HOMEWORK SOLUTIONS #3

1. Matlab code for problems 1 and 2:

```
% generate bandlimited signal
u = randn(1,1000);
N_g = 51;
B = 0.8;
g = remez(N_g-1,[0,B*0.75,B,1],[1,1,0,0]);
x = conv(u,g);
%x = zeros(size(x)); x(505)=1;

% design polyphase filters
N = 11;
L = 15;
if 0,% hamming window method
  H = zeros(L,N);
  h = zeros(N*L,1);
  for p=1:L,
    tau = ((N*L-1)/2 - (p-1))/L;
    H(p,:) = sinc([0:N-1]-tau);
    H(p,:) = H(p,:).*hamming(N).';
    h(p+[0:N-1]*L) = H(p,:);
  end;
  %H = flipud(polydesign(L,N,B/2)'); H = H(1:L,:); h=H(:);
else % master filter method
  if 2*floor(L/2)==L,  % if L even
    mag = [1,1,zeros(1,L)];
    freq = [0]; for p=0:L/2-1, freq = [freq,(2*p+B)/L,(2*(p+1)-B)/L]; end;
    freq = [freq,1]
  else % if L odd
    mag = [1,1,zeros(1,L-1)];
    freq = [0]; for p=0:(L-1)/2, freq = [freq,(2*p+B)/L,(2*(p+1)-B)/L]; end;
    freq = freq(1:end-1);
  end;
  h = L*remez(N*L-1,freq,mag);
  %h = L*firls(N*L-1,freq,mag);
  %h = L*remez(N*L-1,[0,B/L,(2-B)/L,1],[1,1,0,0]);
  %h = L*firls(N*L-1,[0,B/L,(2-B)/L,1],[1,1,0,0]);
  H = zeros(L,N);
  for p=1:L,
    H(p,:) = h(p:L:end);
  end;
end;

% master filter DTFT
figure(1);
Nf = 2048;
hh = fft(h,Nf);
plot([0:Nf/2-1]/(Nf/2),20*log10(abs(hh(1:Nf/2))));
title('master filter'); ylabel('DTFT magnitude [dB]'); xlabel('omega/pi');

% polyphase filter DTFTs
figure(2);
Nf = 256;
hh = fft(H(1,:),Nf);
plot([0:Nf/2-1]/(Nf/2),20*log10(abs(hh(1:Nf/2))));
hold on;
for p=2:L,
  hh = fft(H(p,:),Nf);
  plot([0:Nf/2-1]/(Nf/2),20*log10(abs(hh(1:Nf/2))));
end;
hold off;
grid on;
title('polyphase filters'); ylabel('DTFT magnitude [dB]'); xlabel('omega/pi');

% polyphase filter DTFTs
figure(3);
Nf = 256;
grpdelay(H(1,:));
hold on;
for p=2:L,
  grpdelay(H(p,:));
end;
hold off;
title('polyphase filters'); ylabel('group delay'); xlabel('omega/pi');

% polyphase filtering
X = zeros(L,length(x)+N-1);
for p=1:L,
  X(p,:) = conv(x,H(p,:));
end;
y = X(:); % concatenate columns of X

% test vs standard interpolation
v = zeros(length(x)*L,1);
v(1:L:end) = x;
y2 = conv(h,v);
%max(abs(y2(1:length(y))-y))

% plot spectra
```

```
M = 4096; % dft length
figure(4);
clf;
subplot(211)
  plot(2*[0:M-1]/M,abs(fft(x,M))); title('filtered signal x[n]');
subplot(212)
  plot(2*[0:M-1]/M,abs(fft(y,M))); title('interpolated signal y[m]');

% plot time domain
figure(5)
n1_x = 500; n2_x = 510;
%m1_y = L*(n1_x+(N-1)/2-1)+1; m2_y = L*(n2_x+(N-1)/2-1)+1;
m1_y = L*n1_x+(L*N-1)/2; m2_y = L*n2_x+(L*N-1)/2;
plot([n1_x:1/L:n2_x],y(1+[m1_y:m2_y]),'ro');
hold on; stem([n1_x:n2_x],x(1+[n1_x:n2_x]),'x'); hold off;
title('polyphase interpolation: original (x) and interpolated (o) values');

%t = linspace(-5,5,1000);
%hold on; plot(t+505,sinc(t),'g'); hold off;
```

2. Matlab code for problem 3:

```
 % generate lowpass signal
 u = randn(1,150);
 BW = 2*300/1750;
 h_u = remez(50,[0,0.5*BW,BW,1],[1,1,0,0]);
 x = conv(u,h_u);
%x = zeros(size(x)); x(50)=1;

 % design master decimation filter
 L = 144;
 M = 175;
 N_h = 10*L;

 %% indirect firls design with all transition bands (too computational!)
 %if 2*floor(L/2)==L,   % if L even
 %  cutoffs_left = [0, ([2:2:L]-BW)/L];
 %  cutoffs_right = [([0:2:L-2]+BW)/L, 1];
 %  A = [L,L,zeros(1,L)];
 %else % L odd
 %  cutoffs_left = [0, ([2:2:L-1]-BW)/L];
 %  cutoffs_right = ([0:2:L-1]+BW)/L;
 %  A = [L,L,zeros(1,L-1)];
 %end;
 %F = zeros(1,2*length(cutoffs_left));
 %F(1:2:end) = cutoffs_left; F(2:2:end) = cutoffs_right;
 %h = firls(N_h-1,F,A);

 %% indirect firls design with only one transition band (too computational)
 %F = [0,BW/L,(2-BW)/L,1];
 %A = [L,L,0,0];
 %h = firls(N_h-1,F,A);

 %% indirect kaiser window design
 %err_dB = -65;
 %F = [BW/L,(2-BW)/L];
 %A = [L,0];
 %dev = [1,1]*10^(err_dB/20);
 %[N_h,Wn,beta,type] = kaiserord(F,A,dev);
 %h = fir1(N_h, Wn, type, kaiser( N_h+1,beta ), 'noscale' );
 %h = h*L; % DC gain not used in fir1 nor kaiserord!

 % direct hamming design
 N_p = ceil(N_h/L);
 N_h = N_p*L;
 hh = zeros(L,N_p);
 h = zeros(N_h,1);
 for p=1:L,
   tau = ((N_h-1)/2 - (p-1))/L;
   hh(p,:) = sinc([0:N_p-1]-tau);
   hh(p,:) = hh(p,:).*hamming(N_p).';
   h(p+[0:N_p-1]*L) = hh(p,:);
 end;

 % plot DTFT of master filter
 figure(1)
 P = 8192; % dft length
 H = fft(h,P); plot(2*[0:P/2-1]/P,20*log10(abs(H(1:P/2))));
 ylabel('dB');
 title('DTFT of master resampling filter (for polyphase resampling)');

 % create polyphase filters
 h = [h,zeros(1,L-mod(length(h)-1,L)-1)];% make length a multiple of L
 N_p = length(h)/L;  % polyphase filter length
 hh = zeros(L,N_p); hh(:) = h;  % store polyphase filter in each row

 % polyphase resampling
 m = 0; % output index
 y = [];
 xx = [zeros(1,N_p-1),x]; % need to pre-zero-pad input for implementing
 while floor(m*M/L)+N_p<length(x),
   p = mod(m*M,L); % index of current polyphase filter (>= 0)
   n = floor(m*M/L)+N_p; % leading index in input data stream
   y = [y,hh(p+1,:)*xx(n-[0:N_p-1]).'];
   m = m+1;
 end;

 % plot spectra
 figure(2)
 subplot(211)
   plot(2*[0:P-1]/P,abs(fft(x,P)),'r'); title('input x[n]');
 subplot(212)
   plot(2*[0:P-1]/P,abs(fft(y,P))); title('resampled output y[m]');
 % plot time domain
 t_start = 28e-6; t_stop = 34e-6;
 t_delay = (((N_h-1)/2)/L)/1.75e6;
 n_start = floor(t_start*1.75e6);
 n_stop = ceil(t_stop*1.75e6);
 m_start = floor((t_start+t_delay)*1.44e6);
 m_stop = ceil((t_stop+t_delay)*1.44e6);
 figure(3)
 stem([n_start:n_stop]/1.75e6,x([n_start:n_stop]+1),'rx');
 hold on;
 plot([m_start:m_stop]/1.44e6-t_delay,y([m_start:m_stop]+1),'o');
 hold off;
 xlabel('seconds');
 title('original (x) and resampled (o) values');

return
 % plot polyphase filters
 figure(4)
 P = 64;
 subplot(121)
   Hp = fft(hh(1,:),P); plot(2*[0:P/2-1]/P,abs(Hp(1:P/2)));
   hold on;
   for p=2:L,
     Hp = fft(hh(p,:),P); plot(2*[0:P/2-1]/P,abs(Hp(1:P/2)));
   end;
   hold off;
   title('polyphase magnitudes');
 subplot(122)
   grpdly = grpdelay(hh(1,:),1,P); plot(2*[0:P/2-1]/P,grpdly(1:P/2));
   hold on;
   for p=2:L,
     grpdly = grpdelay(hh(p,:),1,P); plot(2*[0:P/2-1]/P,grpdly(1:P/2));
   end;
   hold off;
   title('polyphase group delays');
```

3. Matlab code for problem 4:

```
 % generate lowpass signal
 u = randn(1,150);
 BW = 2*300/1750;
 h_u = remez(50,[0,0.5*BW,BW,1],[1,1,0,0]);
 x = conv(u,h_u);
 %x = zeros(size(x)); x(56)=1;

 % design master decimation filter
 L = 10; % polyphase branches
 Q = 144/175; % resampling ratio
 N_h = 10*L;

 % firls design with all transition bands
 if 2*floor(L/2)==L,   % if L even
   cutoffs_left = [0, ([2:2:L]-BW)/L];
   cutoffs_right = [([0:2:L-2]+BW)/L, 1];
   A = [L,L,zeros(1,L)];
 else                  % L odd
   cutoffs_left = [0, ([2:2:L-1]-BW)/L];
   cutoffs_right = ([0:2:L-1]+BW)/L;
   A = [L,L,zeros(1,L-1)];
 end;
 F = zeros(1,2*length(cutoffs_left));
 F(1:2:end) = cutoffs_left; F(2:2:end) = cutoffs_right;
 h = firls(N_h-1,F,A);

 %% kaiser window design
 %err_dB = -65;
 %F = [BW/L,(2-BW)/L];
 %A = [1,0];
 %dev = [1,1]*10^(err_dB/20);
 %[N_h,Wn,beta,type] = kaiserord(F,A,dev);
 %h = fir1(N_h, Wn, type, kaiser( N_h+1,beta ), 'noscale' );
 %h = h*L; % DC gain not used in fir1 nor kaiserord!
 %figure(2)
 %plot(h);
 %title('kaiser-window master filter');

 % plot DTFT of master filter
 figure(3)
 P = 8192; % dft length
 H = fft(h,P); plot(2*[0:P/2-1]/P,20*log10(abs(H(1:P/2))));
 ylabel('dB');
 title('DTFT of master resampling filter (for arb-rate resampler)');

 % create polyphase filters
 %h = [h,zeros(1,L+1-mod(length(h)-1,L+1)-1)];% make length a multiple of L+1
 %N_p = length(h)/(L+1);  % polyphase filter length
 %hh = zeros(L+1,N_p); hh(:) = h;  % store polyphase filter in each row
 h = [h,zeros(1,L-mod(length(h)-2,L)-1)];% make length 1 + a multiple of L
 N_p = (length(h)-1)/L; % polyphase filter length
 hh = zeros(L,N_p);
 hh(:) = h(1:end-1);    % store polyphase filter in each row
 hh = [hh;h(L+1:L:end)];

 % arbitrary rate resampling
 m = 0; % output index
 y = [];
 xx = [zeros(1,N_p-1),x]; % need to pre-zero-pad input for implementing
 while floor(m/Q)+N_p<length(x),
   p = floor(L*mod(m/Q,1)); % index of current polyphase filter (>= 0)
   n = floor(m/Q)+N_p; % leading index in input data stream
   alf = mod(m*L/Q,1);
   y = [y,((1-alf)*hh(p+1,:)+alf*hh(p+2,:))*xx(n-[0:N_p-1]).'];
   m = m+1;
 end;
 % plot spectra
 figure(4)
 subplot(211)
   plot(2*[0:P-1]/P,abs(fft(x,P)),'r'); title('input x[n]');
 subplot(212)
   plot(2*[0:P-1]/P,abs(fft(y,P))); title('resampled output y[m]');

 % plot time domain
 t_start = 28e-6; t_stop = 34e-6;
 t_delay = (((N_h-1)/2)/L)/1.75e6;
 n_start = floor(t_start*1.75e6);
 n_stop = ceil(t_stop*1.75e6);
 m_start = floor((t_start+t_delay)*1.44e6);
 m_stop = ceil((t_stop+t_delay)*1.44e6);
 figure(5)
 stem([n_start:n_stop]/1.75e6,x([n_start:n_stop]+1),'rx');
 hold on;
 plot([m_start:m_stop]/1.44e6-t_delay,y([m_start:m_stop]+1),'o');
 hold off;
 xlabel('seconds');
 title('original (x) and resampled (o) values (arbtrary rate resampling)');

return
 % plot polyphase filters
 figure(6)
 P = 64;
 subplot(121)
   Hp = fft(hh(1,:),P); plot(2*[0:P/2-1]/P,abs(Hp(1:P/2)));
   hold on;
   for p=2:L,
     Hp = fft(hh(p,:),P); plot(2*[0:P/2-1]/P,abs(Hp(1:P/2)));
   end;
   hold off;
   title('polyphase magnitudes');
 subplot(122)
   grpdly = grpdelay(hh(1,:),1,P); plot(2*[0:P/2-1]/P,grpdly(1:P/2));
   hold on;
   for p=2:L,
     grpdly = grpdelay(hh(p,:),1,P); plot(2*[0:P/2-1]/P,grpdly(1:P/2));
   end;
   hold off;
   title('polyphase group delays');
```