## HOMEWORK #4 SOLUTIONS

(Note: Matlab code appears at the end.)

1. (a) *Analysis:* We begin by writing the output of the $i^{th}$ analysis filter as

$$\begin{aligned} x_i(n) &= \sum_{r=0}^{M-1} h_r \bar{x}_i(n-r) \quad \text{where} \quad \bar{x}_i(n) \;=\; x(n)e^{-j\frac{2\pi}{N}in} \\ &= \sum_{r=0}^{M-1} h_r x(n-r) e^{-j\frac{2\pi}{N}i(n-r)} \\ &= e^{-j\frac{2\pi}{N}in} \sum_{r=0}^{M-1} h_r x(n-r) e^{j\frac{2\pi}{N}ir}. \end{aligned}$$

Downsampling $x_i(n)$ by the factor $N$ gives

$$\begin{aligned} s_i(m) &= x_i(mN) \\ &= \underbrace{e^{-j\frac{2\pi}{N}imN}}_{=1\ \forall i,m} \sum_{r=0}^{M-1} h_r x(mN-r) e^{j\frac{2\pi}{N}ir} \\ &= \sum_{\ell=0}^{\frac{M}{N}-1} \sum_{p=0}^{N-1} h_{\ell N+p}\, x(mN-\ell N-p) e^{j\frac{2\pi}{N}i(\ell N+p)} \\ &\qquad \text{using} \quad r = \ell N + p \quad \text{for} \quad 0 \le p \le N-1, \\ &= \sum_{\ell=0}^{\frac{M}{N}-1} \underbrace{e^{j\frac{2\pi}{N}i\ell N}}_{=1\ \forall i,\ell} \sum_{p=0}^{N-1} h_\ell^{(p)} x^{(p)}(m-\ell) e^{j\frac{2\pi}{N}ip} \\ &\qquad \text{where} \quad \begin{cases} h_\ell^{(p)} \;:=\; h_{\ell N+p} \\ x^{(p)}(\ell) \;:=\; x(\ell N - p) \end{cases} \\ &= \sum_{p=0}^{N-1} \underbrace{\left( \sum_{\ell=0}^{\frac{M}{N}-1} h_\ell^{(p)} x^{(p)}(m-\ell) \right)}_{h_m^{(p)} * x^{(p)}(m)} e^{j\frac{2\pi}{N}ip} \\ &= \sum_{p=0}^{N-1} w^{(p)}(m) e^{j\frac{2\pi}{N}ip} \\ &= \underbrace{\begin{pmatrix} 1 & e^{j\frac{2\pi}{N}i} & \cdots & e^{j\frac{2\pi}{N}i(N-1)} \end{pmatrix}}_{i^{th} \text{ row of } \sqrt{N}\mathbf{W}_N^*} \begin{pmatrix} w^{(0)}(m) \\ w^{(1)}(m) \\ \vdots \\ w^{(N-1)}(m) \end{pmatrix}, \end{aligned}$$

as in the block diagram, thus we have confirmed the equivalence of the two analysis banks.

(b) <u>*Synthesis:*</u> The upsampled version of the $i^{th}$ sub-band input can be written

$$y_i(n) \;=\; \begin{cases} s_i(n/N) & n/N \in \mathbb{Z} \\ 0 & n/N \notin \mathbb{Z}. \end{cases}$$

Filtering the upsampled signal,

$$\begin{aligned} \bar{y}_i(n) &= \sum_{r=0}^{M-1} k_r y_i(n-r) \\ &= \sum_{\substack{r=0,\ldots,M-1 \\ r:\frac{n-r}{N}\in\mathbb{Z}}} k_r\, s_i\Big(\frac{n-r}{N}\Big). \end{aligned}$$

Modulating and summing the filtered signals,

$$\begin{aligned} u(n) &= \sum_{i=0}^{N-1} u_i(n) \\ &= \sum_{i=0}^{N-1} \bar{y}_i(n) e^{j\frac{2\pi}{N}in} \\ &= \sum_{i=0}^{N-1} \sum_{\substack{r=0,\ldots,M-1 \\ r:\frac{n-r}{N}\in\mathbb{Z}}} k_r\, s_i\Big(\frac{n-r}{N}\Big)\, e^{j\frac{2\pi}{N}in} \end{aligned}$$

The substitutions

$$\begin{aligned} n = mN + p &\quad \text{for} \quad 0 \le p \le N-1 \\ r = \ell N + q &\quad \text{for} \quad 0 \le q \le N-1, \end{aligned}$$

give

$$\begin{aligned} u(mN+p) &= \sum_{i=0}^{N-1} \sum_{\substack{\ell=0,\ldots,\frac{M}{N}-1 \\ q=0,\ldots,N-1 \\ \ell,q:\frac{mN+p-\ell N-q}{N}\in\mathbb{Z}}} k_{\ell N+q}\, s_i\Big(\frac{mN+p-\ell N-q}{N}\Big)\, e^{j\frac{2\pi}{N}i(mN+p)} \\ &= \sum_{i=0}^{N-1} \underbrace{e^{j\frac{2\pi}{N}imN}}_{=1\ \forall\, i,m} \sum_{\ell=0}^{\frac{M}{N}-1} \sum_{\substack{q=0,\ldots,N-1 \\ \ell,q:\frac{p-q}{N}\in\mathbb{Z}}} k_{\ell N+q}\, s_i\Big(m-\ell+\frac{p-q}{N}\Big)\, e^{j\frac{2\pi}{N}ip}. \end{aligned}$$

Due to the limited range of $p$ and $q$, the only time that $\frac{p-q}{N} \in \mathbb{Z}$ is when $q = p$, so that

$$\begin{aligned} u(mN+p) &= \sum_{i=0}^{N-1} \sum_{\ell=0}^{\frac{M}{N}-1} k_{\ell N+p}\, s_i(m-\ell)\, e^{j\frac{2\pi}{N}ip} \\ &= \sum_{\ell=0}^{\frac{M}{N}-1} k_{\ell N+p} \underbrace{\left(\sum_{i=0}^{N-1} s_i(m-\ell)\, e^{j\frac{2\pi}{N}ip}\right)}_{v^{(p)}(m-\ell)}. \end{aligned}$$

As in the block diagram, $v^{(p)}(m)$ is calculated via an inner product between the $p^{th}$ row of $\sqrt{N}\mathbf{W}_N^*$ and the sub-band inputs at block index $m$, i.e., $\{s_0(m), s_1(m), \ldots, s_{N-1}(m)\}$.

Noting from the diagram that the $p^{th}$ polyphase branch contributes the $p^{th}$ sample of every $N$-block of outputs, i.e.,

$$\big\{u(mN), u(mN+1), \dots, u(mN+N-1)\big\} \;=\; \big\{u^{(0)}(m), u^{(1)}(m), \dots, u^{(N-1)}(m)\big\},$$

we claim $u(mN+p) = u^{(p)}(m)$. Using this and the definition $k_\ell^{(p)} := k_{\ell N+p}$, we have

$$\begin{aligned} u^{(p)}(m) &= \sum_{\ell=0}^{\frac{M}{N}-1} k_\ell^{(p)} v^{(p)}(m-\ell) \\ &= k_m^{(p)} * v^{(p)}(m), \end{aligned}$$

as in the diagram, which confirms the equivalence of the synthesis filterbanks.

(c) Implementing the two systems gives the output in Fig. 1.

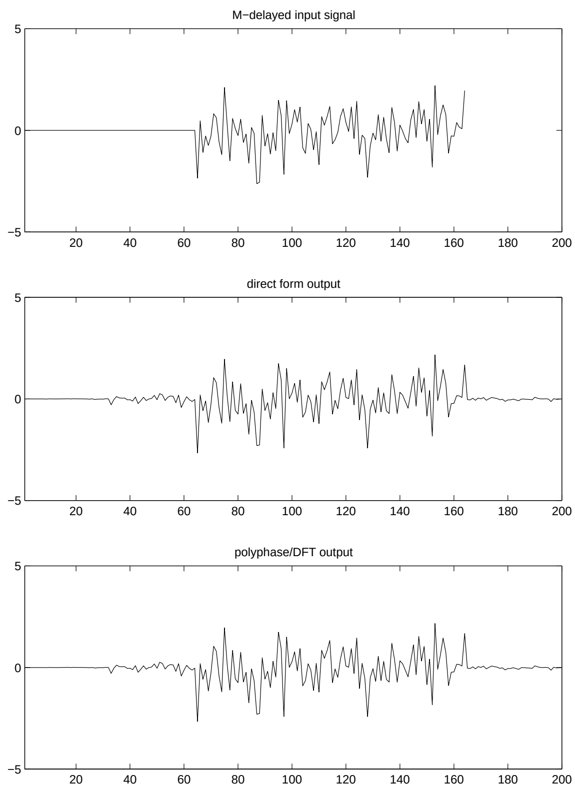


Figure 1: Synthesis filterbank outputs.

2. (a) Using the fact that the source process is unit variance and white:

$$\begin{aligned}
\sigma_e^2 &= \mathrm{E}\left\{|u(n) - x(n-M)|^2\right\} \\
&= \mathrm{E}\left\{\left|\sum_{k=0}^{2M-1} q_k x(n-k) - x(n-M)\right|^2\right\} \\
&= \mathrm{E}\left\{\left|\sum_{k=0}^{2M-1} \bar{q}_k x(n-k)\right|^2\right\} \quad \text{where} \quad \bar{q}_k = \begin{cases} q_k & k \neq M \\ q_k - 1 & k = M \end{cases} \\
&= \mathrm{E}\left\{\sum_{\ell=0}^{2M-1}\sum_{k=0}^{2M-1} \bar{q}_k \bar{q}_\ell x(n-k)x(n-\ell)\right\} \\
&= \sum_{\ell=0}^{2M-1}\sum_{k=0}^{2M-1} \bar{q}_k \bar{q}_\ell \underbrace{\mathrm{E}\{x(n-k)x(n-\ell)\}}_{=\begin{cases} 1 & k=\ell \\ 0 & k \neq \ell \end{cases}} \\
&= \sum_{k=0}^{2M-1} \bar{q}_k^2
\end{aligned}$$

We could go further by substituting the definition of $\bar{q}_k$

$$\sigma_e^2 = (q_M - 1)^2 + \sum_{k \neq M} q_k^2 = q_M^2 - 2q_M + 1 + \sum_{k \neq M} q_k^2 = 1 - 2q_M + \sum_{k=0}^{2M-1} q_k^2.$$

(b) Fig. 2 shows the prototype filters and composite system DTFTs for the MPEG filter. Using the formula from part (a), we calculated $\boxed{\sigma_e^2 = 7.4905 \times 10^{-9}}$. Note that the ideal prototype filter is lowpass with cutoff at $\omega = \pi/2N$ and DC gain of $\sqrt{N}$.

(c) Fig. 3 shows the prototype filters and composite system DTFTs for the `remez`-designed filter. The exact command usage was

```
h = remez(M-1,[0,0.3242/(2*N),2/(2*N),1],[sqrt(N),sqrt(N),0,0]);
```

Using the formula from part (a), we calculated $\boxed{\sigma_e^2 = 4.7828 \times 10^{-7}}$.

(d) Fig. 4 shows the prototype filters and composite system DTFTs for the `firls`-designed filter. The exact command usage was

```
h = firls(M-1,[0,0.3797/(2*N),2/(2*N),1],[sqrt(N),sqrt(N),0,0]);
```

Using the formula from part (a), we calculated $\boxed{\sigma_e^2 = 3.6494 \times 10^{-6}}$.

3. (a) The polyphase/DCT implementation of the MPEG filterbank gave the output and reconstruction error shown in Fig. 5.

(b) The computed value of MSRE was $\boxed{7.5212 \times 10^{-9}}$, which is very close to the value of $\sigma_e^2$ computed for the same filter coefficients.

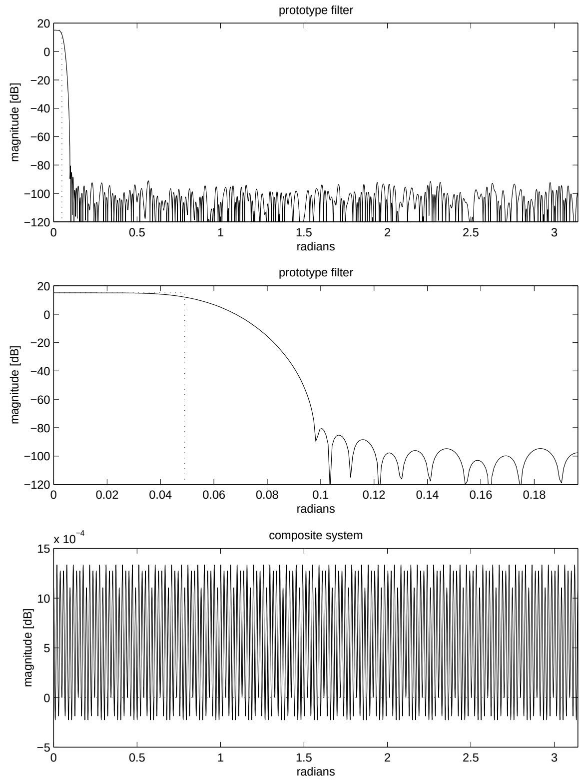


Figure 2: MPEG Prototype filter and composite system DTFT magnitude responses.

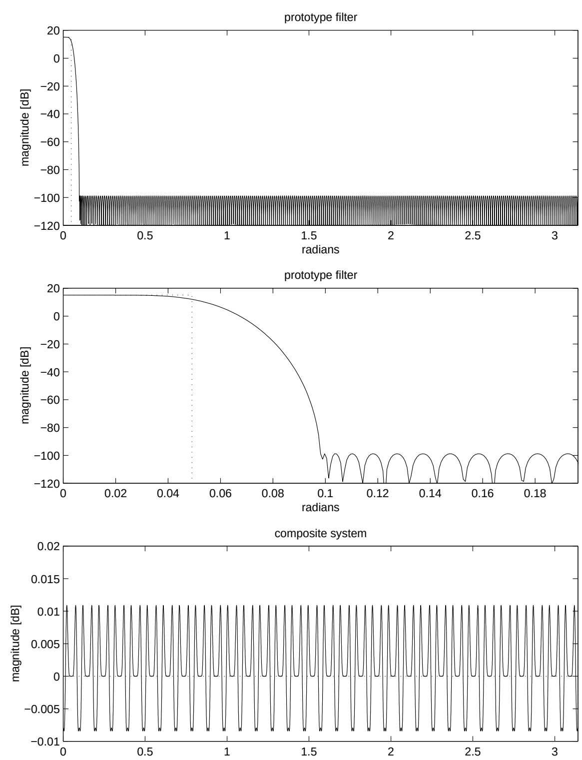


Figure 3: `remez`-designed prototype filter and composite system DTFT magnitude responses.

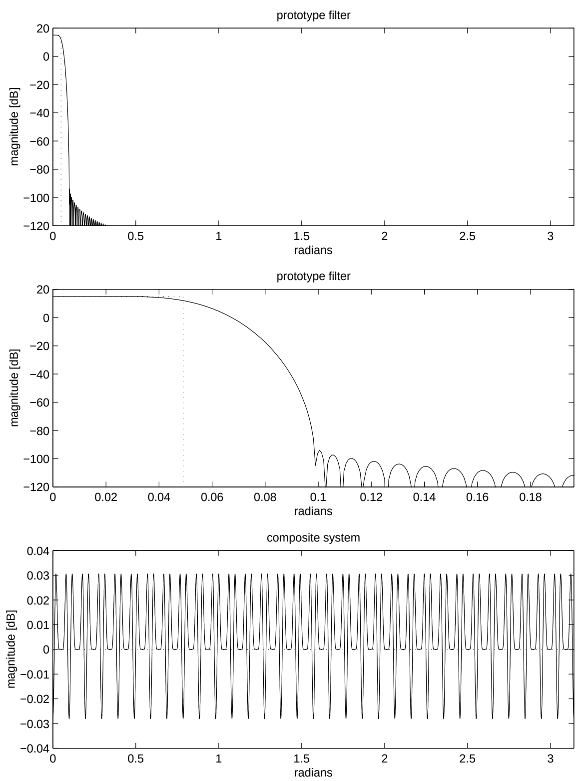


Figure 4: `firls`-designed prototype filter and composite system DTFT magnitude responses.

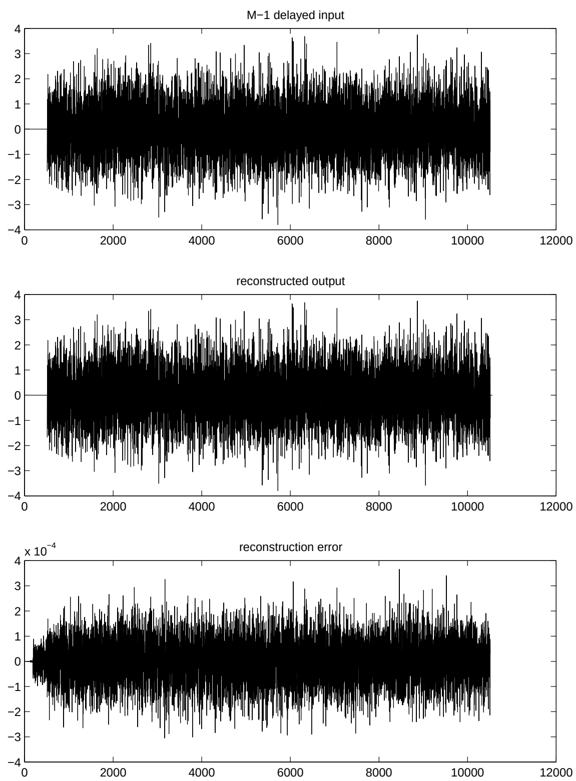


Figure 5: ($M$-delayed) input, output, and reconstruction error for the MPEG filterbank.

## Matlab code for Problem 1:

```
% implements DFT filter bank

 N = 8;  % DFT length
 M = 8*N; % filter length

 % input signal
 L = 100;
 x = randn(1,L);

 % filter
 %h = intfilt(N,(M+1)/2/N,0.95); h = sqrt(N)*h/sum(h);
 %load h_8; M=length(h);
 h = remez(M-1,[0,.8/N,1.2/N,1],[sqrt(N),sqrt(N),0,0]);
 plot(20*log10(abs(fft(h,512))));
 P = zeros(N,M/N);
 P(:) = h; % polyphase filter matrix

 % direct analysis
 Sd = zeros(N,ceil((M+L)/N));
 for i=0:N-1,
   xbarm = x.*exp(-j*2*pi/N*i*[0:L-1]);
   xm = conv(h,xbarm);
   sm = xm([1:N:length(xm)]); % subband outputs
   Sd(i+1,:) = [sm,zeros(1,ceil((M+L)/N)-length(sm))]; % column index = time
 end;

 % direct synthesis
 sm = zeros(size(Sd,2)*N,1);
 ud = zeros(length(sm)+M-1,1);
 for i=0:N-1,
   sm(1:N:length(sm)) = Sd(i+1,:);
   ud = ud + real( conv(sm,h).*exp(j*2*pi/N*i*[0:length(sm)+M-2]).' );
 end;

 % fast analysis
 xp = [zeros(1,N-1),x]; % zero-pad
 X = zeros(N,ceil(length(xp)/N));
 for m=1:ceil(length(xp)/N),
   if m<ceil(length(xp)/N),
     X(:,m) = xp((m-1)*N+[N:-1:1]).';
   else
     tmp = xp(length(xp):-1:(m-1)*N+1).';
     X([N-length(tmp)+1:N],m) = tmp;
   end;
 end;
 Wf = zeros(N,ceil(length(xp)/N)+M/N-1);
 for i=0:N-1,
   Wf(i+1,:) = conv(P(i+1,:),X(i+1,:));
 end;
 Sf = conj(dftmtx(N))*Wf; % subband outputs, column index = time

 % fast synthesis
 Vf = real(conj(dftmtx(N))*Sf);
 Uf = zeros(N,size(Vf,2)+M/N-1);
 for i=0:N-1,
   Uf(i+1,:) = conv(Vf(i+1,:),P(i+1,:));
 end;
 uf = Uf(:);

 % comparison
 err = norm(uf-ud(1:length(uf)))
 subplot(311);
   plot([zeros(1,M),x]);  % zero-padded for comparison
   axis([1,200,-5,5]);
   title('M-delayed input signal');
 subplot(312);
   plot((ud),'g');
   axis([1,200,-5,5]);
   title('direct form output');
 subplot(313);
   plot((uf),'r');
   axis([1,200,-5,5]);
   title('polyphase/DFT output');
 orient tall;
```

## Matlab code for Problem 2:

```
% MPEG filter design

 N = 32;  % number of bands (MPEG:32)
 M = 16*N+1; % filter length (MPEG:16)

 % filter
 h = remez(M-1,[0,0.3242/(2*N),2/(2*N),1],[sqrt(N),sqrt(N),0,0]);
 %h = firls(M-1,[0,0.3797/(2*N),2/(2*N),1],[sqrt(N),sqrt(N),0,0]);
 %load h_mpeg.mat; % stores in h

 % coefs
 aa = zeros(1,N); cc = zeros(1,N);
 for i=0:N-1,
   aa(i+1) = exp(-j*pi*(M+N-1)/(4*N)*(2*i+1));
   cc(i+1) = exp(-j*pi*(M-N-1)/(4*N)*(2*i+1));
 end;

 % transfer function
 n = [0:2*M-2];
 q = zeros(1,2*M-1);
 for i=0:N-1,
   q = q + real(aa(i+1)*cc(i+1))*cos(pi*(2*i+1)/2/N*n)...
    - imag(aa(i+1)*cc(i+1))*sin(pi*(2*i+1)/2/N*n);
 end;
 q = 2/N*q.*conv(h,h);

 % error power
 e = [q(1:M-1),q(M)-1,q(M+1:2*M-1)];
 MSRE = e*e'

 % plot
 figure(1);
 subplot(311);
   Mh_fft = max( 2^(round(log2(M)+4)), 512);
   H = 20*log10(abs(fft(h,Mh_fft))); H = H(1:Mh_fft/2+1);
   plot(linspace(0,pi,Mh_fft/2+1),H);
   hold on;
     plot([0,pi/N/2,pi/N/2],[20*log10(sqrt(N))*[1,1],min(H)],'r:');
   hold off;
   title('prototype filter');
   xlabel('radians');
   ylabel('magnitude [dB]');
   axis([0,pi,-120,20*log10(sqrt(N))+5]);
 subplot(312);
   plot(linspace(0,pi,Mh_fft/2+1),H);
   hold on;
     plot([0,pi/N/2,pi/N/2],[20*log10(sqrt(N))*[1,1],min(H)],'r:');
   hold off;
   title('prototype filter');
   xlabel('radians');
   ylabel('magnitude [dB]');
   axis([0,2*pi/N,-120,20*log10(sqrt(N))+5]);
 subplot(313);
   Mq_fft = max( 2^(round(log2(2*M-1)+4)), 512);
   Q = 20*log10(abs(fft(q,Mq_fft))); Q = Q(1:Mq_fft/2+1);
   plot(linspace(0,pi,Mq_fft/2+1),Q);
   hold on;
     plot([0,pi],[0,0],'r:');
   hold off;
   title('composite system');
   xlabel('radians');
   ylabel('magnitude [dB]');
   axe = axis; axis([0,pi,axe(3:4)]);
 orient tall;
```

## Matlab code for Problem 3:

```
% implements MPEG (cosine-modulated) filterbank

 N = 32;
 M = 512;

 % create input and zero-pad
 L = 10000;
 x = randn(1,L);
 xp = [zeros(1,N-1),x,zeros(1,N-mod(L+N-1,N)),zeros(1,M)];

 % create analysis window
 load h_mpeg; % stores in varible "h"
 hw = 2*h(1:512); % last sample is zero
 for k=1:2:M/2/N,
   hw(k*2*N+[1:2*N]) = -hw(k*2*N+[1:2*N]);
 end;

 % create analysis transform
 Ta = zeros(N,2*N);
 for i=0:N-1,
   Ta(i+1,:) = cos(pi*(2*i+1)/2/N*([0:2*N-1]-N/2));
 end;

 % create synthesis transform
 Ts = zeros(2*N,N);
 for i=0:N-1,
   Ts(:,i+1) = cos(pi*(2*i+1)/2/N*([0:2*N-1]+N/2)).';
 end;

 % analysis
 S = zeros(N,ceil(length(xp)/N));
 xx = zeros(1,M);
 wbar = zeros(1,N);
 for m=1:ceil(length(xp)/N),
   % load new input block
   xx(N+1:M) = xx(1:M-N);
   xx(1:N) = xp( (m-1)*N + [N:-1:1] );

   % window using filter coeffs
   xh = xx.*hw;

   % combine
   w = zeros(1,2*N);
   for k=1:2*N,
     w(k) = sum(xh(k:2*N:M));
   end;

   % slow cosine matrix transformation
   %S(:,m) = Ta*w.';

   % fast cosine matrix transformation
   wbar(0+1) = sqrt(2)*w(16+1);
   wbar([1:16]+1) = w(16+[1:16]+1) + w(16-[1:16]+1);
   wbar([17:31]+1) = w(16+[17:31]+1) - w(80-[17:31]+1);
   S(:,m) = sqrt(N/2)*idct(wbar).'; %norm(tmp-S(:,m))
 end;

 % synthesis
 vv = zeros(1,2*M);
```

```
v = zeros(1,M);
vnew = zeros(1,2*N);
u = zeros(1,N*ceil(length(xp)/N));
for m=1:ceil(length(xp)/N),
  % fast cosine matrix transformation
  vbar = sqrt(N/2)*dct(S(:,m)).'; vbar(1) = sqrt(2)*vbar(1);
  vnew([0:15]+1) = vbar(16+[0:15]+1);
  vnew([17:47]+1) = -vbar(48-[17:47]+1);
  vnew([48:63]+1) = -vbar([48:63]-48+1);

  % slow cosine matrix transformation
  %tmp = Ts*S(:,m);  %norm(tmp.'-vnew)

  % insert new input block
  vv(2*N+1:2*M) = vv(1:2*M-2*N);
  vv(1:2*N) = vnew;

  % extract valid sub-blocks
  for k=0:M/2/N-1,
    v(k*2*N+[1:2*N]) = vv(k*4*N+[1:N,3*N+1:4*N]);
  end;

  % window using filter coeffs
  vh = v.*hw;

  % combine
  for k=1:N,
    u((m-1)*N+k) = sum(vh([k:N:M]));
  end;
end;

% error
e = u([1:M+L])-[zeros(1,M),x];
MSRE = e(M+[1:L])*e(M+[1:L])'/L

subplot(311)
  plot([zeros(1,M),x]);
  title('M-delayed input');
subplot(312)
  plot(u);
  title('reconstructed output');
subplot(313)
  plot(e);
  title('reconstruction error');
orient tall;
```