Author： 黃仕鈞

Reclaim the Autoregressive Model

AR model and its parameters

An autoregressive (AR) model is a representation of a type of random process.

The autoregressive model specifies that the output variable depends linearly on its own **previous values** and on a **stochastic term**.

https://en.wikipedia.org/wiki/Autoregressive_model

The notation **AR**(p) indicates an autoregressive model of order p. The **AR**(p) model is defined as

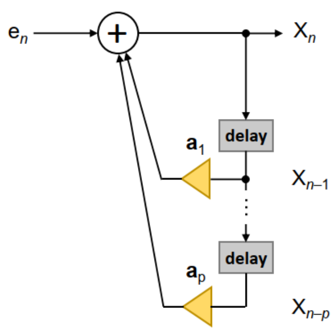
convolution

$$X_n = c + \underbrace{\sum_{i=1}^{p}}_{} a_i X_{n-i} + e_n$$

```
X[n] = c + np.sum(a * X[np.arrange((n-1),(n-p-1),-1)]) + e[n]
```

where a_i are the parameters of the model, c is a constant, and e_t is white noise.

NBE_Aalto University_neuroscience_course



$$X_n = c + \sum_{i=1}^{p} a_i X_{n-i} + e_n$$

Distortion Rate

RATE DISTORTION STUDY FOR TIME-VARYING AUTOREGRESSIVE GAUSSIAN PROCESS

$$x_t = -\sum_{m=1}^{M} a_m(r) x_{t-m} + z_t, \ \text{for } t = 1, 2, \ldots, N$$

- $r = \frac{t}{N}$ is defined as distortion rate.

$$g(r,\omega) = \frac{1}{\sigma_z^2} \left| 1 + \sum_{m=1}^{M} a_m(r) e^{-jm\omega} \right|^2$$

Use Anonymous Function + (fucntion, or insert by struct) and set the integral description.

Use str2func, sprintf(str1+str2) to implement it, Python use the ways like the dictionary.. and str = str1+str2.

And a important skill is add generator string to Anonymous Fcn, design it for nested loops.

$D_\theta = \frac{1}{2\pi}\int_{-\pi}^{\pi}\int_0^1 \min\left[\theta, \frac{1}{g(r,\omega)}\right] dr d\omega$, iteration > 1, need past value

- θ is upper threshold, set to 0.1 close to 0

$R(D_\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\int_0^1 \max\left[0, \frac{1}{2}\log\frac{1}{\theta \times (r,\omega)}\right] dr d\omega$, iteration >1, need past value

- 0 is low limit

```
clear all; clc;
% parameter
N = 7; % quantity of samples
e = randn(N);
e = e / max(e);

% es = sin((1:N))+ sin((1:N).*2+10); % add sin sin wave
% e = 1./6.*(es'+e); % es' is transpose result of es

a = [];
sets = 5; % test sets
lag = 5; % AR(p) p is lag
% (N-lag)>=2 need to be true(==1), '>1' -> lag ,'>2' integral

% Generate the a(r) AR is not able to linear regression but just test
% time varying a --> design varying a
for i = 1:N
    a(i) = (i/N);
end
ac = zeros(sets,1); % index sets with constant
ac = [0.1, 0.9, -0.9 ,0.3, 0.5]; % constant close to 0, 1...

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

for j =1:lag % lag not calc
    for i = 1:5 % which data
        x(j,i) = 0; % a(i).*x(j,i)+e(j);
        xc(j,i) = 0; % ac(i).*x(j,i)+e(j);
    end
end
sigma = zeros(1,N); theta = zeros(1,N); r = zeros(1,N); omega = zeros(1,N);
g = zeros(1,N); xx = zeros(1,N); nn = zeros(1,N);
% omega and m don't know the const values

for j = 1:sets % 這裏開始計算distortion rate N-lag-1 筆資料
    for i = (1+lag):N % calc start from AR(p->lag) --> lag+2
        xc(i,j) = ac(j).*x(i-lag,j)+e(i); % 對照組 constant a
        x(i,j) = a(i).*x(i-lag,j)+e(i);
```

```
        g_struct = struct('sets_index_now',j,'data_index_now',i,'lag_p',lag,...
            'samples',N, 'varing_gain',a,'predict',x,'noise',(e.*0.9), 'input',e );
        sigma(i) = var(g_struct.predict(:,1)); % get the varience
        theta(i) = max(g_struct.predict(:,1)) - sigma(i)./4; % 暫時這麼算 上限 upper threshold
        r(i) = g_struct.varing_gain(i);
        omega(i) = 2.71828;
        x_in = g_struct.predict(:,1);% x(:,1), x(:,2) is the same...
        % calc g value
        % calcl the g
        g = zeros(length(omega),g_struct.samples);
        for ii = 1:length(omega)
            g(ii,:) = 1./( (sigma).^2 ).*abs(1+ sum(x_in .* exp(-j.*g_struct.varing_gain.*omega
        end

        k= 1./g;
        k_flat = reshape(k.',1,[]);
        % find the index of minumax value
        % find( k == min( k(k>theta) ),1,'last'); %get error when k<theta but works
        index = find( k_flat == min( k_flat(k_flat>0) ),1,'last');
        [ii,jj] = find(k == k_flat(index),1,'last'); % 從後面找積分項數比較多的一項

        %g(i) = 1./(sigma(i).^2).* abs(1 + sum(1 + x_in.*exp(-j.*length(N-lag+1).*omega(i)))).
        % min(g(g~=0)) % g的最小數值
        %ind = find(g == min( g(g>0) ),1,'last'); % 偷吃步一下，找曲線波動滿有用的
        ind =jj;
        % g(ind) % g的最小值代入ind這一項
        %D_theta = @(r,w) 1./( 1./(sigma(i).^2).* abs(1 + sum(1 + s(i,:).*exp(-j.*length(s(i,:
        for lengths = 1:length(g_struct.predict(:,1))
            xx(i) = g_struct.predict(i,1);
            %nn = g_struct.noise(i,1);
            nn(i) = 0.212826888264831; % set to const now
            %r = g_struct.data_index_now(i)/g_struct.samples;
            r(i) = a(i);
        end
        str1 = '@(r,w)1./( 1./(';
        str2 = string(sigma(i));
        str3 = ').^2).*abs(1+(';
        for indd = 1:ind % strings of sum part
            list = ['((', string(a(i)), ').*r','+' , '(',string(nn(i)),')']; % x 要改成跟 r.*(輸
            str4 = sprintf( list(1)+list(2)+list(3) );
            str5 = ').*exp(-j.*(';
            str6 = string(indd); % exp(-j.*omega.*m) whatis m?
            str7 = ').*w)+';
            str8 = ').*w)';
            list_fn(indd) = sprintf(str4+str5+str6+str7);
            if indd == ind
                list_fn(indd) = sprintf(str4+str5+str6+str8);
            end
        end
        strff = '';
        for indd = 2:ind
            str9 = strff;
            strff = sprintf(str9+list_fn(indd-1)+list_fn(indd));
        end
```

```
        strfin = ').^2)';
        str_intgral_D = sprintf(str1+str2+str3+strff+strfin); % for D
        D_integral_fun = str2func(str_intgral_D); % struct()
        D = integral2(D_integral_fun, 0,1, -pi, pi);

        str11 = '@(r,w)1./2.*log(1./';
        str12 = string(theta(i)); % multiple min value of 1/g -integral-> D (1/g), use D
        str13 = './( 1./(';
        strfin = ').^2))';
        str1 = sprintf(str11+str12+str13);
        str_intgral_R  = sprintf(str1+str2+str3+strff+strfin);
        R_integral_fun = str2func(str_intgral_R); % struct()
        R = integral2(R_integral_fun, 0,1, -pi, pi);
        Deq(i) = D;
        Req(i) = R;

    end
end
clearvars str11 str12 str13 str1 str2 str3 str4 str5 str6 str7 str8 str9 strff strfin list list
ind % use this minimum value in the final iteration
```

```
ind = 7
```

```
str_intgral_D % the last one integral calc
```

```
str_intgral_D =
"@(r,w)1./( 1./(0.02832).^2).*abs(1+(((1).*r).*exp(-j.*(1).*w)+((1).*r).*exp(-j.*(2).*w)+((1).*r).*exp(-j.*(2).*w)+(
```

```
str_intgral_R % the last one integral calc
```

```
str_intgral_R =
"@(r,w)1./2.*log(1./0.42106./( 1./(0.02832).^2).*abs(1+(((1).*r).*exp(-j.*(1).*w)+((1).*r).*exp(-j.*(2).*w)+((1).*r)
```

```
D
```

```
D = 0.0402
```

```
R
```

```
R = -17.3986
```

```
% calc end %
% print the distortion rate of x(lag:N) t in [lag, N]
```

$$g(r,\omega) = \frac{1}{\sigma_z^2}\left|1 + \sum_{m=1}^{M} a_m(r)e^{-jm\omega}\right|^2$$

$$D_\theta = \frac{1}{2\pi}\int_{-\pi}^{\pi}\int_{0}^{1} \min\left[\theta, \frac{1}{g(r,\omega)}\right] dr d\omega$$

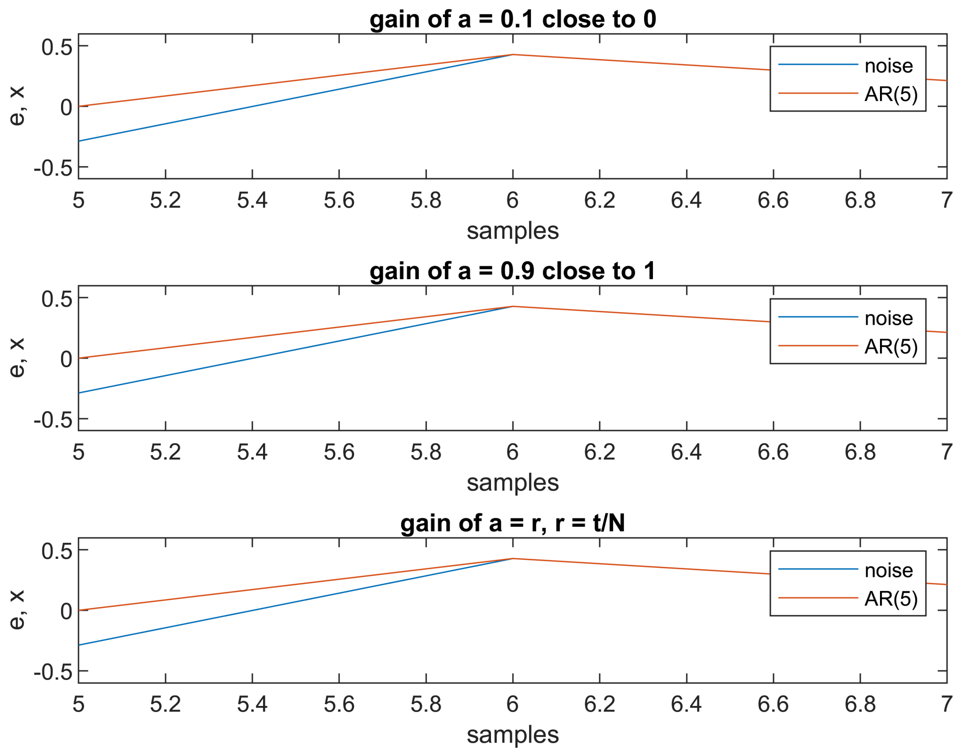
- θ is upper threshold, set to 0.1 close to 0

$$R(D_\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\int_{0}^{1}\max\left[0, \frac{1}{2}\log\frac{1}{\theta \times (r,\omega)}\right] dr d\omega$$

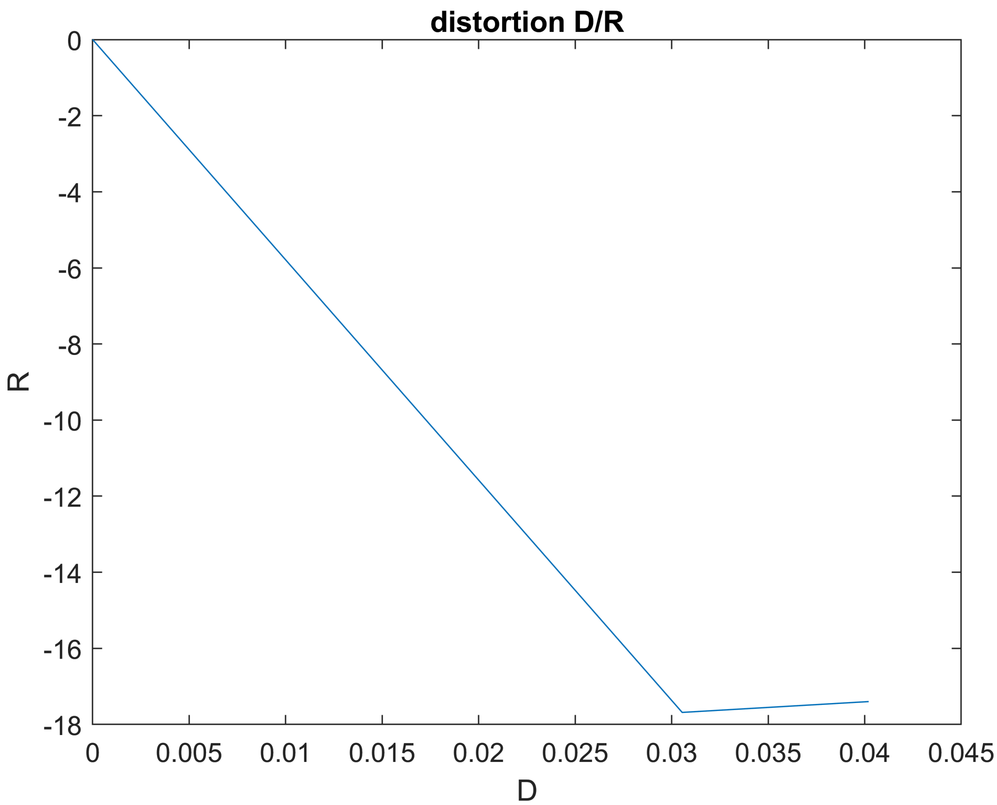
- 0 is low limit

```
% Plot
str1 = ['AR', '(', string(lag), ')' ];
t = linspace(1,N,N);
ylimit_const = [-0.6, 0.6];
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,1));
title('gain of a = 0.1 close to 0');
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),xc(lag:end,2));
title('gain of a = 0.9 close to 1')
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, 3);
plot(t(lag:end),e(lag:end));
hold on;
plot(t(lag:end),x(lag:end,1));
title('gain of a = r, r = t/N')
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)
```




```
figure()
plot(Deq,Req);
xlabel('D');
ylabel('R') ;
title('distortion D/R');
```




```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%     Calc approach ways    %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This block not calculate
% in 2020 MATLAB the function can be defined in the same file
% x(i,j) = a(i).*x(i-lag,j)+e(i);
% s = struct(field1,value1,field2,value2,field3,value3,field4,value4)
% field 儘量有意義一點
% call by s.field1 or s(1) it will get te value
% x(j,i) = a(i).*x(i-lag,j)+e(i); from this seris x(:,1)
s = x(:,1); % input_seris  x(:,1), x(:,2) is the same...
s = zeros(N-lag,N); % it's size initial
s(i,:) = x(:,1); % input_seris  x(:,1), x(:,2) is the same...
sigma(i) = var(x(:,1)); % get the varience
theta(i) = max(s(i,:)) - sigma(i)./4; % 暫時這麼算 上限 upper threshold
r(i) = a(i);
omega(i) = 2.71828;
% after summarize above
% we know it need i, j, lag, N, x, a, e,  for our calc integral
g_struct = struct('sets_index_now',j,'data_index_now',i,'lag_p',lag,...
    'samples',N,'predict',x,'noise',(e.*0.9),'input',e )
```

```
g_struct = struct with fields:
    sets_index_now: 5
    data_index_now: 7
             lag_p: 5
           samples: 7
           predict: [7×5 double]
             noise: [7×1 double]
```

```
input: [7×1 double]
```

```
% searching the minimum of 1/g in our prediction values, O(n) sorting 算法可能更好
% g = zeros(2,2) % (i,g) store the index in g for 丟入參數再for找數字
% 再回去g_struct產生公式積分
%g = zeros(1,N);
%g(i) = 1./(sigma(i).^2).* abs(1 + sum(1 + s(i,:).*exp(-j.*length(s(i,:)).*omega(i)))).^2;% ome
% D function 其他略，看上面s的部分要用struct array 加上字串處理設計出積分的項並做成匿名函數，新的iter
% ignore the D description
% function to get s's parameter in string
%D_theta = @(r,w) 1./( 1./(sigma(i).^2).* abs(1 + sum(1 + s(i,:).*exp(-j.*length(s(i,:)).*w))).
%D(i) = integral2(fun,0,1,-pi,pi);
%R(i) = integral2(fun,xmin,xmax,ymin,ymax);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Matlab Deal the prob

Matlab Deal the prob with Anonymous Function (匿名函式), can solve the nested code. str2func

- anonymous function = @(x) exp(-x); = @(x,y) exp(x)+y
- integral(anon_fun, -inf, inf)
- integral2(anon_fun, -inf,inf, -inf, inf) double integral

>> str1 = '@(x)'

>> str2 = 'exp(-x.^2)'

>> str3 = '.*'

>> str4 = 'log(x).^2'

>> str5 = [str1, str2, str3, str4]

str5 =

'@(x)exp(-x.^2).*log(x).^2'

>> fun = str2func(str5)

fun =

function_handle with value:

@(x)exp(-x.^2).*log(x).^2

>> q = integral(fun,0,Inf)

q =

1.9475

% 有變數要用 sprintf, 因爲 Matlab 好像，少執行了這塊

>> str1 = ['AR', '(', string(lag), ')']

str1 =

1×4 string array

"AR"   "("   "30"   ")"

>> AR = sprintf(str1(1)+str1(2)+str1(3)+str1(4))

AR =

"AR(30)"

Integral Source may from:

```m

% Trapezoidal numerical integration to do it

y = sin(t)

trapz(t,y) %

trapz(x)  % x is meshgrid() result

%  inf -inf prob will failed

```

MonteCarlo approach of the inf -inf prob

But can't deal the $\int_{-\infty}^{\infty}\frac{f(x)}{g(x)}$, if g(x) have log() or exp(), may have problems for me.

So try matlab first.

MonteCarlo may not meet the anwser for me.

目前只會用Matlab做這類積分, 因爲匿名函數整合的很好

可以想想如何用trapz做，pytorch出來之後可以

使用torch.meshgrid(), torch.trapz()，並使用GPU進行計算，

Pytorch厲害在可以使用AMD和NVIDIA GPU進行這些動作。

自己做的話，積分問題還不太瞭解做法。

The probability theory approach People who are into probability theory usually like to interpret integrals as mathematical expectations of a random variable. (If you are not one of those, you can safely jump to Sect. 8.5.2 where we just program the simple sum in the Monte Carlo integration method.) More precisely, the integral $\int_a^b f(x)dx$ can be expressed as a mathematical expectation of $f(x)$ if x is a uniformly distributed random variable on $[a,b]$. This expectation can be estimated by the average of random samples, which results in the Monte Carlo integration method. To see this, we start with the formula for the probability density function for a uniformly distributed random variable X on $[a,b]$:

$$p(x) = \begin{cases} (b-a)^{-1}, & x \in [a,b] \\ 0, & \text{otherwise} \end{cases}$$

Now we can write the standard formula for the mathematical expectation $\mathrm{E}(f(X))$:

$$\mathrm{E}(f(X)) = \int_{-\infty}^{\infty} f(x)p(x)dx = \int_a^b f(x)\frac{1}{b-a}dx = (b-a)\int_a^b f(x)dx\,.$$

The last integral is exactly what we want to compute. An expectation is usually estimated from a lot of samples, in this case uniformly distributed random numbers $x_0,\ldots,x_{n-1}$ in $[a,b]$, and computing the sample mean:

$$\mathrm{E}(f(X)) \approx \frac{1}{n}\sum_{i=0}^{n-1} f(x_i)\,.$$

The integral can therefore be estimated by

$$\int_a^b f(x)dx \approx (b-a)\frac{1}{n}\sum_{i=0}^{n-1} f(x_i),$$

which is nothing but the Monte Carlo integration method.

8.5.2 Implementation of Standard Monte Carlo Integration

To summarize, Monte Carlo integration consists in generating n uniformly distributed random numbers x_i in $[a,b]$ and then compute

$$(b-a)\frac{1}{n}\sum_{i=0}^{n-1} f(x_i)\,. \tag{8.9}$$

We can implement (8.9) in a small function:

之前Llodmax 計算就會，（Llodmax I run before）

```
init:
X =  -1.5000  -0.5000        0   0.5000   1.5000
x =     -Inf  -1.0000  -0.2500   0.2500   1.0000      Inf
D =   0.4854

inter: 1:
X =  -2.0000  -0.5786   0.0000   0.5786   2.0000
x =     -Inf  -1.2893  -0.2893   0.2893   1.2893      Inf
D =   0.3504
SNR =    4.5541

inter: 2:
X =  -2.2893  -0.7073  -0.0000   0.7073   2.2893
x =     -Inf  -1.4983  -0.3537   0.3537   1.4983      Inf
D =   0.2996
SNR =    5.2346

inter: 3:
X =  -2.4983  -0.8191  -0.0000   0.8191   2.4983
x =     -Inf  -1.6587  -0.4095   0.4095   1.6587      Inf
D =   0.2729
SNR =    5.6396

inter: 4:
X =  -2.6587  -0.9074   0.0000   0.9074   2.6587
x =     -Inf  -1.7830  -0.4537   0.4537   1.7830      Inf
D =   0.2582
SNR =    5.8798

inter: 5:
X =  -2.7830  -0.9752   0.0000   0.9752   2.7830
x =     -Inf  -1.8791  -0.4876   0.4876   1.8791      Inf
D =   0.2501
SNR =    6.0196
```

```
inter: 6:
X =  -2.8791  -1.0270   0.0000   1.0270   2.8791
x =     -Inf  -1.9531  -0.5135   0.5135   1.9531      Inf
D =   0.2455
SNR =    6.0998

inter: 7:
X =  -2.9531  -1.0663   0.0000   1.0663   2.9531
x =     -Inf  -2.0097  -0.5331   0.5331   2.0097      Inf
D =   0.2429
SNR =    6.1454

inter: 8:
X =  -3.0097  -1.0960   0.0000   1.0960   3.0097
x =     -Inf  -2.0528  -0.5480   0.5480   2.0528      Inf
D =   0.2415
SNR =    6.1712

inter: 9:
X =  -3.0528  -1.1185        0   1.1185   3.0528
x =     -Inf  -2.0857  -0.5592   0.5592   2.0857      Inf
D =   0.2407
SNR =    6.1859

inter: 10:
X =  -3.0857  -1.1354  -0.0000   1.1354   3.0857
x =     -Inf  -2.1105  -0.5677   0.5677   2.1105      Inf
D =   0.2402
SNR =    6.1941
```

Run with trapz() and scipy.integral.quad(), still no good anwser.

trapz() 這種matlab會有同名function的庫有：

Numpy、Cupy、Scipy、Pytorch　用於一些數值計算

其中Pytorch最近很紅，因爲開始支援AMD & NVIDIA GPU以往都很多限制。

例如：Cupy似乎每個指令之間仍有所延遲，還有CPU、GPU計算比重，效能沒有想像中的好，而且只有NVIDIA可用。

```python
X = [-1.5, -0.5, 0, 0.5, 1.5]
```

```
iter: 99
iter: 100
X:  [-3.187248520083409, -1.187248520080732, 0.0, 1.187248520080732, 3.187248520083409]
 x:  [        -inf -2.18724852 -0.59362426  0.59362426  2.18724852         inf]
 D(MSE):  nan
 SNR(db):  nan
```

```python
# quantize
xi[0] = -inf #-inf
xi[1] = (X[0]+X[1])/2
xi[2] = (X[1]+X[2])/2
xi[3] = (X[2]+X[3])/2
xi[4] = (X[3]+X[4])/2
xi[5] = +inf
```

```
iter: 99
iter: 100
X:  [-3.187248520078561, -1.1872485200789515, 7.293822708266879e-13, 1.1872485200814602, 3.18724852008413
 x:  [-100.          -2.18724852   -0.59362426    0.59362426    2.18724852
          inf]
 D(MSE):  526.6777114594955
 SNR(db):  -27.215449397834007
```

```python
# quantize
xi[0] = -100 #-inf 設定有限
xi[1] = (X[0]+X[1])/2
xi[2] = (X[1]+X[2])/2
xi[3] = (X[2]+X[3])/2
xi[4] = (X[3]+X[4])/2
xi[5] = +inf
```

```
iter: 99
iter: 100
X:  [-0.9529144029785452, -0.05231024121168301, 0.7955043018409754, 1.9826689154313117, 3.982611139890205]
 x:  [-1.6        -0.50261232  0.37159703  1.38908661  2.98264003         inf]
 D(MSE):  0.3569490099011444
 SNR(db):  4.473938183420752
```

```python
# quantize
xi[0] = -1.6 #-inf
xi[1] = (X[0]+X[1])/2
xi[2] = (X[1]+X[2])/2
xi[3] = (X[2]+X[3])/2
xi[4] = (X[3]+X[4])/2
xi[5] = +inf
```