Indian Institute of Technology Bombay
Dept of Electrical Engineering

Handout 18 EE 210 Signals and Systems
Lecture Notes 5 Oct 30, 2015

1 Practical Filter Design

We will now apply the Fourier representations that we learned to design filters. Designing filters illustrates the versatility of these representations. There are a plenty of other applications, from communications, signal acquisition and reconstruction, number-theory and in a variety of other fields. However, none of these is as relevant to us as the filter design problems that we describe.

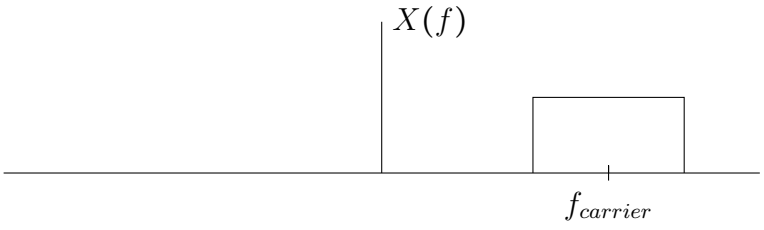
We have already designed several filters in the class. However, we didn't worry about the practicality of such filters, i.e., whether the time domain filter that we found can be implemented in practice?. To shed some light, many time-domain filters that we designed are symmetric, that their response runs into negative time, in the same manner as it is in the right hand side of zero. This poses a practical problem, what is negative time?. We have a precise notion to characterize this, known as 'Causality'.

Causal Filters
In a Causal filter, the response $h(\tau)$ is identically zero for all $\tau < 0$.

Notice that this is same as the notion of *causality* that we considered in the initial sections. There causality required that *no* input signal to **cause** a change at the output before the time it is applied. The design examples of Rectifier, Television Sound etc ignored causality. Does it mean that those designs are not useful?. In fact, we will learn how to make use of those designs without breaking much sweat.

2 Why Design Filters

The question, apart from its philosophical leanings, is important to us, as need spurs creativity. The ubiquitous radio can be used as a motivating example. There are dozens of channels that one can listen to in a radio. The mechanism of tuning is to focus on a particular channel and filter out all the unwanted bands. In this setup, we need a bandpass filter which picks up the required audio-band around the carrier frequency. Further, our listening device should offer a zero gain (open circuit) to all other frequencies outside the band of interest. This is illustrated in the picture below.



The sharp cutoff shown in the thick-lines is not possible in practice. Remember a cutoff in frequency domain will effect a convolution with a sinc$(\cdot)$ function in time domain. Furthermore, as we will see later, a causal operation will introduce considerable delay. To reduce delay, we have to shorten the filter, but this will introduce distortions in the frequency domain. So the art of design is to visit some best parts of both worlds, but not all, since there are conflicting interests.

Many a times we cannot completely specify the filter response over the whole frequency. In literature, the term cutoff frequency, passband and stopband capture the essence of the design.

- **cut-off frequency**: For a low pass filter, the cut-off frequency denotes the frequency at which the filter response falls to $\frac{1}{\sqrt{2}}$ of its value at zero[1]
- **passband**: Passband roughly is the band of frequencies where the signal is passed without much attenuation. In the no amplification case, we will say the attenuation is $1-\delta$, with δ small compared to one.
- **stopband**: Opposite to the passband, in the stopband, frequencies are attenuated down. Ideally we like the filter to present an open circuit for these frequencies, but a low gain can be tolerated.

In any case, filter design is all about mimicking the desired effect over a frequency band. Practical considerations prevents achieving the desired frequency response, the question now becomes how *close* we can get.

3 Analog vs Digital Filters

A few decades back analog filters were the norm, and designers have mastered most of the techniques in developing optimal analog filters. Here *optimal* is used with respect to some practical design criteria. More recently, digital techniques have achieved prominence, primarily due to the demand in consumer electronics and communication segments, where digital technologies are driving the growth. The strategy we undertake is to describe mostly the digital filters and introduce their analog counterparts on a need by need basis. We pursue this approach as we have already described many analog filters. Though our descriptions were at a theoretical level (as impulse response $h(\tau)$), with no insight into how these functions can be realized, we will not go deep into this. A mastery of digital techniques can alternately improve our understanding of analog filters.

Both analog and digital filters can be classified into two groups, Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters.

4 FIR vs IIR Filters

As the name suggests, FIR filters will have a finite delay of response. In the discrete terminology, the FIR filter will have a finite number of taps. Understanding IIR can be a bit more tricky, though the name suggests an infinite number of taps or response time or delay. While a digital IIR filter conceptually has an infinite number of taps, it can be computationally very simple, requiring only a few taps. This statement is not a

[1]There are other definitions also.

contradiction, as recursive computations can perform very long computations in small local steps.

FIR designs are more simpler among the two, and many problems there can be mapped to design of appropriate polynomials. Since we are trying to mimic some desired response, polynomials allow us to approximate the desired response. The higher the polynomial degree, more the freedom we have in bending and stretching it to make it fit for the given response. In this loose sense, it appears that we need very high degree polynomials to approximate functions which jump rapidly, like a low pass or band pass filter. This unfortunately is true, and is a major drawback of FIR filters, as the degree of a polynomial has a direct relation to the filter delay (number of taps).

Even though the delay suffered is high, the FIR filter can still offer some uniformity (fairness) in the delay encountered by different frequencies. In particular, an appropriate design can ensure that all frequencies which are passed through the system will suffer the same delay. This is called *Linear Phase* property. This is very important in applications where the time-alignment of diverse frequency components is critical. We will describe this in detail in the next section. The IIR filter scores over their FIR counterparts when it comes to approximating functions with sharp variations.

By viewing FIR design as polynomial approximations, high degree polynomials have better approximation properties. On the other hand, many infinitely long polynomials have a short rational expression. For example,

$$1 - x + x^2 - x^3 + \cdot\cdot = \frac{1}{1+x} \text{ , whenever } x < 1. \tag{1}$$

Thus rational functions can be used to approximate the desired function, and IIR filter is about designing an appropriate rational function. It is easy to see that with polynomials of small degrees at the numerator as well as denominator, we can make the function fit a desired curve. To design an IIR filter, we need to make a clear map between a given rational function and the corresponding IIR filter, which will be done in the discrete case using the so called $Z-$ transform. While lower delay and better approximation are the irresistible offers of IIR design, there is no linear phase guarantee, meaning that different frequencies suffer varying delays while passing through the system.

FIR	IIR
• Simple Design • Linear Phase • *Long Delay*	• Moderately complex design • Small Delay and close approximation • *No linear phase response*

5 Linear Phase Design

Newcomers to filter design will have some difficulty in understanding the fuss generated on linear phase designs. While we all agree that no signal or part of it should be given better service in terms of the delay incurred, what does it have to do with linear phase.

Imagine a signal which is the sum of two sine waves of different frequencies. The system is free to cater to the requirements by amplifying one frequency more with respect to the other. In fact, the frequency response of the filter is all about that, i.e., how much gain each frequency is going to get. Suppose now that one frequency takes a longer time to come out of the system. With the time-alignment lost, the output in many cases is not what we aimed for, though individual frequencies were controlled in some desired fashion. This mis-alignment is caused by the filter having non-linear phase. Recall that if $X(f)$ is the FT of $x(t)$, then

$$x(t-\tau) \stackrel{FT}{\rightleftharpoons} e^{-j2\pi f\tau}X(f). \tag{2}$$

Consider the case where $X(f)$ is real, which will imply that $x(t)$ is symmetric, i.e., $x(t) = x^*(-t)$. Note that in (2), the phase component is solely supplied by the exponential term in the front. But that exponential term indicates that each frequency is delayed by the same amount τ. This can be seen by noting that the phase component is linear in frequency, corresponding to the constant delay τ at the LHS of (2).

Conversely, a linear phase guarantees that we can get a symmetric time response by shifting the waveform. While the time-shifting makes sense in a finite impulse response case, it is not well-defined in an IIR system. More specifically, a stable IIR system will not be of linear phase, and should not be used in applications which demand linear phase. Let us now consider the design of FIR systems.

6 FIR Systems

We have argued that filters are about specifying a desired frequency response. A desired response, if specified for all frequencies of interest, can be converted to a time domain filter using inverse Fourier Transform. But the computed time-domain filter can be a continuous function. The natural way to proceed is to sample the time response. On the other hand, if we know the sampling rate in advance, we can specify the desired response as DTFT instead of the FT. Inverting the DTFT will lead to the time domain discrete-filter. It appears that the filter design job is done and there is nothing more to do. However, to say that, we are discounting several practical considerations. The inverted DTFT can be of infinite length and non-causal. FIR filter design is about managing these two issues while trying to generate the desired response.

We will consider three different design techniques, the Window Design, Frequency Sampled Design, and Parks-Mclellan's Method. In each of these methods, we will try to approximate the desired response using a finite number of filter delays. It is clear that a finite non-causal design can be converted to a causal one by shifting the time-response to the right. By (2), this will add an additional linear phase component in frequency, which is acceptable in applications which can tolerate a small delay, as the reponse magnitude is unchanged. So we will not enforce causality during the design stages, and will be more concerned about approximating a given desired function with the minimum number of taps.

6.1 Window Design Method

Though the name Window Method is not very revealing at this stage, it will become obvious at the end of this section. Our objective is to design a M tap filter $\bar{h} = h_0, h_1, \cdot\cdot, h_{M-1}$ such that, the *distance* between the desired DTFT $\hat{H}_d(f)$ and the actual DTFT $\hat{H}(f)$ is minimized. But what is meant by distance between continuous functions?. To make this clear, we restate our objective function.

$$\underset{h}{\operatorname{argmin}} \int_{\frac{1}{2}}^{\frac{1}{2}} |\hat{H}_d(f) - \hat{H}(f)|^2 df, \tag{3}$$

where

$$\hat{H}(f) = \sum_{m=0}^{M-1} h[m] \exp(-j2\pi f m). \tag{4}$$

Notice that we are computing the so called L_2 **distance**[2] between two functions residing in C^0, which is analogous to the Euclidean distance in discrete spaces. The integral is over $[-\frac{1}{2}, \frac{1}{2}]$, one period of the DTFT. The M tap causal filter which minimizes the cost function is the answer, which in this case can be explicitly found. To this end, let $h_d[n]$ be the IDTFT of $\hat{H}_d(f)$. There is no constraint on the length or causality of $h_d[n]$. Since,

$$h[n] \overset{DTFT}{\rightleftharpoons} \hat{H}(f) \tag{5}$$

$$h_d[n] \overset{DTFT}{\rightleftharpoons} \hat{H}_d(f), \tag{6}$$

the linearity of FT can be used to write,

$$h_d[n] - h[n] \overset{DTFT}{\rightleftharpoons} \hat{H}_d(f) - \hat{H}(f). \tag{7}$$

Now we can apply Parseval's theorem.

$$\sum_n |h_d[n] - h[n]|^2 = \int_{-\frac{1}{2}}^{\frac{1}{2}} |\hat{H}_d(f) - \hat{H}(f)|^2 df. \tag{8}$$

Notice that

$$\sum_n \left|h_d[n] - h[n]\right|^2 = \sum_{n<0} \left|h_d[n] - h[n]\right|^2 + \sum_{0\le n\le M-1} \left|h_d[n] - h[n]\right|^2 + \sum_{n\ge M} \left|h_d[n] - h[n]\right|^2. \tag{9}$$

Thus for any filter $h[n]$,

$$\sum_n \left|h_d[n] - h[n]\right|^2 \ge \sum_{n<0} \left|h_d[n] - h[n]\right|^2 + \sum_{n\ge M} \left|h_d[n] - h[n]\right|^2, \tag{10}$$

as some non-negative values were removed from the RHS of (9). Let $g[n]$ be a filter such that $g[n] = h_d[n]$ whenever $0 \le n \le M-1$ and zero elsewhere. Using $g[n]$ in place of $h[n]$ in (9),

$$\sum_n \left|h_d[n] - g[n]\right|^2 = \sum_{n<0} \left|h_d[n] - g[n]\right|^2 + \sum_{n\ge M} \left|h_d[n] - g[n]\right|^2, \tag{11}$$

since $g[n] = h_d[n], 0 \le n \le M-1$. Let

$$g[n] \overset{DTFT}{\rightleftharpoons} \hat{G}(f). \tag{12}$$

[2]The L_2 distance $|f|_{l_2}$ of a function $f(\cdot)$ is nothing but $\left(\int |f(x)|^2 dx\right)^{\frac{1}{2}}$.

Combining (10), (11) and the definition of $g[n]$,

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} |\hat{H}_d(f) - \hat{G}(f)|^2 df = \sum_n \left|h_d[n] - g[n]\right|^2 \tag{13}$$

$$= \sum_{n<0} \left|h_d[n]\right|^2 + \sum_{n\geq M} \left|h_d[n]\right|^2 \tag{14}$$

$$\leq \sum_{n<0} \left|h_d[n]\right|^2 + \sum_{n\geq M} \left|h_d[n]\right|^2 + \sum_{0\leq n\leq M-1} \left|h_d(n) - h(n)\right|^2 \tag{15}$$

$$= \int_{-\frac{1}{2}}^{\frac{1}{2}} |\hat{H}_d(f) - \hat{H}(f)|^2 df \tag{16}$$

The last two steps are true for any filter $h[n]$ which is zero whenever $n < 0$ or $n \geq M$. Thus the best causal $M-$tap filter is indeed

$$g[n] = \begin{cases} h_d[n]\ , & 0 \leq n \leq M-1 \\ 0\ , & n < 0 \text{ or } n \geq M \end{cases} \tag{17}$$

The name **window design** method is quite obvious now. We take the time domain filter $h_d[n]$, which is the IDFT of the desired response $\hat{H}_d(f)$, and then multiply by a time domain rectangular window to select the values in $[0, M-1]$. We illustrate this method for an ideal low pass filter.

Example 1. *Construct a low pass filter with $M = 7$, where the desired response is ideal with cutoff frequency $f_c = \frac{3}{10}$ (see Figure 1).*

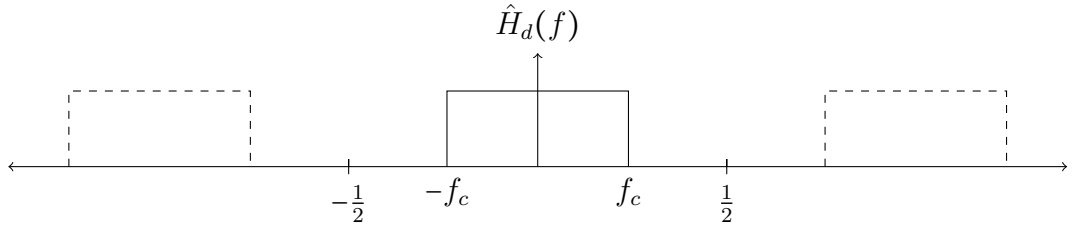


Figure 1: Low pass filter cutoff frequency f_c

We have to design a filter with $M = 7$, i.e., 7 tap filter, such that $|\hat{H}_d(f) - \hat{H}(f)|_{L_2}$ is minimized.
Step I: Find the IDTFT of $\hat{H}_d(f)$.

$$\text{rect}(2f_c) \xrightleftharpoons{IDTFT} 2f_c \,\text{sinc}\,(n2f_c) \tag{18}$$

The ideal filter is shown by thick vertical lines in Figure 2, which is nothing but the samples of a sinc$(\cdot)$ waveform. The window based design just picks up the first 6 samples on the positive (non-negative) axis as shown in the picture.

Notice that this design as such failed to pick up some significant components lying on the negative side of time. The problem is that we insisted $\hat{H}_d(f)$ to be a real filter. Suppose that we demand a linear-phase filter instead of a real filter, i.e. the desired filter is a real response multiplied by a linear phase of the form $\exp(-j2\pi f\tau)$. Correspondingly,

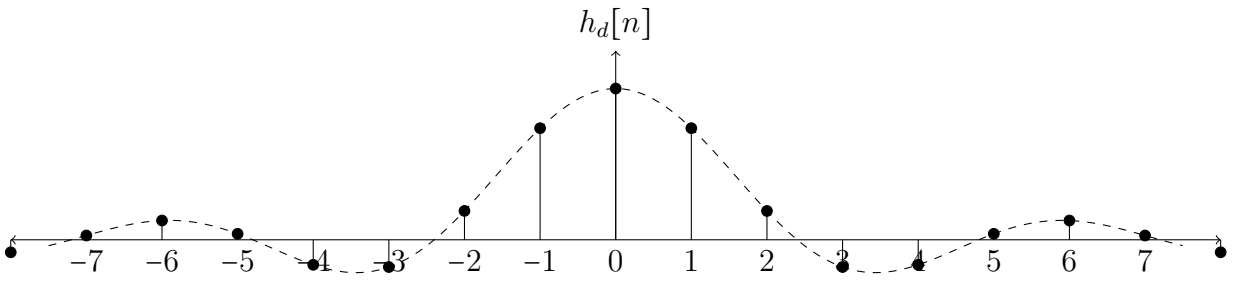


Figure 2: Ideal, but non-causal and infinitely long response

the IDTFT now gets shifted by τ in the time domain. It is easy to see that shifting by $\tau = \frac{M-1}{2}$ will cause the main lobe of the sinc$(\cdot)$ function to appear in the interval $[0, M-1]$, as shown below.

While window design is among the simplest of FIR designs, it suffers from some major draw-backs. Distortion (deviation from the desired response) at individual frequencies can be relatively high, while their sum of squares is still minimal. The next method tries to rectify this problem by ensuring that the deviation at every frequency is kept under control.

6.2 Parks-Mclellan Design

This currently is the most popular FIR design method. The main advantages are a close approximation of the desired response, as well as an iterative design procedure, which can be automated in a computer. The aim is to minimize the deviation from the desired response at every frequency of the DTFT. This amounts to minimizing the maximal deviation. Thus, the optimal filter will find,

$$\min_{h} \max_{f \in [-\frac{1}{2}, \frac{1}{2}]} |\hat{H}_d(f) - \hat{H}(f)|. \tag{19}$$

Comparing to the L_2 minimization design in the last section, here we are trying to minimize the L_∞ norm[3]. This is also known as **MinMax** filter design, for obvious reasons. While the approach is intuitively pleasing, what makes the design ubiquitous is the availability of an iterative algorithm, which can be simplified further in the case of linear-phase designs. We will describe only the linear phase design here. Notice that we are free to design a filter which takes real values in the frequency domain or no-phase filter first, and later shift it in the time domain to get a linear-phase design. The time shifting is necessary as the

[3] L_∞ norm of a function $f(\cdot)$ is $\max_x f(x)$

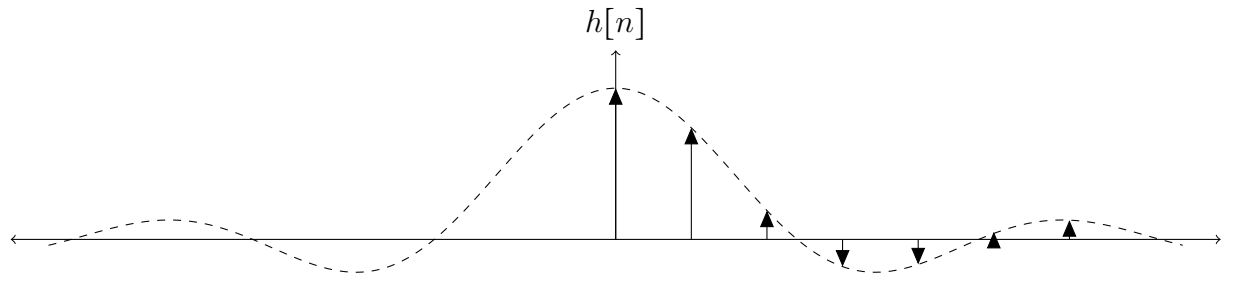


Figure 3: Causal FIR approximation

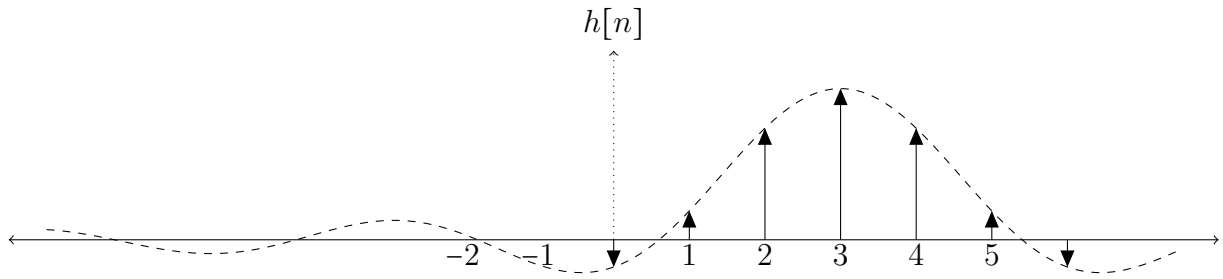


Figure 4: L_2 optimal design with $M = 7$.

real filter cannot be causal as well. Keeping this in mind, we will describe the design of a real, non-causal, symmetric FIR filter of odd length. Since the Fourier Transform of a real filter is symmetric, we will consider only the interval $f \in [0, \frac{1}{2}]$ (positive frequencies) for our design.

For a symmetric filter of odd length, we can write the DTFT as,

$$\hat{H}(f) = \sum_{m=-\frac{M-1}{2}}^{m=\frac{M+1}{2}} h[m] \exp(-j2\pi fm) \tag{20}$$

$$= h[0] + \sum_{m=1}^{\frac{M-1}{2}} (h[m] \exp(-j2\pi fm) + \exp(j2\pi fm)) \tag{21}$$

Define a new filter,

$$g[n] = \begin{cases} 2h[m] \ , \ 1 \leq m \leq \frac{M-1}{2} \\ h[m] \ , \ \text{otherwise.} \end{cases} \tag{22}$$

We can now rewrite (20) as,

$$\hat{H}(f) = \sum_{m=0}^{\frac{M-1}{2}} g[m] \cos(2\pi fm) \tag{23}$$

Using Trigonometric identities,

$$\cos(2\pi fm) = \sum_{l=0}^{m} \beta_{ml} \cos^l(2\pi f). \tag{24}$$

Exercise 1. *Evaluate the values of* β_{ml}.

Substituting in (23),

$$\hat{H}(f) = \sum_{m=0}^{\frac{M-1}{2}} g[m] \sum_{l=0}^{m} \beta_{ml} \cos^l(2\pi f) \tag{25}$$

$$= \sum_{m=0}^{\frac{M-1}{2}} g[m] \sum_{l=0}^{\frac{M-1}{2}} \beta_{ml} 1_{\{l \leq m\}} \cos^l(2\pi f). \tag{26}$$

In the last expression $1_{\{\cdot\}}$ is the indicator function, taking unit value when the expression in subscript is true, and zero otherwise. Now we can rearrange,

$$\hat{H}(f) = \sum_{m=0}^{\frac{M-1}{2}} \sum_{l=0}^{\frac{M-1}{2}} g[m]\beta_{ml} 1_{\{l \le m\}} \cos^l(2\pi f) \tag{27}$$

$$= \sum_{l=0}^{\frac{M-1}{2}} \cos^l(2\pi f) \sum_{m=0}^{\frac{M-1}{2}} g[m]\beta_{ml} 1_{\{l \le m\}} \tag{28}$$

Notice that the second summation is nothing but a linear combination of filter coefficients which can be concisely represented as,

$$\sum_{m=0}^{\frac{M-1}{2}} g[m]\beta_{ml} 1_{\{l \le m\}} = \alpha_l. \tag{29}$$

Thus $\hat{H}(f)$ becomes,

$$\hat{H}(f) = \sum_{l=0}^{\frac{M-1}{2}} \alpha_l \cos^l(2\pi f). \tag{30}$$

Observe that $\hat{H}(f)$ is a polynomial in $\cos(2\pi f)$, i.e. with $x = \cos(2\pi f)$,

$$\hat{H}(f) = \sum_{l=0}^{\frac{M-1}{2}} \alpha_l x^l. \tag{31}$$

Further, since $0 \le f \le \frac{1}{2}$, we will have $-1 \le x \le 1$, and the map $x = \cos(2\pi f)$ is bijective in their respective domains. This implies that our filter design can be completely converted to designing a polynomial of degree $\frac{M-1}{2}$. Approximation by polynomials is a well-researched area[4]. The Russian mathematician Remez has invented an algorithm which will iteratively find the min-max approximation of any given function in the interval $[-1, 1]$. Notice that the approximation can be improved as we increase the degree of the polynomial.

In the filter design context, we first map the desired response $\hat{H}_d(f)$ to a function $g_d(x)$ using the transformation $x = \cos(2\pi f)$. We then design a polynomial $p(x)$ of degree $L = \frac{M-1}{2}$, which is the best approximation to $g_d(x)$ in the max-min sense on the interval $[-1, 1]$. This implies that the maximal deviation from the desired response is minimized by $p(x)$. If this best deviation is not sufficient for our purpose, the filter length L has to be increased. We can make the deviation arbitrary small by increasing L. Once the polynomial is found, we will use Equation (29) to find the actual time domain filter coefficients. Having mapped the filter design problem to that of min-max approximation with polynomials, let us now study the Remez algorithm which performs this task.

6.2.1 Remez Algorithm

Remez algorithm hinges on some fundamental properties of polynomials. Let P_n be the space of all polynomials of degree less than or equal to n. While we consider polynomials with real coefficients, our discussion also applies when the coefficients are taken from the complex domain. P_n is a space under the operations of addition among elements, and multiplication by scalars[5].

[4]The min-max approximation of any function using polynomials is key to an important result in functional analysis called the Weirstrass Theorem

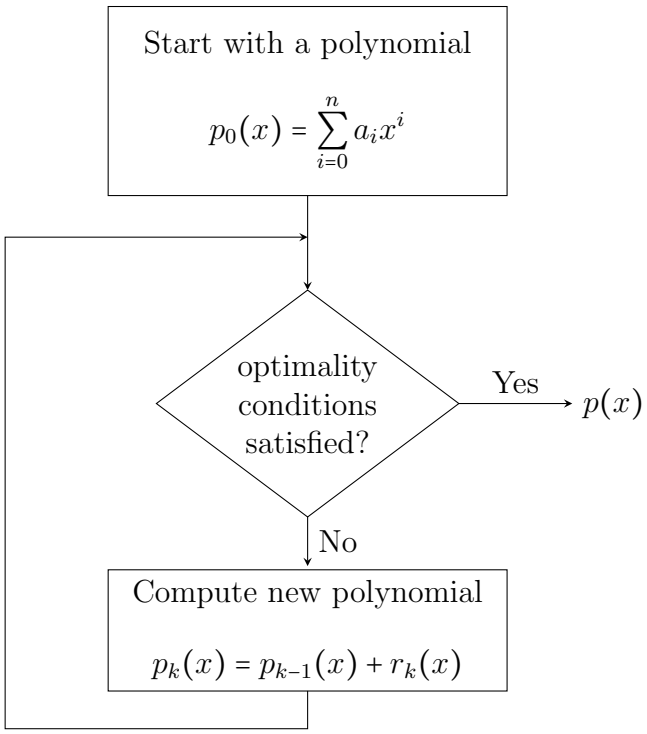
[5]Note that the product of polynomials in P_n may not be in P_n

A polynomial of degree n will have n zeros in $\mathbb{C}$.

Searching though all possible polynomials in P_n to find the best minmax fit is a daunting task, with the *curse of dimensions* stepping in too soon. This is where some side information helps. The use of side-information can be illustrated by the following simple example.

Example 2. *Assume N people have assembled on a ground, and you wish to find the shortest person. Sorting the heights to find this is a resource consuming task. Suppose some side-information is available. For example, a genie tells us that the maximal height difference among the assembled people is $15cms$. Our task is significantly simpler now. It is easy to find the tallest person, and then search for people who are $15cms$ shorter. If it is sufficient to find one among the shorter people in the lot, we will get there in no time.*

In a similar vein to our example, the Remez algorithm has some way of checking whether a given polynomial is optimal or not. If it is not, the polynomial coefficients are adjusted so as to satisfy the optimality requirements. This process is iterated until a reasonable solution is found. This is depicted in figure.



The third step involves finding a new polynomial at every iteration such that the algorithm converges to the optimal solution. This is intimately connected to the optimality test, which we will detail next. The optimality criterion is also known as the **Chebyshev's alternation theorem**.

6.2.2 Chebychev's Alternation Theorem

Our objective in minmax approximation is to find the best polynomial fit $p^*(x)$ of degree $\deg(p^*) \leq L$.

$$p^*(x) = \underset{p:\deg(p)\leq L}{\operatorname{argmin}} \max_{x\in[-1,1]} |g(x) - p(x)|. \tag{32}$$

Theorem 1. *Let p^* be the optimal polynomial of degree at most L. Then there exists $L+2$ points in $[-1,1]$, $-1 \le x_0 \le x_1 \le \cdot\cdot \le x_{L+1} \le 1$ such that,*

$$g(x_i) - p^*(x_i) = (-1)^i \left(g(x_0) - p^*(x_0)\right), \tag{33}$$

and

$$\max_{x\in[-1,1]} |g(x) - p^*(x)| = |g(x_i) - p^*(x_i)| \ , \ 0 \le i \le L+1. \tag{34}$$

Conversely, if there are $L+2$ ordered points such that a polynomial $p^*(x)$ of degree L satisfies the alternation conditions in (33) and (34), then $p^*(x)$ is the minmax approximation of $g(x)$.

The points of alternation are also known as the equi-oscillation points, as the error function $e(x) = g(x) - p^*(x)$ attains its extrema at these points. Thus $e(x)$ oscillates between these extremes. Notice that $e(x)$ may not even be a polynomial. Proving the converse part of the alternation theorem is straightforward. To this end, let $q^*(x)$ be a degree L polynomial which satisfies (33) and (34), with a minmax deviation δ . To get a contradiction, assume that $q^*(x) + r(x)$ is a better minmax approximation than $q^*(x)$. Now, since the deviation everywhere is less than δ with this approximation, on each of the alternation points of $q^*(x)$, we have to make the deviation below δ. This implies that $r(x)$ has to change sign at least $L+1$ times. But this is impossible since $r(x) \in P_L$. Thus $q^*(x)$ indeed is the best approximation. The direct part is a little more involved. More than that, it will be a digression, and will be discussed in an appendix later.

Notice that the key component of the alternation theorem deals with the extrema of the error function. Thus our filter design algorithm needs to keep track of the extremes of $e(x)$ and then check for the alternation condition at each stage of the iteration. Suppose $\bar{x}_k$ are the set of extremes at iteration k. If the alternation condition is not met at these points, we will try to equalize the deviation at each of the extreme points in $\bar{x}_k$. This is done by designing a new polynomial which has equal deviation at each point in $\bar{x}_k$. Let us hope that this polynomial $p_{k+1}(x)$ minimizes the minmax error. However, there is no guarantee that the new error function $f(x) - p_{k+1}(x)$ will have its extrema at $\bar{x}_k$. So we will find the new extremes, say $\bar{x}_{k+1}$ and continue our iteration. Let us consolidate this procedure as an algorithm.

6.2.3 The Algorithm

Step I:

Guess the initial $L+2$ extremes of the error function, say $(x_0 \le x_1 \le \cdot\cdot \le x_{L+1})$.

Step II:

Solve the set of equations

$$g(x_i) - p(x_i) = (-1)^{i+1}\delta \ , \ 0 \le i \le L+1 \tag{35}$$

Notice that there are $L+2$ equations. There are also $L+2$ unknowns, one of them being the parameter $\delta \in \mathbb{R}$. The rest of them are the coefficients of the polynomial $p(x) = \sum_{i=0}^{L} a_i x^i$. It is a fact that these equations are linearly independent for a distinct set of points x_i , $0 \le i \le L+1$. Solving this will give us the polynomial $p(x)$.

Exercise 2. *Show that (35) describes a set of linearly independent equations, when $x_i \ne x_j, \forall i \ne j$.*

Step III:

Check whether the extrema of the error function is still $x_i, 0 \le i \le L+1$. If not replace x_i's with the new $L+2$ extrema and go back to step II.

6.2.4 Computational Efficiency

Notice that the second step of filter design is equivalent to a matrix inversion, which can be quite time consuming. However, the structure of the problem allows us to solve the problem in an efficient manner. In particular, we can solve for δ directly from (35). We will discuss this computation in the next section.

Once δ is known, the problem becomes finding a polynomial which evaluates to a set of desired values at $L+2$ points $x_0, x_1, \cdot, x_{L+1}$. It is easy to find an L^{th} degree polynomial which passes through $L+1$ points, the technique is called Lagrange interpolation. We can take the first $L+1$ points $x_i, 0 \le i \le L$ and find the polynomial which *passes through* $g(x_i) - (-1)^{i+1}\delta$. Since the solution to (35) is unique, we have identified the solution.

In step III, one has to find $L+2$ extrema. Remember that a function changes its direction at the extrema. So, in case there are more candidates, pick the ones which have the highest deviation, as we want to control them. (In class I said to pick the highest deviations. In fact, choosing the highest extrema is the sensible thing to do, as we want alternation of the error function between any two consecutive extrema.)

6.2.5 Filter Design

Remez algorithm gives us an effective way to design optimal minmax polynomials. In fact, a particular variant of Remez algorithm is very suitable for filter design. Here we consider the weighted error function

$$e(x) = w(x)\left(g(x) - p(x)\right), \tag{36}$$

where $w(x)$ is supported in some compact domain of $[-1, 1]$. Let us now find the polynomial which is the weighted minmax approximation of $f(x)$. The Remez algorithm stays the same, except for using the weighted $e(x)$ as the error function. In addition, the alternation theorem should be satisfied inside the region where $w(x)$ is defined, i.e. the extrema should happen in the closed intervals where $w(x)$ is non-zero. During Step II of Remez algorithm, we can evaluate δ in closed form,

$$\delta = \frac{\sum_{i=0}^{L+1} \gamma_i g(x_i)}{\sum_{i=0}^{L+1} \frac{(-1)^{i+1}\gamma_i}{w(x_i)}}, \tag{37}$$

where γ_i is given by

$$\gamma_i = \prod_{j \neq i} \frac{1}{x_j - x_i}. \tag{38}$$

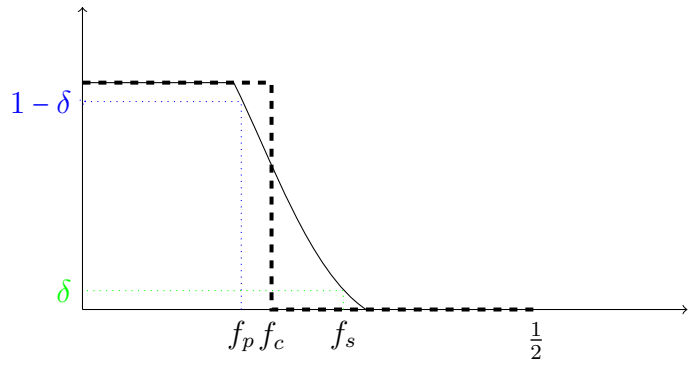
Now, using this δ and Lagrange interpolation, we can find the new polynomial which has equal deviation at the old extrema, and iterate till the extrema are not changing.

Exercise 3. *Show that δ given in* (37) *solves the set of equations,*

$$w(x_i)(f(x_i) - p(x_i)) = (-1)^{i+1}\delta, \ \forall 0 \le i \le L+1, \tag{39}$$

where for each of these points $w(x_i) \neq 0$.

The introduction of $w(x)$ into the error function gives us great flexibility. In fact in many practical filter design contexts, we have a passband, where all frequencies are passed without much attenuation, and a stop band where all the frequencies should be filtered out. Typically there is a frequency gap between the pass and stopbands, and a tolerance factor inside these bands, which is illustrated for a low pass filter in the figure. The unit amplitude rectangle (dashed line) represents an ideal (non-practical) low pass filter.



Notice that we allowed a tolerance level δ which is met at f_p and f_s, where $[0, f_p]$ is the passband and $[f_s, \frac{1}{2}]$ is the stopband.

So we demand a passband characteristic as well as a stopband characteristic and allow the design to be flexible outside these bands. This amounts to placing a $w(x)$ which is unity inside those intervals containing the passbands and stopbands. Thus the weighted minmax design produces a filter which satisfies the alternation theorem within the passband and stopband frequencies.