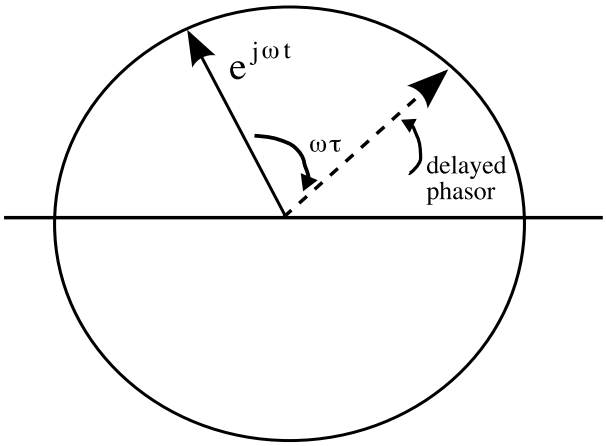

Figure 1: The effect of delaying a phasor by τ sec. The dashed phasor is the delayed version of the original

These notes are based on material from the Digital Signal Processing Primer by Ken Steiglitz.

Feedforward Filters

Filters combine delayed versions of signals and signals are made up of phasors. Therefore, understanding the effect of delaying a phasor is key to everything there is to know about filters.

If we delay the phasor $e^{j\omega t}$ by τ sec, we get

$$e^{j\omega(t-\tau)} = e^{-j\omega\tau}e^{j\omega t} \tag{1}$$

We se from this that a delay of τ sec multiplies the phasor by the complex factor $e^{-j\omega t}$, which does not depend on time t, but only on the amount of delay τ and the frequency ω. This factor rotates the original phasor by the angle $-\omega t$, while leaving its magnitude unchanged. This is illustrated in Figure 1.

A simple filter

We will consider a very simple filter: start with a signal x and add to it some constant a_1 times a delayed version of itself:

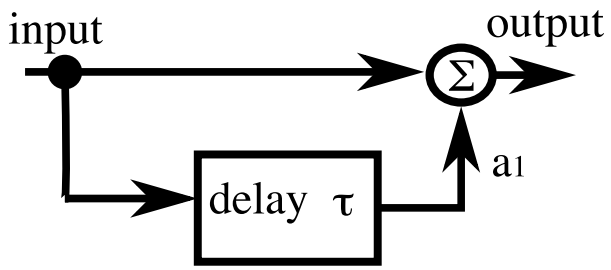

Figure 2: The effect of delaying a phasor by τ sec. The dashed phasor is the delayed version of the original

$$y_t = x_t + a_1 x_{t-\tau} \tag{2}$$

The corresponding signal flowgraph is shown in Figure 2. Filters that use only branches that delay the input signal are called *feeeedforward* filters.

Let's consider the effect of this simple filter when the input x is a phasor at a particular frequency ω, $e^{j\omega t}$. Based on equation 2 we see that the output is the sum of two phasors of the same frequency, which we know is also a phasor of that frequency:

$$y_t = e^{j\omega t} + a_1 e^{j\omega(t-\tau)} \tag{3}$$

We can rewrite this equation by factoring out the phasor:

$$y_t = [1 + a_1 e^{-j\omega\tau}] e^{j\omega t} \tag{4}$$

This equation shows that the output of the filter when the input is a single phasor at frequency ω is also a phasor at the frequency ω. It also shows that the effect of the filter on the input phasor is to multiply it by the complex function in brackets on the right-hand side. Let's call the filter H and denote that complex function by:

$$H(\omega) = 1 + a_1 e^{-j\omega\tau} \tag{5}$$

This function is called the filter's *frequency response*. Notice that $H(\omega)$ depends on the frequency ω and on the fixed parameter τ of the filter, but not on time t. $H(\omega)$ tells everything we need to know about what the filter does to the input phasor.

To see how we can use that information write $H(\omega)$ in polar form as:

$$H(\omega) = |H(\omega)|e^{j\theta(\omega)} \tag{6}$$

The magnitude $|H(\omega)|$ is called the *magnitude response* of the filter, and the angle $\theta(\omega)$ is called the *phase response*. Since the input phasor is multiplied by $H(\omega)$, this tells us that the filter multiplies the size of the input phasor by the filter's magnitude response, and shifts its phase by the filter's phase response.

The magnitude response of our filter is:

$$|H(\omega)| = |1 + a_1 e^{-j\omega\tau}| \tag{7}$$

which is:

$$|H(\omega)| = |1 + a_1^2 + 2a_1 cos(\omega\tau)|^{\frac{1}{2}} \tag{8}$$

Digital Filters

Implementing filters digitally on a computer presents some specific constraints and pecularities that we discuss. When we use a computer to store signals in arrays we are only alloed to delay signals by an integer number of samples because the time variable corresponds to the array index. Therefore on a computer, the supported delays have to be integer multiples of the sampling period T_s.

The other important restriction is that in the digital world, frequencies above half the sampling frequency, don't really exist - we think of them as aliased to frequencies below the Nyquist frequency. This means that the frequency plots of the magnitude responses for digital filters need not extend beyond the Nyquist frequency. It is usually convenient to normalize the frequency variable in such plots to the sampling rate, making the Nyquist frequency equal to 0.5. In this case frequency is expressed as fractions of the sampling rate.

In the digital domain instead of measuring frequency in radians/second we will use radians/sample. We can always go back to "real" time by knowledge of the sampling period T_s. That way the digital sampling frequency is then $\omega = 2\pi$ radians per sample (a full cycle between samples), and the Nyquist frequency is $\omega = \pi$ radians per sample (half a cycle between samples).

Using this new notations we will measure t in integer samples. A very simple example of a digital filter is using a delay of one sample with one feedforward term:

$$y_t = x_t + a_1 x_{t-1} \tag{9}$$

Digital filters can have more than one or two terms. In fact it is possible to design and implement filters with hundreds of terms. How can we possibly know how to selecct a few hundred coefficients so that the resulting digital filter has some desired, predetermined effect ? This question is called the filter design problem. Fortunately, it is almost completely solved for feedforward digital filters. The mathematical problems involved were worked out in the 1960s and 1970s, and design packages are now widely available. The narrower the transition bands and the more exacting the applitude specifications, the more terms we need in the filter to meet the specifications.

Delay as an operator

If the input to the following feedforward filter is the phasor $e^{j\omega t}$,

$$y_t = a_0 x_t + a_1 x_{t-1} \tag{10}$$

the output is also a phasor:

$$y_t = x_t[a_0 + a_1 e^{-j\omega}] \tag{11}$$

In general, with many delay terms, each term in Equation (10) of the form $a_k x_{t-k}$ will result in a term of the form $a_k e^{-kj\omega}$ in Equation (11).

Instead of writing $e^{j\omega}$ over and over, we introduce the symbol:

$$z = e^{j\omega} \tag{12}$$

A *delay* of a phasor by k sampling intervals is the represented simply by multiplication by z^{-k}. Multiplication by z means the phasor is *advanced* one sampling interval, an operation that will be much less common than delay because it is more difficult or impossible to achieve in practical situations.

Notation can have a profound effect on the way we think. Finding the right notation is often the key to making progress in a field. A simple thing like using the symbol z^{-1} for delay is such an example. We are going to treat z^{-1} in two fundamentally different ways: as an operator (in this section) and

as a complex variable (in the next). Both interpretations will be useful in advancing our understanding of digital filters.

An *operator* is a symbol that represents the application of an action on an object. For example we can represent rotating a piece of paper by +90 degrees by the operator p. If we represent the paper by the symbol P, then we write pP to represent the result of applying the operator p to P; that is, pP represents the page actually rotated by +90 degrees. The operator p^{-1} is the inverse operator, in this case rotation by -90 degrees. The operator p^2 applied to a page turns it upside down and p^4 has no net effect, etc.

In the same way, let's use the symbol X to represent a signal with sample values x_t. Note carefully the distinction: X represents the entire signal whereas x_t represents the valu of the signal at a particular time.

The entire signal delayed by one sampling interval is then represented by z^{-1}. Here z^{-1} is an *operator*, which operates on the signal X. We can then rewrite the filter equation:

$$y_t = a_0 x_t + a_1 x_{t-1} \tag{13}$$

as:

$$Y = a_0 X + a_1 z_{-1} X = [a_0 + a_1 z^{-1}] X \tag{14}$$

Notice that in this equation when we write $a_0 X$ it represent the signal obtained by multiplying every value of of X by the constant a_0. It doesn't matter whether we multiply by a constant and then delay a signal, or first delay and then multiply so the order we write these operators is immaterial. In other words, the operator "multiply by a constant" commutes with the delay operator.

The above notation is very suggestive. It tells us to interpret the exper-ession in brackets as a single operator that represents the entire filter. We will therefore rewrite the equation as:

$$Y = H(z) X \tag{15}$$

where

$$H(z) = a_0 + a_1 z^{-1} \tag{16}$$

The operator $H(z)$ will play a central role in helping us think about and manipulate filters; it is called the filter's $tranferfunction$, H.

As a first example of how we can use tranfer functions, consider what happens when we have two simple filters one after the other. This is called

a cascade conection. The first filter produces the output signal W from the input signal X; the second produces the output Y from input W. Suppose the first filter has the transfer function:

$$G(z) = a_0 + a_1 z^{-1} \tag{17}$$

and the second:

$$H(z) = b_0 + b_1 z^{-1} \tag{18}$$

The overall transfer function of the two filters combined can be written:

$$Y = H(z)W = H(z)[G(z)X] \tag{19}$$

The suggestive notation we've derived is now presenting us with a great temptation. Why not multiply the two operators together as if they were polynomials ? If we do we get:

$$H(z)G(z) = (a_0+a_1z^{-1})(b_0+b_1z-1) = a_0b_0+(a_0b_1+a_1b_0)z^{-1}+a_1b_1z^{-2} \tag{20}$$

Can we get away with this ? The answer follows quickly from what we know about ordinary polynomials. We get away with this sort of thing in that case because of the distributive, associative, and commutative laws of albebra hold for combining the operators in transfer functions: delays, additions and multiplies-by-constants. For example, delaying the sum of two signals is completely equivalent to summing after the signals are delayed. Therefore, we are permitted to treat transfer functions the way we treat ordinary polynomials and arrive at correct conclusions.

Multiplying the transfer functions shows that connection of two filters is equivalent to a single three-term filter. This just begins to illustrate how useful transfer functions are. We got to this equivalent form of the cascade filter with hardly any effort at all.

Here's another example of how useful transfer functions are. Multiplication commutes: therefore filtering commutes. That means we get the same result if we filter first by H and then by G, because:

$$G(z)H(z) = H(z)G(z) \tag{21}$$

This is hardly obvious from the filter equations alone. Interpreting z^{-1} as the delay operator gives as a whole new way to represent the effect of a digital filter: as multiplication by a polynomial. We now consider the interpretation of z as a complex variable.

The z-plane

We can gain some useful insights into how filters work by looking at the features of the transfer function in the complex z-plane. Let's go back to a simple digital filter like the one we used earlier:

$$y_t = x_t - a_1 x_{t-1} \tag{22}$$

The effect on a phasor is to multiply it by the compelx function of ω

$$1 - a_1 e^{-j\omega} = 1 - a_1 z^{-1} \tag{23}$$

Remember that we introduced z as shorthand:

$$z = e^{j\omega} \tag{24}$$

If we now now have any transfer function at all, say $H(z)$, the corresponding frequency response is therefore:

$$H(\omega) = H(e^{j\omega}) \tag{25}$$

That is, to get the frequency response, we simply interpret the transfer function as a function of the complex variable z, and evaluate it for values of z on the unit circle. The range of values we are interested in runs from $\omega = 0$ to the Nyquist frequency $\omega = \pi$ radians per sample. This is the top half of the unit circle in the z-plane.

Let's take a closer look at the transfer function in our example. It's just

$$H(z) = 1 - a_1 z^{-1} \tag{26}$$

We can rewrite this as a ratio of polynomials, so we can see where the roots are:

$$H(z) = \frac{z - a_1}{z} \tag{27}$$

There is a zero in the numerator at $z = a_1$, and a zero in the denominator at $z = 0$. That is the transfer function becomes zero at a_1 and infinite at the origin.

The magnitude response is the magnitude of H(z) for z on the unit circle:

$$|H(\omega)| = \frac{|z - a_1|}{|z|} \quad for \;\; z = e^{j\omega} \tag{28}$$

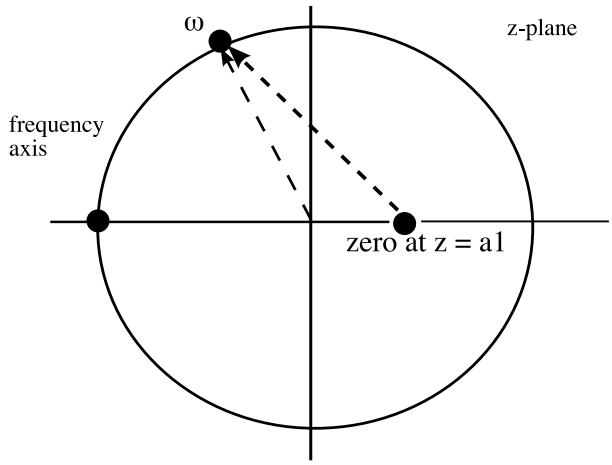

Figure 3: Evaluating the magnitude response of a simple feedforward filter. The factor $|z - a_1|$ is the length of the vector from the zero at a_1 to the point on the unit circle corresponding to the frequency ω.

The demonimator is one, because z is on the unit circle. In fact, for feedforward filters the only possible zeros in the denominator occur at the origin, and these don't affect the magnitude response for the same reason - the magnitude of z on the unit circle is one. We can therefore rewrite the equation as:

$$|H(\omega)| = |z - a_1| \quad for \;\; z = e^{j\omega} \tag{29}$$

Figure 3 shows a geometric interpretation of this expression. It is the length of the vector from the zero at $z = a_1$ to the point on the unit circle representing the frequency ω at which we are evaluating the magnitude response.

This is an elightening interpretation. Picture walking along the unit circle from 0 to the Nyquist frequency. When we are close to the zero, the length of this vector is small, and therefore so is the magnitude response at the frequency corresponding to our position on the unit circle. Conversely, when we are far from the zero, the magnitude response will be large. We can tell directly from Figure 3 that a zero near $z = 1$ will result in a filter that passes high frequencies better than low - a highpass filter. On the other hand a zero at $z = -1$ results in a lowpass filter.

Homework

Show the detailed steps of going from equation (7) to equation (8). Using MATLAB plot this magnitude response when $a_1 = 0.99$ and the delay $\tau =$ 167 milliseconds. Where are the notches of the filter located ?

Find the magnitude response of the digital filter of equation (9). How many notches does that filter have ? Where are they ? Where is the peak of the filter ? Is it high-pass or low-pass ?