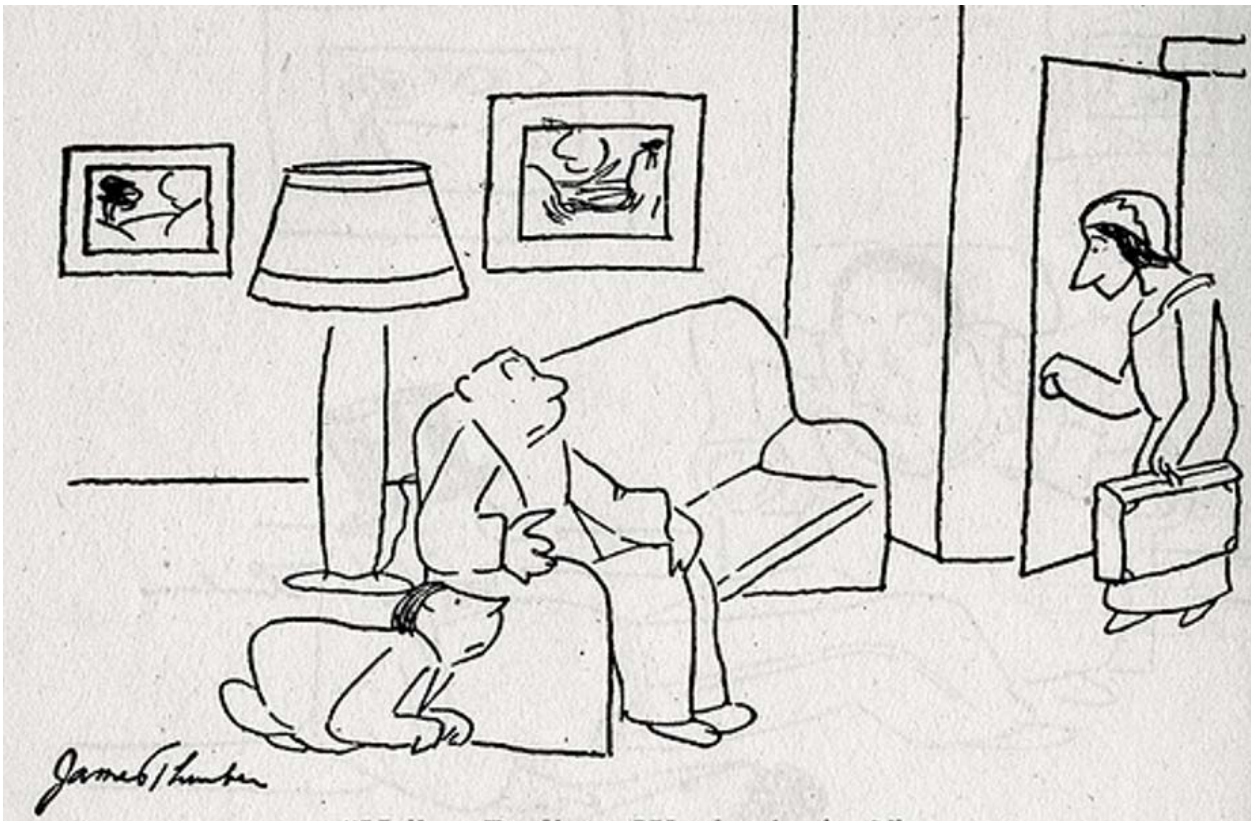

"Hello, Darling—Woolgathering?"

EE 311 **April 1, 2019**
Lecture 19-20

Numerical Direct Design – Pade method

In this method, we begin with an ideal filter with a transfer function $H_D(z)$ and an impulse response of $h_D[n]$. We want to find a transfer function, $H(z)$ with an impulse response $h[n]$ such that

$$\varepsilon = \sum_{n=0}^{\infty} \left(h_D[n] - h[n] \right)^2 \tag{6.35}$$

is minimized for a given filter order. This is a *least squares* filter design [6]. The transfer function we are seeking is of the form

$$H(z) = \frac{\sum_{n=0}^{M} b_n z^{M-n}}{z^N + \sum_{n=1}^{N} a_n z^{N-n}} \tag{6.36}$$

which gives rise to a difference equation given by

$$\begin{aligned} y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_M x[n-M] \\ - a_1 y[n-1] - a_2 y[n-2] - \cdots - a_N y[n-N] \end{aligned}$$

the impulse response is

$$\begin{aligned} h[n] = b_0 \delta[n] + b_1 \delta[n-1] + \cdots + b_M \delta[n-M] \\ - a_1 h[n-1] - a_2 h[n-2] - \cdots - a_N h[n-N] \end{aligned} \tag{6.37}$$

Which can be written as:

$$h[n] = b_n - a_1 h[n-1] - a_2 h[n-2] - \cdots - a_N h[n-N] \qquad for \quad 0 \le n \le M \tag{6.38}$$

and

$$h[n] = -a_1 h[n-1] - a_2 h[n-2] - \cdots - a_N h[n-N] \qquad for \quad n > M \tag{6.39}$$

In (6.38) and (6.39), $h[n]$ is the impulse response of the filter we are trying to design and is unknown. We want to find the values of a_i and b_i such that the error given in (6.35) is minimized. We can make the error zero if

$$h[n] = h_D[n].$$

Making this substitution changes (6.38) and (6.39) to the following

$$h[n] = b_n - a_1 h_D[n-1] - a_2 h_D[n-2] - \cdots - a_N h_D[n-N] \qquad for \quad 0 \le n \le M \tag{6.40}$$

and

$$h[n] = -a_1 h_D[n-1] - a_2 h_D[n-2] - \cdots - a_N h_D[n-N] \qquad for \quad n > M \tag{6.41}$$

In (6.40) and (6.41) there are $N+M+1$ unknowns (the a_i and b_i coefficients).

We can write $N+M+1$ equations by making $h[n]=h_D[n]$ for $0 \le n \le M+N+1$.

The obvious limitation to this method is that it does not make $h[n] = h_D[n]$ for $n > N+M+1$

so that we are effectively truncating the ideal impulse response (a rectangular window). This method is referred to as the Pade approximation method for IIR filter design meeting the least squares criterion.

Example 6.17
Use the Pade approximation method to create a filter which uses the ideal impulse response given by:
$h[n] = \{4, 3, 2, 1, 0.5, 0.5, 0.125, 0.0625, \cdots\}$

Choose $M = N = 2$.

Solution:
We will need $M + N + 1 = 5$ equations. From (6.40) and (6.41) for this example are

$$h[n] = b_n - a_1 h_D[n-1] - a_2 h_D[n-2] \qquad for \quad 0 \le n \le 2$$

and

$$h[n] = -a_1 h_D[n-1] - a_2 h_D[n-2] \qquad for \quad n > 2$$

$$h[0] = 4 = b_0$$
$$h[1] = 3 = b_1 - a_1(4)$$
$$h[2] = 2 = b_2 - a_1(3) - a_2(4)$$
$$h[3] = 1 = b_3 - a_1(2) - a_2(3)$$
$$h[4] = 0.5 = -a_1(1) - a_2(2)$$

In matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -4 & 0 \\ 0 & 0 & 1 & -3 & -4 \\ 0 & 0 & 0 & -2 & -3 \\ 0 & 0 & 0 & -1 & -2 \end{bmatrix} \bullet \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \\ 0.5 \end{bmatrix}$$

The MATLAB® code for evaluation of this set of equations is:

```
A = [1 0 0  0  0;...
     0 1 0 -4  0;...
     0 0 1 -3 -4;...
     0 0 0 -2 -3;...
     0 0 0 -1 -2];

b = [4;3;2;1;0.5];
x = A\b;  % same as A^-1*b;
```

This code produces the following values:
$b_0 =$ 4.0000, $b_1 =$ 1.0000, $b_2 =$ 0.5000
$a_0 = 1$, $a_1 =$ -0.5000, $a_2 =$ 0

which corresponds to a transfer function of

$$H(z) = \frac{4z^2 + z + 0.5}{z^2 - 0.5z}$$

Using MATLAB® the impulse response can be found from the following code:

```
b0 = 4.0000;b1 = 1.0000;b2 = 0.5000;
a0 = 1;a1 = -0.5000;a2 = 0;
num = [b0 b1 b2];
den = [a0 a1 a2];
[h t] = impz(num, den, 10);
```

The corresponding impulse response is

$$h[n] = \{4, 3, 2, 1, 0.5, 0.25, 0.125, 0.0625, \cdots\}$$

Comparing this impulse response to the original impulse response we see that the first five terms correspond exactly since we had 5 equations and were able to determine the exact coefficients to match those terms. After the fifth term the two sequences differ. Increasing the order would help remedy this.

The Pade approximation provides a method of getting an exact match between and ideal filter's first $M+N+1$ impulse response terms and the filter being designed. The method is somewhat limited in practice because it is often necessary to make $M+N+1$ large in order to meet frequency specifications.

It is necessary, in Pade's method to choose the first $M + 1$ values of the impulse response to form the first $M + 1$ equations since these are the only equations in which the values of the numerator coefficients appear (the b_k's).

The last N equations, however, may be chosen by using any of the impulse response terms beyond $M + 1$ and is not limited to those which are contiguous. In Example 6.17, for instance, we might have chosen $h_D[0]$ to $h_D[3]$ to use in the first four equations followed by $h_D[5]$, $h_D[7]$, and $h_D[9]$. In some cases a better filter can be designed by the judicious choice of the impulse response terms that are used.

Pade's method has two principle limitations: First, since the first $N + M + 1$ coefficients of the impulse response are used the remaining coefficients are ignored. Second, it is difficult to estimate the order of the filter needed to meet a given set of specifications.

Numerical Direct Design – Prony's method

One approach that has been taken to address the limitations in Pade's method is to develop methods that rely on statistical properties of the impulse response function rather than on specific impulse response terms [6].

Thus, it is possible to write a set of equations which relate the autocorrelation sequence of the impulse response function to the filter coefficients. One such set of equations is referred to as the Prony's method.

Prony's method uses the same numerator coefficients as Pade's method.

The first M terms of the impulse response of the filter we are designing will match those of the desired filter. Equation (6.40) can be used to calculate the numerator coefficients, if we know the denominator coefficients just as it was done in the Pade' approximation. The denominator coefficients are found such that we minimize the error over the remainder of the impulse response terms.

The transfer function for an IIR filter may be written as:

$$H(z) = \frac{\sum_{n=0}^{M} b_n z^{M-n}}{z^N + \sum_{n=1}^{N} a_n z^{N-n}} = \frac{N(z)}{D(z)}$$

The impulse response function is:

$$h[n] = b_n - \sum_{i=1}^{N} a_i h[n-i] \qquad for\ \ 0 \le n \le M$$

and

$$h[n] = -\sum_{i=1}^{N} a_i h[n-i] \qquad for\ \ n > M$$

The function of the error that we are trying to minimize was given as

$$\varepsilon = \sum_{n=0}^{\infty} \left(h_D[n] - h[n] \right)^2$$

$$E[n] = h_D[n] - h[n]$$

Taking the *z* transform gives

$$E(z) = H_D(z) - H(z)$$

To minimize the error we need to expand ε and take the partial derivatives with respect to the coefficients and set them to zero. This process yields a nonlinear equation that is unwieldy. An analytic solution is difficult to find. To make the problem tractable we redefine the error as follows:

$$E(z) = H_D(z) - H(z) = H_D(z) - N(z) / D(z)$$

And

$$\hat{E}(z) = D(z) \cdot E(z) = D(z) \cdot H_D(z) - N(z) \qquad (6.42)$$

$\hat{E}(z)$ is sometimes referred to as the *equation error*.

Multiplication in *z* corresponds to convolution in time. Rewriting (6.42) in the time domain gives

$$D(z) \Leftrightarrow 1 + \sum_{i=1}^{N} a_i \delta[n-i]$$

$$N(z) \Leftrightarrow \sum_{m=0}^{M} b_m \delta[n-m]$$

$$\hat{E}[n] = \left[1 + \sum_{i=1}^{N} a_i \delta[n-i] \right] * h_D[n] - \sum_{m=0}^{M} b_m \delta[n-m]$$

$$\hat{E}[n] = \left[h_D[n] + \sum_{i=1}^{N} a_i h_D[n-i] \right] - b_n, \qquad 0 \le n \le M$$

For $n > M$ this equation becomes

$$\hat{E}[n] = \left[h_D[n] + \sum_{i=1}^{N} a_i h_D[n-i] \right] \qquad n > M$$

The squared error when *n* is greater than *M* is

$$\varepsilon_{sq} = \sum_{n=M+1}^{\infty} \left| \left[h_D[n] + \sum_{i=1}^{N} a_i h_D[n-i] \right] \right|^2 = \sum_{n=M+1}^{\infty} \hat{E}[n] \hat{E}^*[n]$$

$$\frac{\partial \varepsilon_{sq}}{\partial a_j^*} = \sum_{n=M+1}^{\infty} \frac{\partial (\hat{E}[n] \cdot \hat{E}^*[n])}{\partial a_j^*} = \sum_{n=M+1}^{\infty} \hat{E}[n] \frac{\partial \hat{E}^*[n]}{\partial a_j^*} = 0 \qquad (6.43)$$

In this equation

$$\hat{E}^*[n] = \left[h_D^*[n] + \sum_{j=1}^{N} a_j^* h_D^*[n-j] \right] \qquad n > M$$

and

$$\frac{\partial \hat{E}^*[n]}{\partial a_j^*} = h_D^*[n-j]$$

Putting this value for the partial derivative into (6.43) gives

$$\sum_{n=M+1}^{\infty} \hat{E}[n] \frac{\partial \hat{E}^*[n]}{\partial a_j^*} = \sum_{n=M+1}^{\infty} \left[h_D[n] + \sum_{i=1}^{N} a_i h_D[n-i] \right] \cdot h_D^*[n-j] = 0$$

This equation can be written as

$$\sum_{i=1}^{N} a_i \left[\sum_{n=N+1}^{\infty} h_D[n-i] \cdot h_D^*[n-j] \right] = -\sum_{n=M+1}^{\infty} h_D[n] \cdot h_D^*[n-j] \qquad j = 1, 2, \cdots N \tag{6.44}$$

This formidable set of equations represents N equations with N unknowns (b_1, b_2, ... b_N).

Equation (6.44) can be written in a more concise form by defining a new function that is similar to the autocorrelation function.

For a sequence of length L the autocorrelation function is defined as

$$\Phi[n] = (1/L) \sum_{i=0}^{L-1-n} x[i] \cdot x[i+n]$$

We define a similar function as

$$r_h(i, j) \equiv \sum_{n=M+1}^{\infty} h_D[n-i] \cdot h_D[n-j] \tag{6.45}$$

Applying the definition in (6.45) to (6.44) gives

$$r_h(0, j) = -\sum_{i=1}^{N} a_i r_h(i, j) \qquad j = 1, 2, ...N \tag{6.46}$$

The equations in (6.46) are referred to as the *normal* equations or sometimes as *Prony's normal* equations. These equations can be used to determine value of the coefficients $a_1 \ldots a_N$. The values of the b coefficients, $b_0 \ldots b_M$ are determined in the same manner as in Pade's method.

Example 6.18

Use the Prony normal equations to find the transfer function for a digital filter if the ideal filter is given as the following fourth order elliptic filter.

$$H(z) = \frac{0.1282z^4 + 0.1519z^3 + 0.2458z^2 + 0.1519z + 0.1282}{z^4 - 1.0752z^3 + 1.2430z^2 - 0.5038z + 0.1501}$$

Solution:

The Prony normal equations are implemented in a MATLAB® function which takes the form `[num den] = prony(h, numOrder, denOrder);` In this equation, `h` is the impulse response for the filter we are trying to duplicate, `num` and `den` are the polynomials in z for the transfer function for the new filter, and `numOrder` and `denOrder` are the desired order for the numerator and denominator.

The MATLAB® code below creates the filter and plots its frequency response for order four. The frequency response of the resulting filter is shown in Figure 6.35. This response is identical to that of the original fourth order elliptical filter. Reducing the numerator and denominator order for the Prony filter in the MATLAB® code significantly reduces the quality of the approximation.

```
fs = 11025;
nume = [0.1282 0.1519 0.2458 0.1519 0.1282];
dene = [1 -1.0752 1.2430 -0.5038 0.1501];
[hideal T] = impz(nume, dene, 32, fs);
[num den] = prony(hideal, 4, 4);
```

```
[h T] = impz(num, den, 32, fs);
[H f] = freqz(num, den, 1024, fs);
plot(f, abs(H));
```

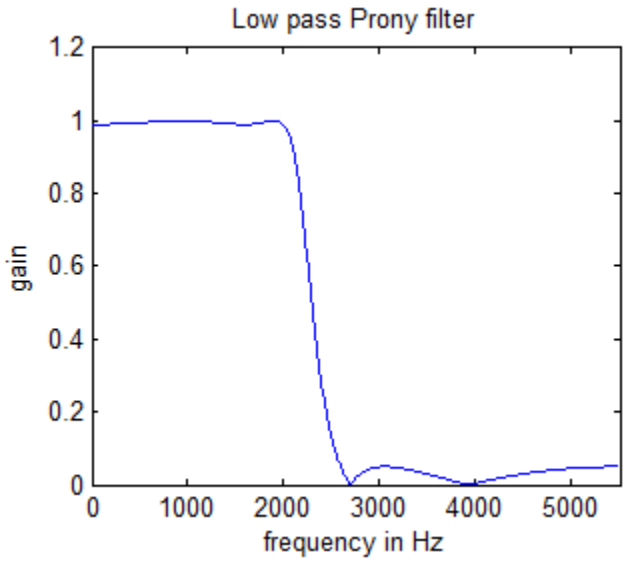

Figure 6.35
A low pass filter designed using the Prony normal equations.

Numerical Direct Design – Yule-Walker method

The Yule-Walker method is similar to Prony's method for the direct design of a digital filter in the time domain. The Yule-Walker equations may be written as

$$r_h(k) + \sum_{i=1}^{N} a_i r_h(k-i) = \begin{cases} \sum_{j=k}^{M} b_j h[j-k], & 0 \le k \le M \\ 0, & k > M \end{cases}$$

where

$r_h(k) = (1/N) \sum_{i=0}^{N-1-k} h[i] \cdot h[i+k]$ is the autocorrelation function [6].

Thus the Yule-Walker equations relate the filter coefficients a_i and b_i to the autocorrelation sequence. The solution to these equations may be achieved by iterative techniques. If we are given the ideal frequency spectrum of a filter we can use the inverse Fourier transform to find its impulse response function. From the impulse response function we can find the autocorrelation sequence and eventually determine the filter coefficients. This is not a trivial process. It has been implemented in MATLAB® under the function name `yulewalk`. Note that the Yule-Walker equations yield a method that is very similar to Prony's method. The difference between the two lies in the definition of $r_h(k)$.

Example 6.19

Use the `yulewalk` function from MATLAB® to design a digital filter from the ideal filter definition given in Figure 6.36.

Solution:

The `yulewalk` function in MATLAB® requires three arguments. The first is the filter order and the second two are vectors which contain the x/y coordinates of the normalized ideal filter. The filter is normalized such that $f_s/2 = 1$. The MATLAB® which creates and plots the filter is given below. The magnitude plot for the filter is in Figure 6.36 right for various orders.

```
fs = 11025;
fpass = 2000;
fstop = 3000;
F = [0 fpass/(fs/2) fstop/(fs/2) 1];
M = [1 1-Rpass Rstop 0];
N = [4 8 12];
[num den] = yulewalk(N(i), F, M);
[H f] = freqz(num, den, 1024, fs);
plot(f, (abs(H)));
```

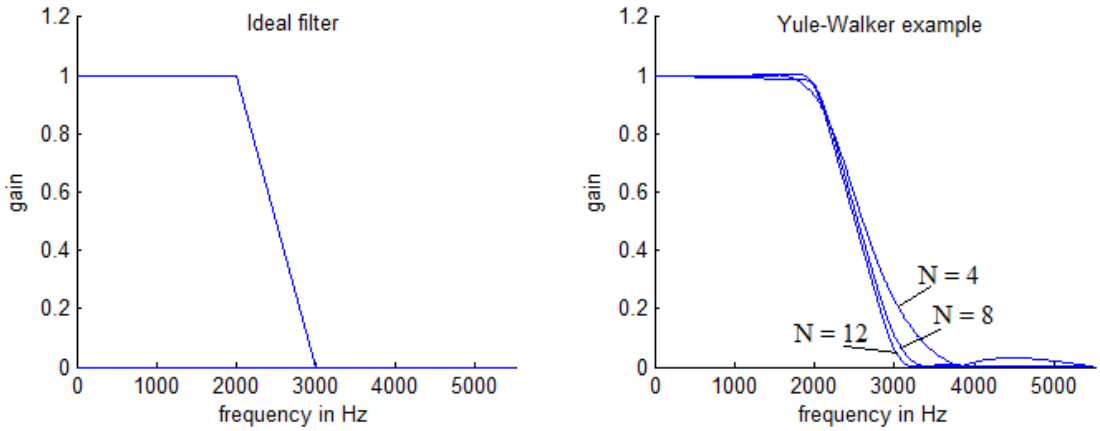

Figure 6.36
Three digital filters designed from the ideal filter (left). The value of N is the order of the Yule-Walker filter.