1 Short help on Parks-McClellan design of FIR Low Pass Filters using Matlab

The design of an FIR filter using Parks-McClellan algorithm is a two-step process. First, you need to use the *firpmord* command to estimate the order of the optimal Parks-McClellan FIR filter to meet your design specifications. The syntax of the command is as follows:

```
[n,fo,mo,w]=firpmord(f,m,dev)
```

`f` is the vector of band frequencies. For a low pass filter, `f=[wp ws]` where `wp` is the upper edge of the passband and `ws` is the lower edge of the stopband. The vector `m` contains the desired magnitude response values at the passbands and the stopbands of the filter. Since a lowpass filter consists of a passband followed by a stopband, `m` has two entires. Namely, `m=[1 0]` because you would like the magnitude response to be equal to 1 in the passband and equal to 0 in the stopband. The vector `dev` has the maximum allowable deviations of the magnitude response of the filter from the desired magnitude response. It has the same number of entries as there are `m`. Thus, for a low-pass filter it has two entries and is equal to `[passband ripple, stopband ripple]`. After you specify the vectors `f`, `m` and `dev`, you can run the *firpmord* command in the syntax given above to compute the order of the filter `n`. The *firpmord* command also outputs the resulting `fo` and `mo` vectors. In actually designing your filter using *firpm* , you should use these two vectors instead of `f` and `m`. The second stage is the actual design of the filter, using the *firpm* command. After running *firpmord* and finding the `n`, `fo` and `mo`, type

```
b=firpm(n,fo,mo)
```

to find the impulse response `b` of the Parks-McClellan FIR filter you need to design. The vector `b` contains the coefficients for the $Z-$transform of your filter $H(z)$. That is,

$$H(z) = b(1) + b(2)z^{-1} + b(3)z^{-2} + \cdots + b(n+1)z^{-n}$$

This conludes the design of your filter. The *firpmord* and *firpm* can be used to design also highpass, bandpassand multiband filters. See the Matlab help for details.

2 Matlab Help on `firpmord`

```
 FIRPMORD  Parks-McClellan optimal equiripple FIR order estimator.
    [N,Fo,Ao,W] = FIRPMORD(F,A,DEV,Fs) finds the approximate order N,
    normalized frequency band edges Fo, frequency band magnitudes Ao and
    weights W to be used by the FIRPM function as follows:
        B = FIRPM(N,Fo,Ao,W)
    The resulting filter will approximately meet the specifications given
```

Questions or comments concerning this laboratory should be directed to Prof. Charles A. Bouman, School of Electrical and Computer Engineering, Purdue University, West Lafayette IN 47907; (765) 494-0340; bouman@ecn.purdue.edu

```
by the input parameters F, A, and DEV.  F is a vector of cutoff
frequencies in Hz, in ascending order between 0 and half the sampling
frequency Fs. If you do not specify Fs, it defaults to 2.  A is a
vector specifying the desired function's amplitude on the bands defined
by F. The length of F is twice the length of A, minus 2 (it must
therefore be even).  The first frequency band always starts at zero,
and the last always ends at Fs/2.  It is not necessary to add these
elements to the  F vector.  DEV is a vector of maximum deviations or
ripples (in linear units) allowable for each band.  DEV must have the
same length as A.

C = FIRPMORD(F,A,DEV,FS,'cell') is a cell-array whose elements are the
parameters to FIRPM.

EXAMPLE
   Design a lowpass filter with a passband-edge frequency of 1500Hz, a
   stopband-edge of 2000Hz, passband ripple of 0.01, stopband ripple
   of 0.1, and a sampling frequency of 8000Hz:

      [n,fo,mo,w] = firpmord( [1500 2000], [1 0], [0.01 0.1], 8000 );
      b = firpm(n,fo,mo,w);

   This is equivalent to
      c = firpmord( [1500 2000], [1 0], [0.01 0.1], 8000, 'cell');
      b = firpm(c{:});

CAUTION 1: The order N is often underestimated. If the filter does not
meet the original specifications, a higher order such as N+1 or N+2 will.
CAUTION 2: Results are inaccurate if cutoff frequencies are near zero
frequency or the Nyquist frequency.

See also FIRPM, KAISERORD.
```

3 Matlab Help on `firpm`

```
FIRPM Parks-McClellan optimal equiripple FIR filter design.
   B=FIRPM(N,F,A) 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 A 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. At least one frequency band must have a non-zero
   width. A is a real vector the same size as F which specifies the
   desired amplitude of the frequency response of the resultant filter B.
```

```
The desired response is the line connecting the points (F(k),A(k)) and
(F(k+1),A(k+1)) for odd k; FIRPM 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 amplitude is piecewise linear with transition bands. The
maximum error is minimized.

For filters with a gain other than zero at Fs/2, e.g., highpass
and bandstop filters, N must be even.  Otherwise, N will be
incremented by one. Alternatively, you can use a trailing 'h' flag to
design a type 4 linear phase filter and avoid incrementing N.

B=FIRPM(N,F,A,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 A) which tells
FIRPM how much emphasis to put on minimizing the error in each band
relative to the other bands.

B=FIRPM(N,F,A,'Hilbert') and B=FIRPM(N,F,A,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. amplitude
of 1 across the entire band, e.g. B=FIRPM(30,[.1 .9],[1 1],'Hilbert').

B=FIRPM(N,F,A,'differentiator') and B=FIRPM(N,F,A,W,'differentiator')
also design filters with odd symmetry, but with a special weighting
scheme for non-zero amplitude 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.

B=FIRPM(...,{LGRID}), where {LGRID} is a one-by-one cell array
containing an integer, controls the density of the frequency grid. The
frequency grid size is roughly LGRID*N/2*BW, where BW is the fraction
of the total band interval [0,1] covered by F. LGRID should be no less
than its default of 16. Increasing LGRID often results in filters which
are more exactly equiripple, at the expense of taking longer to
compute.

[B,ERR]=FIRPM(...) returns the maximum ripple height ERR.

[B,ERR,RES]=FIRPM(...) returns a structure RES of optional results
computed by FIRPM, and contains the following fields:

   RES.fgrid: vector containing the frequency grid used in
              the filter design optimization
     RES.des: desired response on fgrid
```

```
      RES.wt: weights on fgrid
       RES.H: actual frequency response on the grid
   RES.error: error at each point on the frequency grid (desired - actual)
   RES.iextr: vector of indices into fgrid of extremal frequencies
   RES.fextr: vector of extremal frequencies

FIRPM is now a "function function", similar to CFIRPM, allowing you
to write a function which defines the desired frequency response.

B=FIRPM(N,F,@fresp,W) returns a length N+1 FIR filter which has the
best approximation to the desired frequency response as returned by the
function handle @fresp.  The function is called from within FIRPM using
the syntax:
                 [DH,DW] = fresp(N,F,GF,W);
where:
N is the filter order.
F is the vector of frequency band edges which must appear monotonically
  between 0 and +1, where 1 is the Nyquist  frequency.  The frequency
  bands span F(k) to F(k+1) for k odd; the intervals  F(k+1) to F(k+2)
  for k odd are "transition bands" or "don't care" regions during
  optimization.
GF is a vector of grid points which have been linearly interpolated
  over each specified frequency band by FIRPM, and determines the
  frequency grid at which the response function will be evaluated.
W is a vector of real, positive weights, one per band, for use
  during optimization.  W is optional; if not specified, it is set
  to unity weighting before being passed to 'fresp'.
DH and DW are the desired complex frequency response and
  optimization weight vectors, respectively, evaluated at each
  frequency in grid GF.

The predefined frequency response function handle for FIRPM is
@firpmfrf, but you can write your own.  See the help for
PRIVATE/FIRPMFRF for more information.

B=FIRPM(N,F,{@fresp,P1,P2,...},W) specifies optional arguments
P1, P2, etc., to be passed to the response function handle @fresp.

B=FIRPM(N,F,A,W) is a synonym for B=FIRPM(N,F,{@firpmfrf,A},W),
where A is a vector of response amplitudes at each band edge in F.

FIRPM normally designs symmetric (even) FIR filters. B=FIRPM(...,'h')
and B=FIRPM(...,'d') design antisymmetric (odd) filters. Each frequency
response function handle @fresp can tell FIRPM to design either an even
or odd filter in the absence of the 'h' or 'd' flags.  This is done
```

```
with:
      SYM = fresp('defaults',{N,F,[],W,P1,P2,...})
FIRPM expects @fresp to return SYM = 'even' or SYM = 'odd'. If @fresp
does not support this call, FIRPM assumes 'even' symmetry.

% Example of a length 31 lowpass filter:
 h=firpm(30,[0 .1 .2 .5]*2,[1 1 0 0]);

% Example of a low-pass differentiator:
 h=firpm(44,[0 .3 .4 1],[0 .2 0 0],'differentiator');

% Example of a type 4 highpass filter:
    h=firpm(25,[0 .4 .5 1],[0 0 1 1],'h');

See also FIRPMORD, CFIRPM, FIRLS, FIR1, FIR2, BUTTER, CHEBY1, CHEBY2,
         ELLIP, FREQZ, FILTER, and, in the Filter Design Toolbox, FIRGR.
```