2023 Spring VLSI DSP Homework Assignment #2

4108064101 杜冠廷

Q1

Matlab code:

```
clear all
clc
clf

%Define number of sample
n = 1 : 3000;

%Define input signal
s = sin(2*pi*(n/12)) + cos(2*pi*(n/4));

%Define filter cofficients
M = 15;
b = zeros(M ,1);

%Define target signal
d = sin(2*pi*(n/12));

%Define step size
mu = 10^(-2);

%Initialize variable
u = zeros(15 ,1);
r = zeros(1  ,(length(n) - M - 1));
error_number = 16;
e = zeros(1 ,error_number);

%Define coverge display parameter
con = 0;

%Create a memory to storage the filter cofficients history
b_h = zeros([15 (length(n) - M - 1)]);
```

```
% Implement LMS adaptive filter
for i = 1 : length(n)
    %Get input signal
    for o = 15 : -1 : 2
        u(o) = u(o - 1);
    end
    u(1 ,1) = s(1 ,i)';

    %Compute output signal
    d_tilde = b' * u;

    %Compute error
    for z = 16 : -1 : 2
        e(z) = e(z - 1);
    end
    e(1 ,1) = d(i) - d_tilde;

    %Update filter cofficients
    b = b + mu*e(1)*u;
    for a = 1 : 15
        b_h(a ,i) = b(a);
    end

    %Caculate the RMS value
    r(1 ,i) = sqrt(mean(e.^2));

    %Check for convergence
    if (r(1 ,i) < (0.1/sqrt(2))) && (con == 0)
        disp(['Converged after ',num2str(i),' samples (mu = 10^(-2))']);
        con = 1;
    end
end

disp(['Min RMS value : ' ,num2str(min(r)),' (mu = 10^(-2))']);

% Plot prediction error (r) vs. sample index (n)
figure(1);
```

```
plot(1 : length(n),r);
xlabel('Sample index');
ylabel('Prediction error (RMS)');
xlim([1 300]);
title('Prediction error vs. sample index (mu = 1e-2)');

% Plot filter coefficients vs. sample index
figure(2);
plot(1 : length(n),b_h);
xlabel('Sample index');
ylabel('Coefficient value');
xlim([1 300]);
title('Filter coefficients vs. sample index (mu = 1e-2)');

% Apply 64-point FFT to impulse response of converged filter
b_fft = fft([b;zeros(49,1)],64);
f = linspace(0,1,64/2+1)*0.5;
mag_response = abs(b_fft(1:64/2+1));
phase_response = angle(b_fft(1:64/2+1));

% Plot magnitude response of filter
figure(3);
plot(f,mag_response);
xlabel('Normalized frequency');
ylabel('Magnitude');
title('Magnitude response of filter(mu = 1e-2)');

% Change step size to 1e-4 and repeat simulation

%Define number of sample
n = 1 : 100000;

%Define input signal
s = sin(2*pi*(n/12)) + cos(2*pi*(n/4));

%Define filter cofficients
M = 15;
b = zeros(M ,1);
```

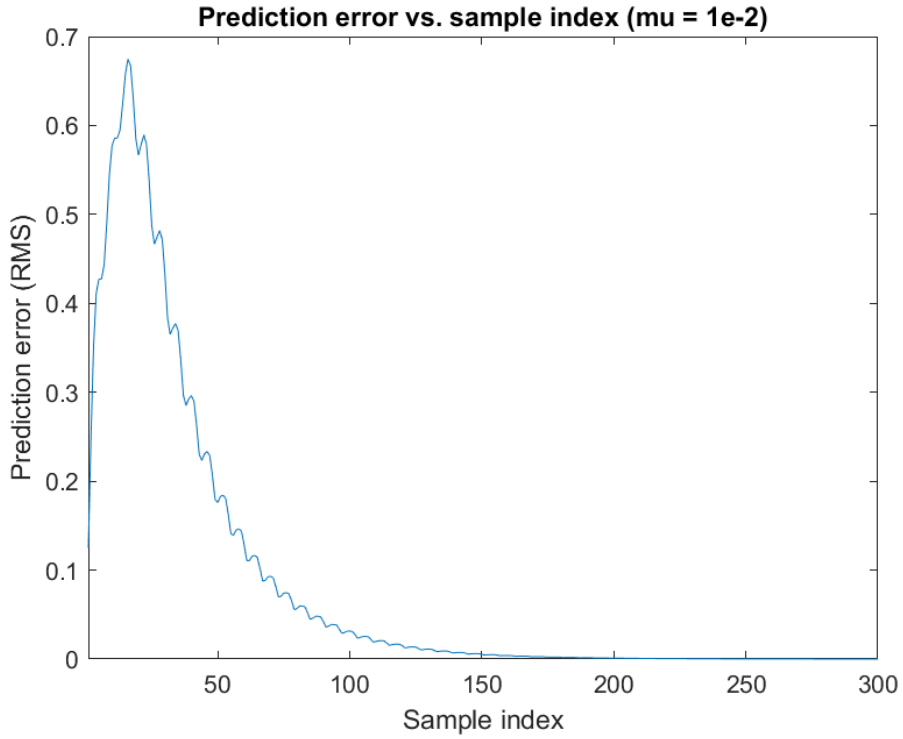
```
%Define target signal
d = sin(2*pi*(n/12));

%Define step size
mu = 10^(-4);

%Initialize variable
u = zeros(15 ,1);
r = zeros(1  ,(length(n) - M - 1));
error_number = 16;
e = zeros(1 ,error_number);

%Define coverge display parameter
con = 0;

%Create a memory to storage the filter cofficients history
b_h = zeros([15 (length(n) - M - 1)]);

% Implement LMS adaptive filter
for i = 1 : length(n)
    %Get input signal
    for o = 15 : -1 : 2
        u(o) = u(o - 1);
    end
    u(1 ,1) = s(1 ,i)';

    %Compute output signal
    d_tilde = b' * u;

    %Compute error
    for z = 16 : -1 : 2
        e(z) = e(z - 1);
    end
    e(1 ,1) = d(i) - d_tilde;

    %Update filter cofficients
    b = b + mu*e(1)*u;
```

```
    for a = 1 : 15
        b_h(a ,i) = b(a);
    end

    %Caculate the RMS value
    r(1 ,i) = sqrt(mean(e.^2));

    %Check for convergence
    if (r(1 ,i) < (0.1/sqrt(2))) && (con == 0)
        disp(['Converged after ',num2str(i),' samples (mu = 10^(-4))']);
        con = 1;
    end
end

disp(['Min RMS value : ' ,num2str(min(r)),' (mu = 10^(-4))']);

% Plot prediction error (r) vs. sample index (n)
figure(4);
plot(1 : length(n),r);
xlabel('Sample index');
ylabel('Prediction error (RMS)');
xlim([1 8000]);
title('Prediction error vs. sample index (mu = 1e-4)');

% Plot filter coefficients vs. sample index
figure(5);
plot(1 : length(n),b_h);
xlabel('Sample index');
ylabel('Coefficient value');
xlim([1 8000]);
title('Filter coefficients vs. sample index (mu = 1e-4)');

% Apply 64-point FFT to impulse response of converged filter
b_fft = fft([b;zeros(49,1)],64);
f = linspace(0,1,64/2+1)*0.5;
mag_response = abs(b_fft(1:64/2+1));
phase_response = angle(b_fft(1:64/2+1));
```

```
% Plot magnitude response of filter
figure(6);
plot(f,mag_response);
xlabel('Normalized frequency');
ylabel('Magnitude');
title('Magnitude response of filter(mu = 1e-4)');
```

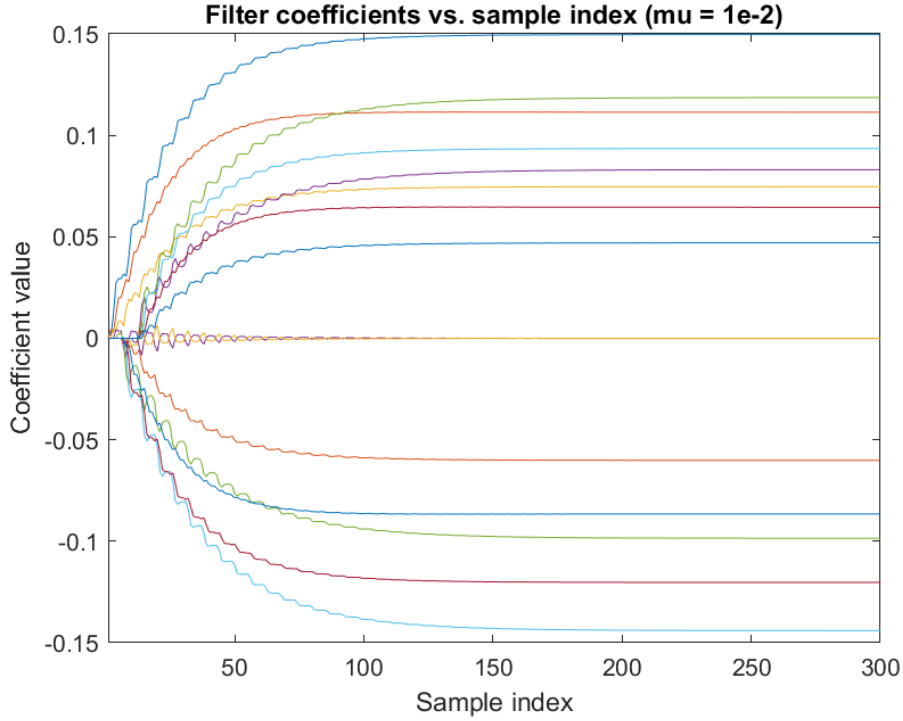
Ans:

r vs. n $(\mu = 10^{-2})$



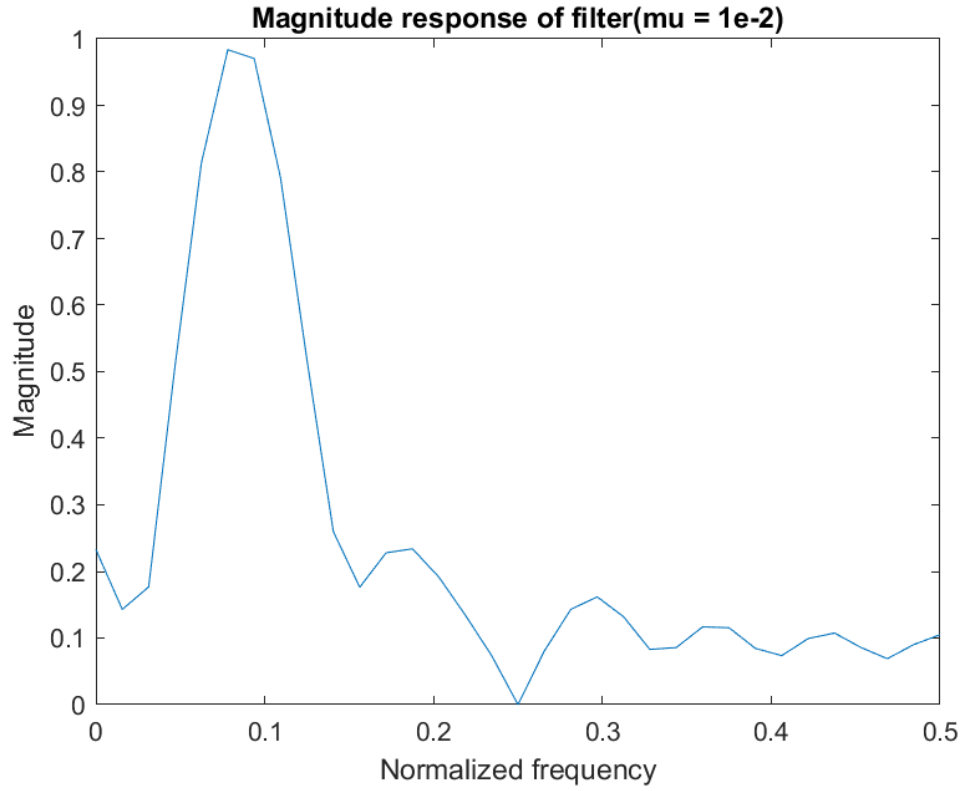

```
Converged after 73 samples (mu = 10^(-2))
Min RMS value : 1.5409e-14 (mu = 10^(-2))
```

b_i vs. n $(\mu = 10^{-2})$

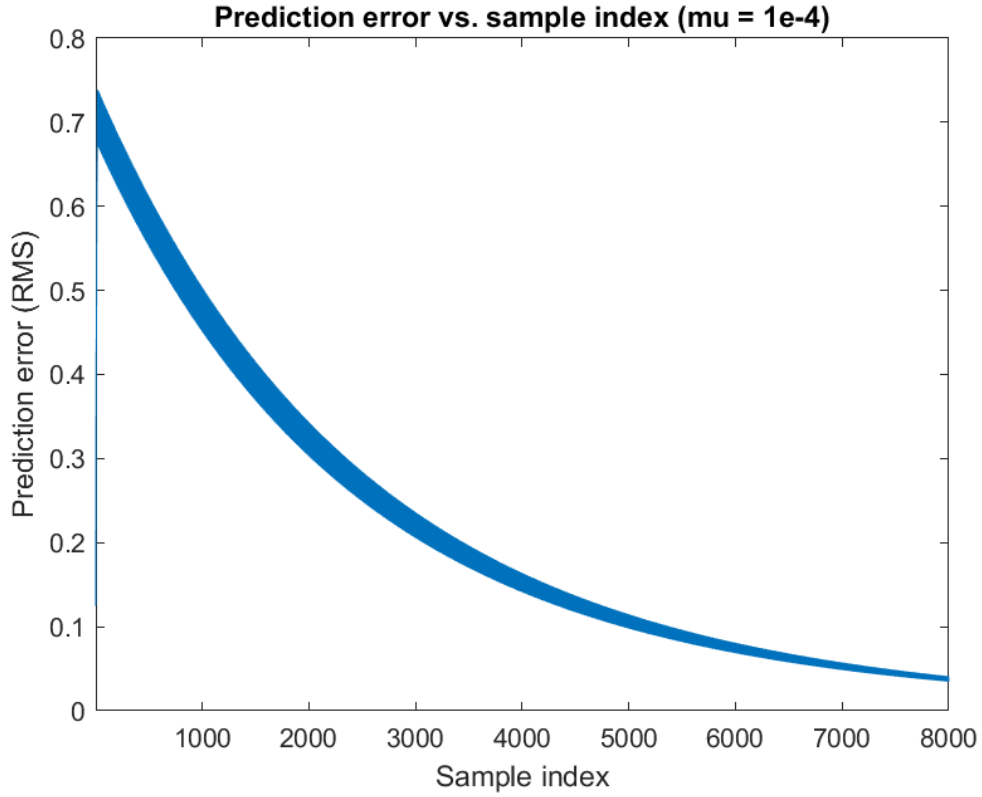


在經過一段時間後系數幾乎保持不變

FFT $(\mu = 10^{-2})$

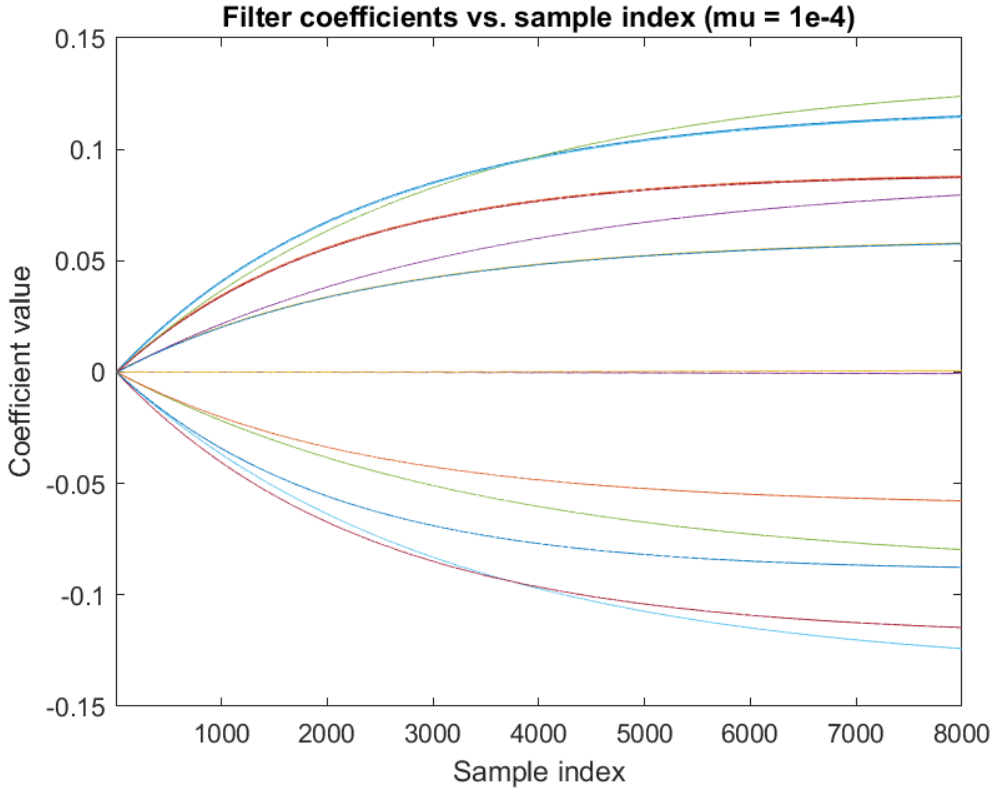


r vs. n $(\mu = 10^{-4})$

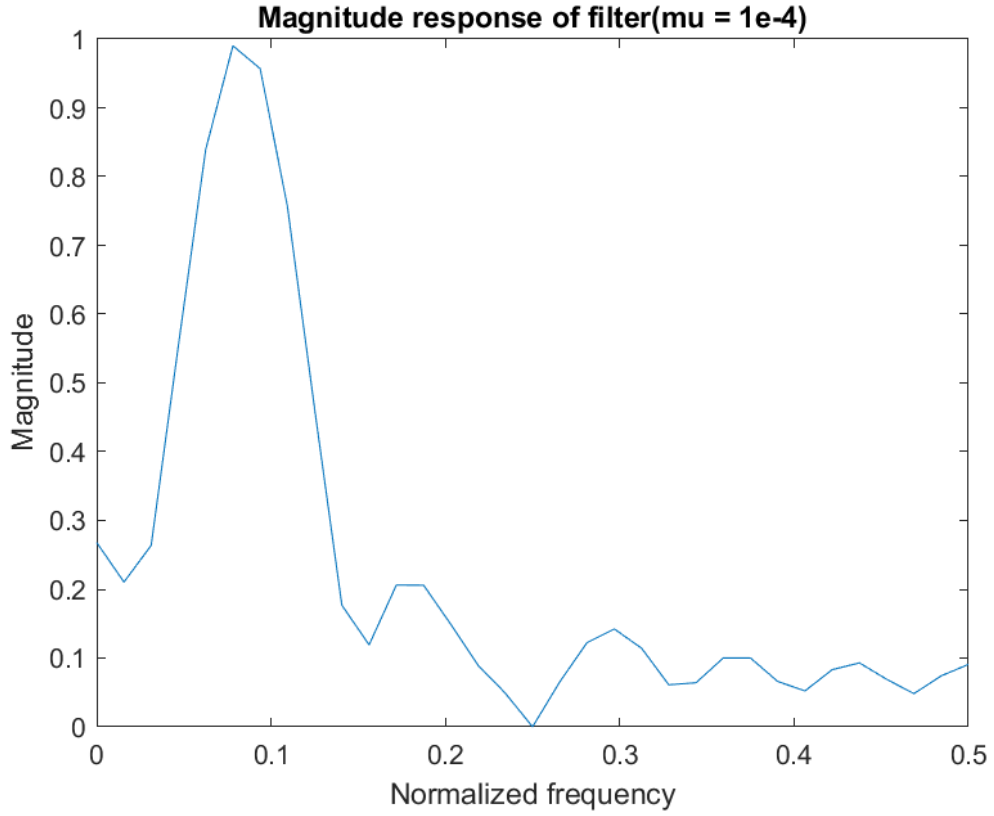



```
Converged after 5911 samples (mu = 10^(-4))
Min RMS value : 1.2358e-12 (mu = 10^(-4))
```

b_i vs. n $(\mu = 10^{-4})$



FFT $(\mu = 10^{-4})$



After changing the step size, the filter will converge more slowly, and RMS will also decrease more slowly.

Q2

Matlab code:

```
clear all
clc
clf

%Read image
test_image = double(imread('HW2_test_image.bmp'));

%Define output result matrix
DWT_out = zeros(512,512);

%Define image process length
L = 512;

%DWT
[out_l ,out_h] = DWT_row_processing(L ,test_image);
[out_ll ,out_hl ,out_lh ,out_hh] = DWT_column_processing(L ,out_l ,out_h);
DWT_out(1 : 256 ,257 : 512)   = out_hl;
DWT_out(257 : 512 ,1 : 256)   = out_lh;
DWT_out(257 : 512 ,257 : 512) = out_hh;

[out_l_2 ,out_h_2] = DWT_row_processing(256 ,out_ll);
[out_ll_2 ,out_hl_2 ,out_lh_2 ,out_hh_2] =
DWT_column_processing(256 ,out_l_2 ,out_h_2);
DWT_out(1 : 128 ,129 : 256)   = out_hl_2;
DWT_out(129 : 256 ,1 : 128)   = out_lh_2;
DWT_out(129 : 256 ,129 : 256) = out_hh_2;

[out_l_3 ,out_h_3] = DWT_row_processing(128 ,out_ll_2);
[out_ll_3 ,out_hl_3 ,out_lh_3 ,out_hh_3] =
DWT_column_processing(128 ,out_l_3 ,out_h_3);
DWT_out(1 : 64 ,1 : 64)     = out_ll_3;
DWT_out(1 : 64 ,65 : 128)   = out_hl_3;
DWT_out(65 : 128 ,1 : 64)   = out_lh_3;
DWT_out(65 : 128 ,65 : 128) = out_hh_3;
```

```
%IDWT
[I_out_l_2 ,I_out_h_2] =
IDWT_column_processing(128 ,out_ll_3 ,out_lh_3 ,out_hl_3 ,out_hh_3);
[inv_pic_2] = IDWT_row_processing(128 ,I_out_l_2 ,I_out_h_2);

[I_out_l_1 ,I_out_h_1] =
IDWT_column_processing(256 ,inv_pic_2 ,out_lh_2 ,out_hl_2 ,out_hh_2);
[inv_pic_1] = IDWT_row_processing(256 ,I_out_l_1 ,I_out_h_1);

[I_out_l ,I_out_h] =
IDWT_column_processing(L ,inv_pic_1 ,out_lh ,out_hl ,out_hh);
[inv_pic] = IDWT_row_processing(L ,I_out_l ,I_out_h);

[I_out_l_0 ,I_out_h_0] = IDWT_column_processing(L ,inv_pic_1 ,0 ,0 ,0);
[inv_pic_b] = IDWT_row_processing(L ,I_out_l_0 ,I_out_h_0);

%PSNR
MSE = 0;
for i = 1 : 512
    for j = 1 : 512
         MSE = MSE + ((test_image(i ,j) - inv_pic(i ,j)) ^ 2);
    end
end
MSE = MSE / (512 ^ 2);
PSNR = 10 * (log10((255 ^ 2) / MSE));
disp(['(a) PSNR = ',num2str(PSNR) ,' dB']);

MSE_b = 0;
for i = 1 : 512
    for j = 1 : 512
         MSE_b = MSE_b + ((test_image(i ,j) - inv_pic_b(i ,j)) ^ 2);
    end
end
MSE_b = MSE_b / (512 ^ 2);
PSNR_b = 10 * (log10((255 ^ 2) / MSE_b));
disp(['(b) PSNR = ',num2str(PSNR_b) ,' dB']);

%Display image after processing
```

```
figure(1)
imshow(mat2gray(test_image));

figure(2)
imshow(mat2gray(DWT_out));

figure(3)
imshow(mat2gray(inv_pic));

figure(4)
imshow(mat2gray(inv_pic_b));
```

```
function [out_l ,out_h] = DWT_row_processing(L ,pic)

%Filter coefficients
h = [ 0.037828455507;
     -0.023849465020;
     -0.110624404418;
      0.377402855613;
      0.852698679009;
      0.377402855613;
     -0.110624404418;
     -0.023849465020;
      0.037828455507];

g = [-0.064538882629;
      0.040689417609;
      0.418092273222;
     -0.788485616406;
      0.418092273222;
      0.040689417609;
     -0.064538882629];

%Symmetric extension at picture boundary
p_l = zeros(L ,L + 8);
p_l = [pic( : ,5) pic( : ,4) pic( : ,3) pic( : ,2) pic pic( : ,L - 1) pic( : ,L -
2) pic( : ,L - 3) pic( : ,L - 4)];
p_h = zeros(L ,L + 6);
```

```
p_h = [pic( : ,4) pic( : ,3) pic( : ,2) pic pic( : ,L - 1) pic( : ,L - 2)
pic( : ,L - 3)];

%Compute output picture
for i = 1 : L
    %Lowpass filter
    temp_l  = conv(p_l(i , :) ,h);
    out_l(i , 1 : (L / 2)) = temp_l(1 ,9 : 2 : (L + 7));

    %Highpass filter
    temp_h  = conv(p_h(i , :) ,g);
    out_h(i , 1 : (L / 2)) = temp_h(1 ,8 : 2 : (L + 6));
end

end
```

```
function [out_ll ,out_hl ,out_lh ,out_hh] =
DWT_column_processing(L ,input_l ,input_h)

%Filter coefficients
h = [ 0.037828455507;
     -0.023849465020;
     -0.110624404418;
      0.377402855613;
      0.852698679009;
      0.377402855613;
     -0.110624404418;
     -0.023849465020;
      0.037828455507];

g = [-0.064538882629;
      0.040689417609;
      0.418092273222;
     -0.788485616406;
      0.418092273222;
      0.040689417609;
     -0.064538882629];

%Symmetric extension at picture boundary
```

```
input_l_extension_for_l = zeros(L + 8 ,L / 2);
input_l_extension_for_l = [input_l(5 , : ); input_l(4 , : ); input_l(3 , : );
input_l(2 , : ); input_l; input_l(L - 1 , : ); input_l(L - 2 , : ); input_l(L -
3, : ); input_l(L - 4, : )];
input_l_extension_for_h = zeros(L + 6 ,L / 2);
input_l_extension_for_h = [input_l(4 , : ); input_l(3 , : ); input_l(2 , : );
input_l; input_l(L - 1 , : ); input_l(L - 2 , : ); input_l(L - 3, : )];
input_h_extension_for_l = zeros(L + 8 ,L / 2);
input_h_extension_for_l = [input_h(5 , : ); input_h(4 , : ); input_h(3 , : );
input_h(2 , : ); input_h; input_h(L - 1 , : ); input_h(L - 2 , : ); input_h(L -
3, : ); input_h(L - 4, : )];
input_h_extension_for_h = zeros(L + 6 ,L / 2);
input_h_extension_for_h = [input_h(4 , : ); input_h(3 , : ); input_h(2 , : );
input_h; input_h(L - 1 , : ); input_h(L - 2 , : ); input_h(L - 3, : )];

%Compute output picture
for i = 1 : L/2
    %Lowpass filter
    temp_ll = conv(input_l_extension_for_l( : ,i) ,h);
    out_ll(1 : (L / 2) ,i) = temp_ll(9 : 2 : (L + 7) ,1);

    temp_hl = conv(input_h_extension_for_l( : ,i) ,h);
    out_hl(1 : (L / 2) ,i) = temp_hl(9 : 2 : (L + 7) ,1);

    %Highpass filter
    temp_lh = conv(input_l_extension_for_h( : ,i) ,g);
    out_lh(1 : (L / 2) ,i) = temp_lh(8 : 2 : (L + 6) ,1);

    temp_hh = conv(input_h_extension_for_h( : ,i) ,g);
    out_hh(1 : (L / 2) ,i) = temp_hh(8 : 2 : (L + 6) ,1);
end

end
```

```
function [out_l ,out_h] =
IDWT_column_processing(L ,input_ll ,input_lh ,input_hl ,input_hh)

%Filter coefficients
q = [-0.064538882629;
```

```
      -0.040689417609;
       0.418092273222;
       0.788485616406;
       0.418092273222;
      -0.040689417609;
      -0.064538882629];

p = [-0.037828455507;
      -0.023849465020;
      0.110624404418;
      0.377402855613;
      -0.852698679009;
       0.377402855613;
       0.110624404418;
      -0.023849465020;
      -0.037828455507];

%Symmetric extension at picture boundary and up-sampling
in_ll = zeros(L + 6 ,L / 2);
in_ll(4 : 2 : (L + 2) , : ) = input_ll;
in_ll(1 : 3 , : ) = [in_ll(7 , : ); in_ll(6 , : ); in_ll(5 , : )];
in_ll((L + 4) : (L + 6) , : ) = [in_ll(L + 2 , : ); in_ll(L + 1 , : ); in_ll(L +
0 , : )];

in_hl = zeros(L + 6 ,L / 2);
in_hl(4 : 2 : (L + 2) , : ) = input_hl;
in_hl(1 : 3 , : ) = [in_hl(7 , : ); in_hl(6 , : ); in_hl(5 , : )];
in_hl((L + 4) : (L + 6) , : ) = [in_hl(L + 2 , : ); in_hl(L + 1 , : ); in_hl(L +
0 , : )];

in_lh = zeros(L + 8 ,L / 2);
in_lh(6 : 2 : (L + 4) , : ) = input_lh;
in_lh(1 : 4 , : ) = [in_lh(9 , : ); in_lh(8 , : ); in_lh(7 , : ); in_lh(6 , : )];
in_lh((L + 5) : (L + 8) , : ) = [in_lh(L + 3 , : ); in_lh(L + 2 , : ); in_lh(L +
1 , : ); in_lh(L - 0 , : )];

in_hh = zeros(L + 8 ,L / 2);
in_hh(6 : 2 : (L + 4) , : ) = input_hh;
```

```
in_hh(1 : 4 , : ) = [in_hh(9 , : ); in_hh(8 , : ); in_hh(7 , : ); in_hh(6 , : )];
in_hh((L + 5) : (L + 8) , : ) = [in_hh(L + 3 , : ); in_hh(L + 2 , : ); in_hh(L +
1 , : ); in_hh(L - 0 , : )];

%Compute output picture
for i = 1 : (L / 2)
    temp_ll = conv(in_ll( : ,i) ,q);
    out_ll(1 : L ,i) = temp_ll(7 : L + 6 ,1);

    temp_hl = conv(in_hl( : ,i) ,q);
    out_hl(1 : L ,i) = temp_hl(7 : L + 6 ,1);

    temp_lh = conv(in_lh( : ,i) ,p);
    out_lh(1 : L ,i) = temp_lh(9 : L + 8 ,1);

    temp_hh = conv(in_hh( : ,i) ,p);
    out_hh(1 : L ,i) = temp_hh(9 : L + 8 ,1);
end

out_l = out_ll + out_lh;
out_h = out_hl + out_hh;

end
```

```
function [pic] = IDWT_row_processing(L ,input_l ,input_h)

%Filter coefficients
q = [-0.064538882629;
     -0.040689417609;
      0.418092273222;
      0.788485616406;
      0.418092273222;
     -0.040689417609;
     -0.064538882629];

p = [-0.037828455507;
     -0.023849465020;
      0.110624404418;
      0.377402855613;
```

```
    -0.852698679009;
     0.377402855613;
     0.110624404418;
    -0.023849465020;
    -0.037828455507];

%Symmetric extension at picture boundary and up-sampling
in_l = zeros(L ,L + 6);
in_l( : ,4 : 2 : (L + 2)) = input_l;
in_l( : ,1 : 3) = [in_l( : ,7) in_l( : ,6) in_l( : ,5)];
in_l( : ,(L + 4) : (L + 6)) = [in_l( : , L + 2 ) in_l( : , L + 1 ) in_l( : , L +
0 )];

in_h = zeros(L ,L + 8);
in_h( : ,6 : 2 : (L + 4)) = input_h;
in_h( : ,1 : 4) = [in_h( : ,9) in_h( : ,8) in_h( : ,7) in_h( : ,6)];
in_h( : ,(L + 5) : (L + 8)) = [in_h( : , L + 3 ) in_h( : , L + 2 ) in_h( : , L +
1 ) in_h( : , L + 0 )];

%Compute output picture
for i = 1 : L
    temp_l = conv(in_l(i , : ) ,q);
    out_l(i ,1 : L)  = temp_l(1 ,7 : L + 6);

    temp_h = conv(in_h(i , : ) ,p);
    out_h(i ,1 : L)  = temp_h(1 ,9 : L + 8);
end

pic = out_l + out_h;

end
```

Ans:

原圖：

After DWT

a) After IDWT

(a) PSNR = 234.2033 dB

b) After IDWT

(b) PSNR = 23.2903 dB