2.161 Signal Processing: Continuous and Discrete
Fall 2008

Massachusetts Institute of Technology
Department of Mechanical Engineering

2.161 *Signal Processing - Continuous and Discrete*
Fall Term 2008

Lecture 17[1]

Reading:

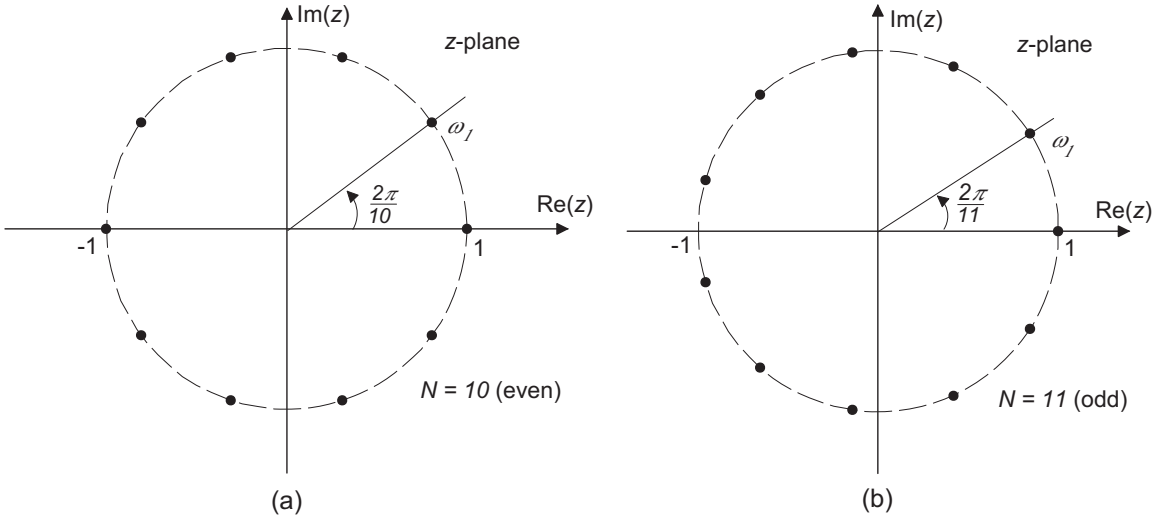
- Class Handout: *Frequency-Sampling Filters.*
- Proakis and Manolakis: Secs. 10.2.3, 10.2.4
- Oppenheim, Schafer and Buck: 7.4
- Cartinhour: Ch. 9

1 Frequency-Sampling Filters

In the frequency-sampling filters the parameters that characterize the the filter are the values of the desired frequency response $H(e^{j\omega})$ at a discrete set of equally spaced sampling frequencies. In particular, let

$$\omega_k = \frac{2\pi}{N}k \qquad k = 0, \ldots, N-1 \tag{1}$$

as shown below for the cases of N even, and N odd. Note that when N is odd there is no sample at the Nyquist frequency, $\omega = \pi$. The frequency-sampling method guarantees that the resulting filter design will meet the given design specification at each of the sample frequencies.



For convenience denote the complete sample set $\{H_k\}$ as

$$H_k = H(e^{j\omega_k}) \qquad k = 1, \ldots, N-1.$$

For a filter with a real impulse response $\{h_n\}$ we require conjugate symmetry, that is

$$H_{N-k} = \bar{H}_k$$

and further, for a filter with a real, even impulse response we require $\{H_k\}$ to be real and even, that is

$$H_{N-k} = H_k.$$

Within these constraints, it is sufficient to specify frequency samples for the upper half of the z-plane, that is for

$$\omega_k = \frac{2\pi}{N}k \qquad \begin{cases} k = 0, \ldots, \frac{N-1}{2} & N \text{ odd} \\ k = 0, \ldots, \frac{N}{2} & N \text{ even.} \end{cases}$$

and use the symmetry constraints to determine the other samples.

If we assume that $H(e^{j\omega})$ may be recovered from the complete sample set $\{H_k\}$ by the cardinal sinc interpolation method, that is

$$H(e^{j\omega}) = \sum_{k=0}^{N-1} H_k \frac{\sin(\omega - 2\pi k/N)}{\omega - 2\pi k/N}$$

then $H(e^{j\omega})$ is completely specified by its sample set, and the impulse response, of length N, may be found directly from the inverse DFT,

$$\{h_n\} = \text{IDFT}\,\{H_k\}$$

where

$$h_n = \frac{1}{N}\sum_{k=0}^{N-1} H_k e^{j\frac{2\pi kn}{N}} \qquad n = 0, \ldots, N-1$$

As mentioned above, this method guarantees that the resulting FIR filter, represented by $\{h_n\}$, will meet the specification $H(e^{j\omega}) = H_k$ at $\omega = \omega_k = 2k\pi/N$. Between the given sampling frequencies the response $H(e^{j\omega})$ will be described by the cardinal interpolation.

1.1 Linear-Phase Frequency-Sampling Filter

The filter described above is finite, with length N, but is non-causal. To create a causal filter with a linear phase characteristic we require an impulse response that is real and symmetric about its mid-point. This can be done by shifting the computed impulse response to the right by $(N-1)/2$ samples to form

$$H'(z) = z^{-(N-1)/2}H(z)$$

but this involves a non-integer shift for even N. Instead, it is more convenient to add the appropriate phase taper to the frequency domain samples H_k before taking the IDFT. The non-integer delay then poses no problems:

- Apply a phase shift of

$$\phi_k = -\frac{\pi k(N-1)}{N} \tag{2}$$

to each of the samples in the upper half z-plane

$$H'_k = H_k e^{j\phi_k} \qquad \begin{cases} k = 0, \ldots, (N-1)/2 & \text{(for } n \text{ odd)} \\ k = 0, \ldots, N/2 & \text{(for } n \text{ even)} \end{cases}$$

- Force the lower half plane samples to be complex conjugates.

$$H'_{N-k} = \bar{H}'_k \qquad \begin{cases} k = 1, \ldots, (N-1)/2 & \text{(for } n \text{ odd)} \\ k = 1, \ldots, N/2 - 1 & \text{(for } n \text{ even)} \end{cases}$$

- Then the linear-phase impulse response is

$$\{h_n\} = \text{IDFT}\,\{H'_k\}$$

1.2 A Simple MATLAB Frequency-Sampling Filter

The following is a MATLAB script of a tutorial frequency-sampling filter

`h = firfs(samples)`

that takes a vector `samples` of length N of the desired frequency response in the range $0 \le \omega \le (N-1)\pi/N$, and returns the linear-phase impulse response $\{h_n\}$ of length $2N-1$.

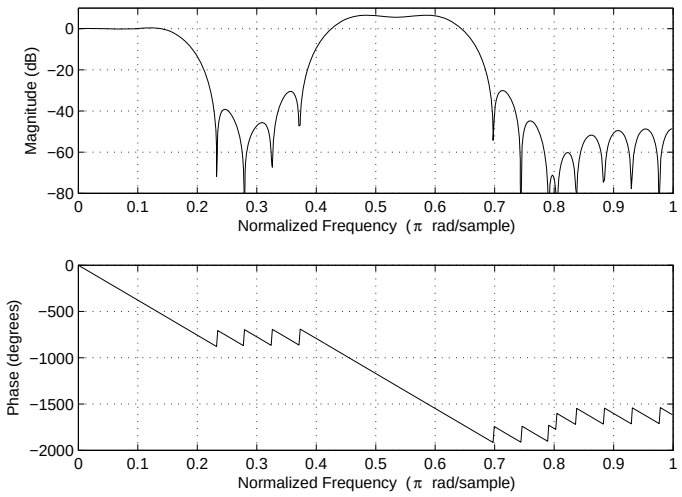
```
------------------------------------------------------------------------
% 2.161 Classroom Example - firfs - A simple Frequency-Sampling Linear-Phase FIR
%                           Filter based on DFT interpolation.
% Usage :    h = firfs(samples)
%            where samples -  is a row vector of M equi-spaced, real values
%                    of the freq. response magnitude.
%                    The samples are interpreted as being equally spaced around
%                    the top half of the unit circle at normalized (in terms of
%                    the Nyquist frequency f_N) frequencies from
%                           0 to 2(M-1)/(2M-1) x f_N,
%                    or at frequencies 2k/(2N-1)xf_N for k = 0...M-1
%                    Note: Because the length is odd, the frequency response
%                    is not specified at f_N.
%                  h - is the output impulse response of length 2M-1 (odd).
%           The filter h is real, and has linear phase, i.e. has symmetric
%           coefficients obeying h(k) = h(2M+1-k), k = 1,2,...,M+1.
%------------------------------------------------------------------------
function h = firfs(samples)
%
% Find the length of the input array...
% The complete sample set on the unit circle will be of length (2N-1)
```

```
%
N = 2*length(samples) -1;
H_d = zeros(1,N);
%
% We want a causal filter, so the resulting impulse response will be shifted
% (N-1)/2 to the right.
% Move the samples into the upper and lower halves of H_d and add the
% linear phase shift term to each sample.
%
Phi = pi*(N-1)/N;
H_d(1) = samples(1);
for j = 2:N/2-1
   Phase        = exp(-i*(j-1)*Phi);
   H_d(j)       = samples(j)*Phase;
   H_d(N+2-j)   = samples(j)*conj(Phase);
end
%
% Use the inverse DFT to define the impulse response.
%
h = real(ifft(H_d));
```

The following MATLAB commands were used to generate a filter with 22 frequency samples, generating a length 43 filter.

```
h=firfs([1 1 1 1 0.4 0 0 0 0 0.8 2 2 2 2 0.8 0 0 0 0 0 0 0 ]);
freqz(h,1)
```

The filter has two pass-bands; a low-pass region with a gain of unity, and a band-pass region with a gain of two. Notice that the band-edges have been specified with transition samples, this is discussed further below. The above commands produced the following frequency response for the filter.



1.3 The Effect of Band-Edge Transition Samples

One of the advantages of the frequency-sampling filter is that the band-edges may be more precisely specified than the window method. For example, low-pass filters might be specified by

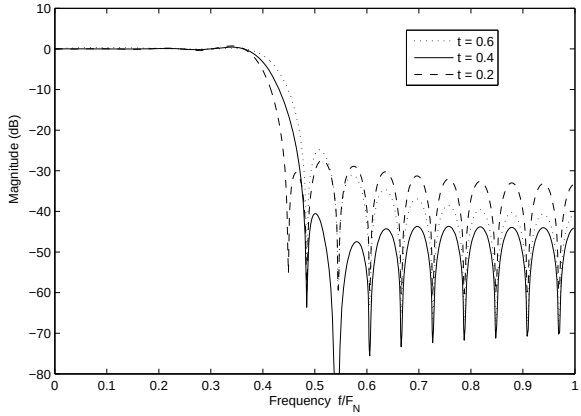
```
h = firfs([1 1 1 1 1 0.4 0 0 0 0 0 0]);
```

with one transition value of 0.4, or

```
h = firfs([1 1 1 1 0.7 0.2 0 0 0 0 0 0]);
```

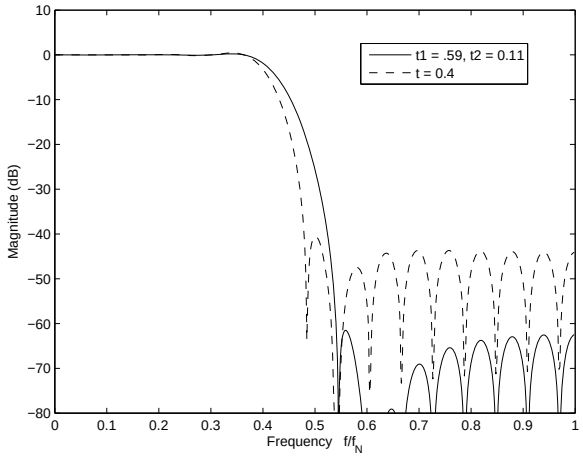
with a pair of transition specifications. The frequency-sampling filter characteristic will pass through these points, and they can have a significant effect on the stop-band characteristics of the filter.

The figure below shows the effect of varying the value of a single transition point in a filter of length $N = 33$.



The values shown are for $t = 0.6, 0.4$ and 0.2. There is clearly a significant improvement in the stop-band attenuation for for the case $t = 0.4$.

Similarly the following figure compares the best of these single transition values ($t = 0.4$) with a the response using two transition points ($t_1 = 0.59$, $t_2 = 0.11$). The filter using two transition points shows a significant improvement in the stop-band over the single point case, at the expense of the transition width.



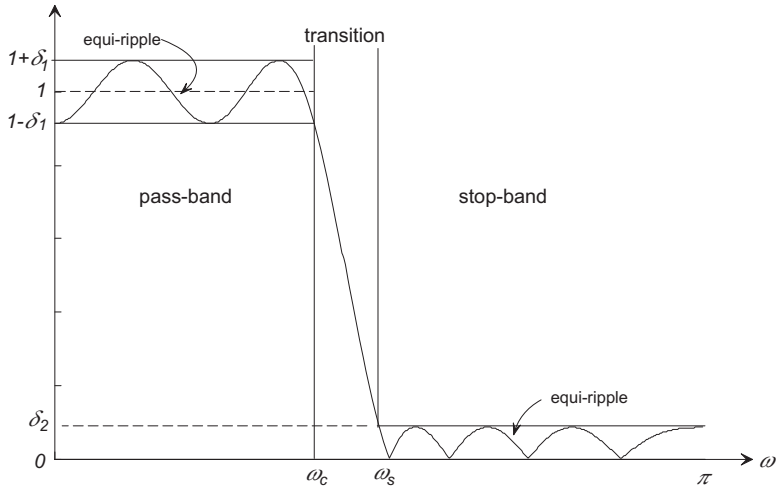
Rabiner et al. (1970) did an extensive linear-programming optimization study to determine the optimum value of band edge transition values, and tabulated the results for even and odd filters of different lengths. The results show that for one transition point $t_{opt} \approx 0.4$, and for two points $t_{opt} \approx 0.59$, and 0.11.

2 FIR Filter Design Using Optimization

These methods allow much greater flexibility in the filter specification. In general they seek the filter coefficients that minimize the error (in some sense) between a desired frequency response $H_d(\mathrm{e}^{\mathrm{j}\omega})$ and the achieved frequency response $H(\mathrm{e}^{\mathrm{j}\omega})$. The most common optimization method is that due to Parks and McClellan (1972) and is widely available in software filter design packages (including MATLAB).

The Parks-McClellan method allows

- Multiple pass- and stop-bands.
- Is an equi-ripple design in the pass- and stop-bands, but allows independent weighting of the ripple in each band.
- Allows specification of the band edges.



For the low-pass filter shown above the specification would be

$$\begin{aligned} 1-\delta_1 \;&<\; H(\mathrm{e}^{\mathrm{j}\omega}) \;<\; 1+\delta_1 \qquad \text{in the pass-band } 0<\omega\le\omega_c \\ -\delta_2 \;&<\; H(\mathrm{e}^{\mathrm{j}\omega}) \;<\; \delta_2 \qquad \text{in the stop-band } \omega_s<\omega\le\pi. \end{aligned}$$

where the ripple amplitudes δ_1 and δ_2 need not be equal. Given these specifications we need to determine, the length of the filter $M+1$ and the filter coefficients $\{h_n\}$ that meet the specifications in some optimal sense.

If $M+1$ is odd, and we assume even symmetry

$$h_{M-k} = h_k \qquad k = 0 \ldots M/2$$

and the frequency response function can be written

$$\begin{aligned} H(\,\mathrm{e}^{\mathrm{j}\,\omega}) &= h_0 + 2\sum_{k=1}^{M/2} h_k \cos(\omega k) \\ &= \sum_{k=0}^{M/2} a_k \cos(\omega k) \end{aligned}$$

Let $H_d(\,\mathrm{e}^{\mathrm{j}\,\omega})$ be the desired frequency response, and define a weighted error

$$E(\,\mathrm{e}^{\mathrm{j}\,\omega}) = W(\,\mathrm{e}^{\mathrm{j}\,\omega})\left(H_d(\,\mathrm{e}^{\mathrm{j}\,\omega}) - H(\,\mathrm{e}^{\mathrm{j}\,\omega})\right)$$

where $W(\,\mathrm{e}^{\mathrm{j}\,\omega})$ is a frequency dependent weighting function, but by convention let $W(\,\mathrm{e}^{\mathrm{j}\,\omega})$ be constant across each of the critical bands, and zero in all transition bands. In particular for the low-pass design

$$W(\,\mathrm{e}^{\mathrm{j}\,\omega}) = \begin{cases} \delta_2/\delta_1 & \text{in the pass-band} \\ 1 & \text{in the stop-band} \\ 0 & \text{in the transition band.} \end{cases}$$

This states that the optimization will control the *ratio* of the pass-band to stop-band ripple, and that the transition will not contribute to the error criterion.

Let Ω be a compact subset of the frequency band from 0 to π representing the pass- and stop-bands. The goal is to find the set of filter parameters $\{a_k\}$, $k = 0, \ldots, M/2+1$ that *minimize the maximum value* of the error $E(\,\mathrm{e}^{\mathrm{j}\,\omega})$ over the pass- and stop-bands.

$$\min_{\text{over } a_k}\left[\max_{\text{over } \Omega}\left[\left|E(\,\mathrm{e}^{\mathrm{j}\,\omega})\right|\right]\right] = \min_{\text{over } a_k}\left[\max_{\text{over } \Omega}\left[\left|W(\,\mathrm{e}^{\mathrm{j}\,\omega})\left(H_d(\,\mathrm{e}^{\mathrm{j}\,\omega}) - \sum_{k=0}^{M/2} a_k \cos(\omega k)\right)\right|\right]\right]$$

where Ω is the disjoint set of frequency bands that make up the pass- and stop-bands of the filter.

The solution is found by an iterative optimization routine. We do not attempt to cover the details of the algorithm here, and merely note:

- The method is based on reformulating the problem as one in polynomial approximation, using Chebyshev polynomials, where

  $$\cos(\omega k) = T_k(\cos(\omega))$$

  where $T_k(x)$ is a polynomial of degree k, (see the Class Handout on Chebyshev filter design). Consequently

  $$H(\,\mathrm{e}^{\mathrm{j}\,\omega}) = \sum_{k=0}^{M/2} a_k \cos(\omega k) = \sum_{k=0}^{M/2} a'_k \left(\cos(\omega)\right)^k$$

- The algorithm uses Chebyshev's *alternation theorem* to recognize the optimal solution. In general terms the theorem is stated:

  Define the error $E(x)$ as above, namely

  $$E(\mathrm{e}^{\mathrm{j}\omega}) = W(\mathrm{e}^{\mathrm{j}\omega})\left(H_d(\mathrm{e}^{\mathrm{j}\omega}) - H(\mathrm{e}^{\mathrm{j}\omega})\right)$$

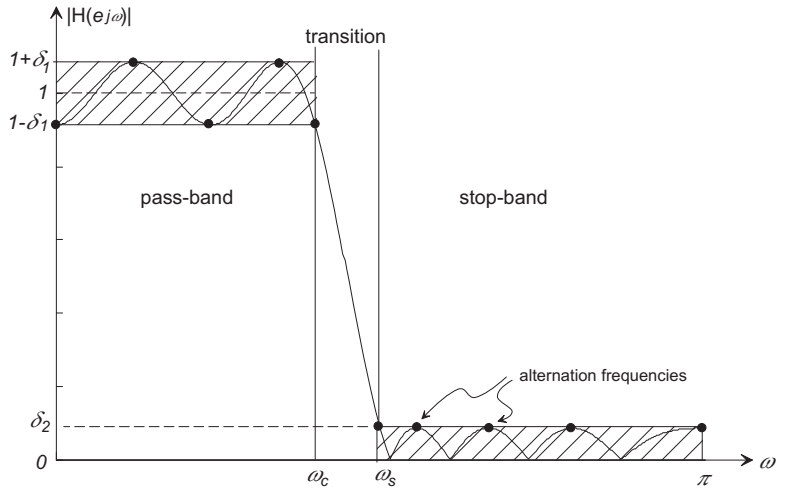
  and the maximum error as

  $$\|E(\mathrm{e}^{\mathrm{j}\omega})\|_\infty = \mathrm{argmax}_{x\in\Omega}\left|E(\mathrm{e}^{\mathrm{j}\omega})\right|$$

  A necessary and sufficient condition that $H(\mathrm{e}^{\mathrm{j}\omega})$ is the unique Lth-order polynomial minimizing $\|E(\mathrm{e}^{\mathrm{j}\omega})\|_\infty$ is that $E(\mathrm{e}^{\mathrm{j}\omega})$ exhibit at least $L+2$ extremal frequencies, or "alternations", that is there must exist at least $L+2$ values of ω, $\omega_k \in \Omega$, $k = [0, 1, \ldots, L+1]$, such that $\omega_0 < \omega_1 < \ldots < \omega_{L+1}$, and such that

  $$E(\mathrm{e}^{\mathrm{j}\omega_k}) = -E(\mathrm{e}^{\mathrm{j}\omega_{k+1}}) = \pm\left(\|E(\mathrm{e}^{\mathrm{j}\omega})\|_\infty\right).$$

  Note that the alternation theorem is simply a way of recognizing the optimal equi-ripple solution. For example, the following figure is from a Parks-McClellan low-pass filter with length $M+1 = 17$.



  From above, $H(\mathrm{e}^{\mathrm{j}\omega})$ is written as a polynomial of degree $M/2$,

  $$H(\mathrm{e}^{\mathrm{j}\omega}) = \sum_{k=0}^{M/2} a'_k \left(\cos(\omega)\right)^k$$

  so that $L = M/2$ and the pass- and stop-bands must exhibit at least $M/2+2 = 10$ points of alternation. These 10 points are shown in the figure.

- The Parks-McClellan algoritm uses the Remez exchange optimization method.

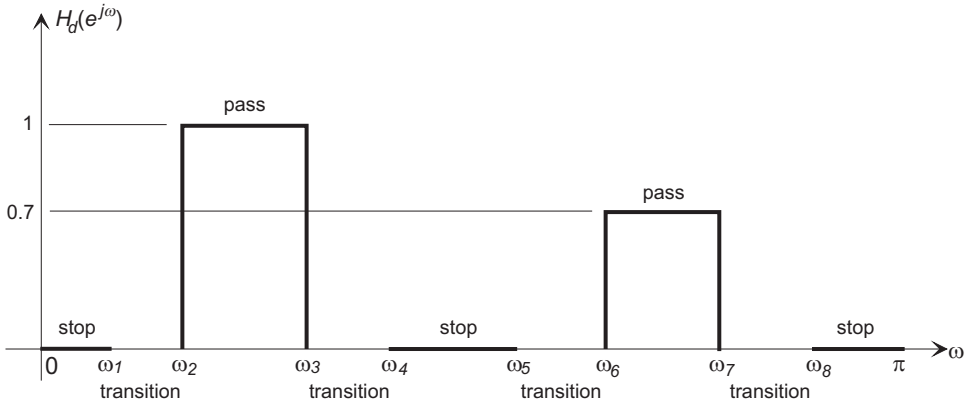
See Proakis and Manolakis Sec. 10.2.4 or Openheim, Schafer and Buck Sec. 7.4 for details.

MATLAB Parks-McClellan Function: The Parks-McClellan algorithm is implemented in the MATLAB function

```
b = firpm(M,F,A,W)
```

where `b` is the array of filter coefficients, `M` is the filter order (`M+1` is the length of the filter), `F` is a vector of band edge frequencies in ascending order, `A` is a set of filter gains at the band edges, and `W` is an optional set of relative weights to be applied to each of the bands.

For example, consider a band-pass filter with two pass-bands as shown below:



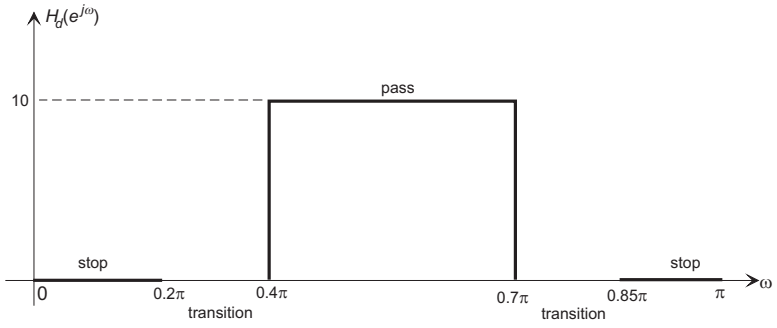
There are five distinct bands in this filter, separated by four transition regions. The filter would require the following specifications:

```
F = [0  ω1  ω2  ω3  ω4  ω5  ω6  ω7  ω8  1]
A = [0  0   1   1   0   0   0.7 0.7 0   0]
W = [10 1   10  1   10]
```

where the errors in the stop-bands have been weighted 10 times more heavily than in the pass-bands. See the MATLAB help/documentation for more details.

■ Example 1

Design a length 33 Parks-McClellan band-pass filter with the following band specifications:



Weight the stop-band ripple ten times more heavily than the pass-band.

Solution:

```
h=firpm(32,[0 0.2 0.4 0.7 0.85 1],[0 0 10 10 0 0],[10 1 10])
freqz(h,1)
```