**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** |
| **bi vs. n**    在經過一段時間後系數幾乎保持不變 |
| **FFT** |
| **r vs. n** |
| **bi vs. n** |
| **FFT** |
| 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 |
| 1. After IDWT |
| 1. After IDWT |