Appendix E: Program for designing the filter considered in transparencies 236 - 239 in the lecture notes

```
**********************************************************************************************
% Matlab m-file (firex1.m)
% This program shows how to design a filter that
% is a cascade of a fixed term H_fix(z) and
% and adjustable term H_adj(z) to meet in the
% minimiax sense the given criteria.
% As an example, we consider the case treated
% in the lecture notes of Section 7 in the FIR filter
% design chapter.
%-----------------------------------------------
% Tapio Saramäki   1.2.96
% Can be found in SUN's: ~ts/matlab/dsp/firex1.m
%-----------------------------------------------
% Fixed filter part
%-----------------------------------------------
for k=1:6
  ome=.4*pi+(k-1)*pi*.05;
  zer(2*k-1)=exp(j*ome);
  zer(2*k)=exp(-j*ome);
end
hfix=poly(zer);
hfix=real(hfix);
%-----------------------------------------------
% Desired and weightingting function for the Remez
% algorithm
%-----------------------------------------------
f=[0 .15 .1501 .3 .4 .6 .601 1];
m=[1 1 1 1 0 0 0 0];
w=[5 1 100 10];
h=remez1(hfix,55,f,m,w);
%-----------------------------------------------
% The basic difference compared to the conventional
% case is that now we use hfix as a first argument
%-----------------------------------------------
% Plot the responses
%-----------------------------------------------
hove=conv(h,hfix);
[H,w]=zeroam(hove,.0,1.,8000);
figure(1)
impz(hove);title('Impulse response for the ovarall filter');
ylabel('Impulse response');xlabel('n in samples');
figure(2)
plot(w/pi,20*log10(H));
title('Amplitude response for the overall filter');
```

```
axis([0 1 -120 10]); grid;
ylabel('Amplitude in dB');
xlabel('Angular frequency omega/pi')
figure(3)
plot(w/pi,(H));
title('Passband details for the overall filter');
axis([0 .3 .99 1.01]); grid;
ylabel('Amplitude');
xlabel('Angular frequency omega/pi')
```

****

BASIC MODIFICATIONS REQUIRED BY remez.m; now remez1.m

***f

```
unction [h,ha] = remez1(hfix,nfilt, ff, aa, wtx, ftype)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
% Diffrence number one: the first parameter is the impulse
% response of the fixed filter part
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
%REMEZ   Parks-McClellan optimal equiripple FIR filter design.
%   B=REMEZ(N,F,M) returns a length N+1 linear phase (real, symmetric
%   coefficients) FIR filter which has the best approximation to the
%   desired frequency response described by F and M in the minimax
%   sense. F is a vector of frequency band edges in pairs, in ascending
%   order between 0 and 1. 1 corresponds to the Nyquist frequency or
half
%   the sampling frequency. M is a real vector the same size as F
%   which specifies the desired magnitude of the frequency response of
the
%   resultant filter B. The desired response is the line connecting the
%   points (F(k),M(k)) and (F(k+1),M(k+1)) for odd k; REMEZ treats the
%   bands between F(k+1) and F(k+2) for odd k as "transition bands" or
%   "don't care" regions. Thus the desired magnitude is piecewise linear
%   with transition bands. The maximum error is minimized.
%
%   B=REMEZ(N,F,M,W) uses the weights in W to weight the error. W has
one
%   entry per band (so it is half the length of F and M) which tells
%   REMEZ how much emphasis to put on minimizing the error in each
band
%   relative to the other bands.
%
%   B=REMEZ(N,F,M,'Hilbert') and B=REMEZ(N,F,M,W,'Hilbert') design
filters
%   that have odd symmetry, that is, B(k) = -B(N+2-k) for k = 1, ..., N+1.
%   A special case is a Hilbert transformer which has an approx.
```

```
magnitude
%       of 1 across the entire band, e.g. B=REMEZ(30,[.1 .9],[1 1],'Hilbert').
%
%       B=REMEZ(N,F,M,'differentiator') and
B=REMEZ(N,F,M,W,'differentiator')
%       also design filters with odd symmetry, but with a special weighting
%       scheme for non-zero magnitude bands. The weight is assumed to be
equal
%       to the inverse of frequency times the weight W. Thus the filter has a
%       much better fit at low frequency than at high frequency. This designs
%       FIR differentiators.
%
%       See also FIRLS, FIR1, FIR2, BUTTER, CHEBY1, CHEBY2, ELLIP,
FREQZ and
%       FILTER.

% Old help:
%
%REMEZ     Parks-McClellan optimal equiripple FIR filter design.
%       B = REMEZ(N,F,M) designs an N'th order FIR digital filter,
%       with the frequency response specified by vectors F and M,
%       and returns the      filter coefficients in length N+1 vector B.
%       Vectors F and M specify the frequency and magnitude
%       breakpoints for the filter such that PLOT(F,M) would show a
%       plot of the desired frequency response. The elements of M must
%       appear in equal-valued pairs.  The frequencies in F must be
%       between 0.0 < F < 1.0, with 1.0 corresponding to half the
%       sample rate. They must be in increasing order, start with 0.0,
%       and end with 1.0.
%       B = REMEZ(N,F,M,W) uses vector W to specify weighting in each
%       of the pass or stop bands in vectors F and M.
%       B = REMEZ(N,F,M,FTYPE) or B = REMEZ(N,F,M,W,FTYPE), where
FTYPE
%       is the string 'Hilbert' or 'differentiator', designs Hilbert
%       transformers or differentiators, respectively.  For the Hilbert
%       case, the lowest frequency should not be 0.
%
%       See also FIR1, FIR2, BUTTER, CHEBY1, CHEBY2, YULEWALK,
FREQZ
%       and FILTER.

%       Author(s): L. Shure, 3-27-87
%                  L. Shure, 6-8-88, revised
%                  T. Krauss, 3-17-93, fixed hilbert bug in m-file version
%       Copyright (c) 1984-94 by The MathWorks, Inc.
%       $Revision: 1.14 $  $Date: 1994/01/25 17:59:43 $
```

```
%       References:
%         [1] "Programs for Digital Signal Processing", IEEE Press
%             John Wiley & Sons, 1979, pg. 5.1-1.
%         [2] "Selected Papers in Digital Signal Processing, II",
%             IEEE Press, 1976, pg. 97.

%       Note: Frequency transitions much faster than 0.1 can cause large
%       amounts of ripple in the response.

lgrid = 16;     % Grid density (should be at least 16)
nfilt = nfilt + 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------
% Difference number two: instead of nargin, we use
% nargin1=nargin-1, since the number of input parameters
% has increased by one
%------------------------------------------------------
nargin1=nargin-1;
%------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
if (nargin1 < 3 | nargin1 > 5)
        error('Incorrect number of input arguments.')
end
if nfilt < 4
        error('Filter order must be 3 or more.')
end
if nargin1 == 4
        if isstr(wtx)
                ftype = wtx;
                wtx = ones(fix((1+max(size(aa)))/2),1);
        else
                ftype = 'f';
        end
end
if nargin1 < 4
        ftype = 'f';
        wtx = ones(fix((1+max(size(aa)))/2),1);
end
if rem(length(ff),2)
        error('Frequencies must be specified in bands.');
end
if any((ff < 0) | (ff > 1))
```

```
        error('Frequencies must lie between 0 and 1.')
end
daa = diff(aa);
%if abs(any(daa(1:2:length(daa)))) > eps
%       error('Bands must be specified with constant magnitudes.')
        %nomex=1;   % can't call mex-file version
%       nomex=0;
%else
%       nomex=0;
%end

clear daa;

if length(ftype)==0, ftype = 'f'; end

if ftype(1)=='m'
        nomex=1;  if length(ftype)==1, ftype = 'f'; else ftype(1)=[]; end
else
        nomex=0;
end
if ftype(1) == 'h' | ftype(1) == 'H'
        jtype = 3;      % Hilbert transformer
elseif ftype(1) == 'd' | ftype(1) == 'D'
        jtype = 2;      % Differentiator
else
        jtype = 1;      % Regular filter
end
if nargin > 3 & (max(size(wtx)) ~= fix((1+max(size(aa)))/2))
        error('There should be one weight per band.')
end
ha = 1;

ff = ff(:)';
aa = aa(:)';
wtx = wtx(:)';
[mf,nf] = size(ff);
[ma,na] = size(aa);
if na ~= nf
        error('Frequency and amplitude vectors must be the same length.')
end

nbands = nf/2;
jb = 2*nbands;
if jb ~= nf
        error('The number of frequency points must be even.')
end
```

```
% The following constraint is not necessary:
% if jtype ~= 3 & (abs(ff(1)) > eps | abs(ff(jb) - 1) > eps)
%       error('The first frequency must be 0 and the last 1.')
% end

% Not necessary: allow filter designer to shoot him/herself in the foot
% if jtype == 3 & ff(1) == 0
%       error('The first frequency for a Hilbert transformer must not be 0.')
% end

% interpolate breakpoints onto large grid
edge = ff;
fx = aa(2:2:jb);

df = diff(ff);
if (any(df < 0))
        error('Frequencies must be non-decreasing.')
end

% Prevent discontinuities in desired function
for k=2:2:jb-2
   if edge(k) == edge(k+1)
      error('Adjacent bands not allowed.')
   end
end

edge = edge/2;
cmptr = computer;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--------------------------------------------------------
% Difference number three: we use use remezfff, instead
% of remezf. This forces the routine ti use matlab code,
% instead of the Mex-file.
%--------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% if (exist('remezf') == 3) & (nfilt < 128) & ~nomex
if (exist('remezfff') == 3) & ~nomex
        % Use MEX-file
        % disp('MEX-file version')
        %h = eval('remezf(nfilt,edge,fx,wtx,jtype)');
        h = eval('remezfff(nfilt,edge,aa,wtx,jtype)');  % for new remezf - TPK
        h = [h; sign(.5-(jtype ~= 1))*h(max(size(h))-rem(nfilt,2):-1:1)].';
        h = h(max(size(h)):-1:1);
```

```
% put in this code since the mex-file sometimes returns nans - in which
% case we want to use the m-code which produces correct answers
        if ~any(isnan(h))
                return
        else
                h = [];
        end
end
% disp('M-file version')
neg = 1 - (jtype == 1);   % neg == 1 ==> antisymmetric imp resp,
                        % neg == 0 ==> symmetric imp resp
nodd = rem(nfilt,2);      % nodd == 1 ==> filter length is odd
                        % nodd == 0 ==> filter length is even
nfcns = fix(nfilt/2);
if nodd == 1 & neg == 0
        nfcns = nfcns + 1;
end
grid(1) = edge(1);
delf = .5/(lgrid*nfcns);
if neg ~= 0 & edge(1) < delf
        grid(1) = delf;
end
j = 1;
l = 1;
while (l+1)/2 <= nbands
        fup = edge(l+1);
        grid = [grid (grid(j)+delf):delf:(fup+delf)];
        jend = max(size(grid));
        grid(jend-1) = fup;
        sel = j:jend-1;
%       if (jtype==2),   % "differentiator"
%               des(sel) = fx(lband)*((grid(sel)-1)*(jtype == 2) + 1);
%       else
%               slope=(aa(l+1)-aa(l))/(edge(l+1)-edge(l));
%               des(sel) = polyval([slope aa(l)-slope*edge(l)],grid(sel));
%       end
        % desired magnitude is line connecting aa(l) to aa(l+1)
        if edge(l+1)~=edge(l)   %
            slope=(aa(l+1)-aa(l))/(edge(l+1)-edge(l));
            des(sel) = polyval([slope aa(l)-slope*edge(l)],grid(sel));
       else   % zero bandwidth band
           des(sel) = (aa(l)+aa(l+1))/2;
       end
         wt(sel) = wtx((l+1)/2)./ ...
              (1 +((jtype == 2) & aa(l+1) >= .0001)*(grid(sel) - 1));
         j = jend;
```

```
        l = l + 2;
        if (l+1)/2 <= nbands
                grid(j) = edge(l);
        end
end
ngrid = j - 1;
if neg == nodd & grid(ngrid) > .5-delf
        ngrid = ngrid - 1;
end

%clf reset, plot(grid(1:ngrid),des(1:ngrid),'o',grid(1:ngrid),wt(1:ngrid),'r*');

if neg <= 0
        if nodd ~= 1
                des = des(1:ngrid)./cos(pi*grid(1:ngrid));
                wt = wt(1:ngrid).*cos(pi*grid(1:ngrid));
        end
elseif nodd ~= 1
        des = des(1:ngrid)./sin(pi*grid(1:ngrid));
        wt = wt(1:ngrid).*sin(pi*grid(1:ngrid));
else
        des = des(1:ngrid)./sin(2*pi*grid(1:ngrid));
        wt = wt(1:ngrid).*sin(2*pi*grid(1:ngrid));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
%------------------------------------------------------------------------
% Difference number four: change the desired and weighting
% functions using zeroamfi to be desribed separately
%------------------------------------------------------------------------
[hfixres,w]=zeroamfi(hfix,2*pi*grid(1:ngrid));
wt=wt.*hfixres;
des=des./hfixres;
%-------------------------------------------------------------------------
*********************************************************************************************************f
unction [A,w]=zeroamfi(h,w)
%evaluates the amplitude response
%of a linear phase FIR filter at points
%given by vector w
%------------------------------------------
%Tapio Saramäki 17.11.1997
%----------------------------------------
N=length(h);
NN=floor((N-1)/2);
iodd=0;
if 2*round(N/2)==N iodd=1; end
```

```
isy=1;
sum1=sum(h(1:NN));sum2=sum(h(N+1-NN:N));
if abs(sum2-sum1)/abs(sum1) < .5 isy=0; end
if iodd==0 & isy==0 itype=1;end
if iodd==1 & isy==0 itype=2;end
if iodd==0 & isy==1 itype=3;end
if iodd==1 & isy==1 itype=4;end
L=floor((N+2)/2);
A=zeros(size(w));
if itype==1 A=A+h(L);end
for k=1:L-1
  if itype==1
  A=A+2*h(L-k)*cos(k*w);end
  if itype==2
  A=A+2*h(L-k)*cos((k-.5)*w);end
  if itype==3
  A=A+2*h(L-k)*sin(k*w);end
  if itype==4
  A=A+2*h(L-k)*sin((k-.5)*w);end
end
A=abs(A);
%--------------------------------------
% all values of A must be greater than
% 10^(-20)
%--------------------------------------
for k=1:length(A)
  if A(k)< 10^(-20); A(k)=10^(-20); end
end
%--------------------------------------
*****************************************************************************************
```