was closed with an 'end', but at least one other function definition was not. All
 functions in a script must be closed with an 'end'.
```