DIFFERENCE EQUATIONS

Any function f whose input is restricted to positive or negative integer values n has any output in the form of a sequence of numbers (Stroud & Booth, 2011)

Arithmetic Sequence

$$f(n) = a + nd$$

a=first term, d=common difference

Sum of an arithmetic sequence

$$S_n = \frac{n}{2}[2a + (n-1)d]$$

Geometric Sequence

$$f(n) = Ar^n$$

a=first term, r=common ratio

Sum of a Geometric sequence

$$S_n = \frac{a(1-r^n)}{1-r}$$

Harmonic Sequence

A sequence is said to be harmonic if the reciprocals of its terms form an arithmetic sequence. Accordingly, the sequence defined by the prescription

$$f(n) = \frac{1}{g(n)} \quad g(n) = Arithmetic\ sequence$$

a=first term, r=common ratio

Recursive Prescriptions

$$f(n+1) = f(n) + c$$

e.g.

$f(n) = 5n - 2$ ..... *(1)*

$f(n+1) = 5(n+1) - 2$

$f(n+1) = 5n + 5 - 2$

$f(n+1) = 5n - 2 + 5$ ..... *(2)*

Substitute 5n – 2 in equation (2) into equation (1) we have

$f(n+1) = f(n) + 5$ *where f(1)=3 and n>=1*

Note that the prescription above must have an initial or first term.

Difference Equations

Any equation of the form

$a_{n+m}f(n+m) + a_{n+m-1}f(n+m-1) + a_{n+m-2}f(n+m-2) + .. + a_nf(n) = 0$

where the a's are constants is an mth-order, homogeneous, constant coefficient, linear difference equation also referred to as recurrence relation.

First Order Difference Equations Steps

Any first-order difference equation $af(n+1) + bf(n) = 0$ will have a solution of the form $f(n) = Kw^n$, where the value of n can be found from the characteristic

equation obtained by substituting $f(n) = Kw^n$ into the difference equation. The value of K is found by applying the initial term.

Second Order Difference Equations Steps

Any second-order difference equation $af(n+2) + bf(n+1) + cf(n) = 0$ will have a solution of the form $f(n) = Kw^n$, where the value of n can be found from the characteristic equation obtained by substituting $f(n) = Kw^n$ into the difference equation. Since the characteristic equation is a quadratic there will be two values of w yielding the solution:

$$f(n) = \begin{cases} Aw_1^n + Bw_2^n & if\ w_1 \neq w_2\ and \\ (A+B)w^n & if\ w_1 = w_2 = w \end{cases}$$

The values of A and B are found by applying the initial two given terms

Limits

The limit of a sequence is the number that the output approaches as the input increases to infinity. A sequence with a finite limit is said to be convergent and to converge at that limit. A sequence without a finite limit is said to diverge.

Limits Rules

Given $\lim_{n\to\infty} f(n) = F\ and \lim_{n\to\infty} g(n) = G$

Multiplication by a constant

$$\lim_{n\to\infty} kf(n) = k\lim_{n\to\infty} f(n) = kF\ where\ k\ is\ a\ constant$$

Sums and differences

The limit of a sum (or difference) is the sum (or difference) of the limits:

$$\lim_{n\to\infty}\{f(n) \pm g(n)\} = \lim_{n\to\infty} f(n) \pm \lim_{n\to\infty} g(n)$$

Products and quotients

The limit of products is the product of the limits

$$\lim_{n\to\infty}\{f(n) \times g(n)\} = \lim_{n\to\infty} f(n) \times \lim_{n\to\infty} g(n) = FG$$

and the limit of a quotient is the quotient of the limits

$$\lim_{n\to\infty}\left\{\frac{f(n)}{g(n)}\right\} = \frac{\lim_{n\to\infty} f(n)}{\lim_{n\to\infty} g(n)} = \frac{F}{G}\ \ provided\ G \neq 0$$

If the limits of both the numerator and denominator are infinite the limit is called an indeterminate limit and cannot be found without some manipulation on the quotient.

$$\lim_{n\to\infty} k^n = \begin{cases} 0\ if\ -1 < k < 1 \\ 1\ if\ k = 1 \\ \infty\ if\ k > 1 \\ undefined\ if\ k \leq -1 \end{cases}$$

Difference Equation Examples

Consider the equation:

$f(n+1) + 9f(n) = 0\ where\ f(0) = 6$

To find the form of the general term f(n) that satisfies the this equation we first assume a solution of the form:

$f(n) = Kw^n$

THE Z TRANSFORM DEFINITION AND PROPERTIES

The z-transform is similar inform to the DFT, and for a discrete signal x[n], the unilateral version is given by

$$X(z) = Z\{x[n]\} = \sum_{n=0}^{\infty} x[n]z^{-1} \quad \ldots. \quad (1)$$

Where Z denotes the z-transform. The bilateral transform is similar, but with n ranging from -∞ to +∞. Because we are dealing with casual systems, the unilateral manifestation is sufficient for our purposes. As with Fourier transform and the Laplace transform, the original discrete-space time-domain signal may be recovered from the z-transformed signal, using the inverse z-transform and thus:

$$x[n] = Z^{-1}\{X(z)\} \quad \ldots \quad (2)$$

Where Z^{-1} denotes the inverse z-transform. Although the variable z is more often than not manipulated as simple algebraic symbol, it is important to remember that it is a complex variable, rather like the Laplace operator s. However, an important difference between the Laplace transform and the z-transform is that, whilst the former maps the complex frequency space using Cartesian coordinates (i.e. σ and jω), the latter employs polar coordinates, where any point is defined by its (magnitude) from the origin, and the angle (frequency) that it subtends to it. The relationship between s and z is therefore given by:

$$z = e^{\sigma T} e^{j\omega T} = e^{(\sigma + j\omega)T} = e^{sT} \quad \ldots \quad (3)$$

Hence,

$$z^{-1} = e^{-\sigma T} e^{-j\omega T} = e^{-(\sigma + j\omega)T} = e^{-sT} \quad \ldots \quad (4)$$

Where T reperesents the sample period. Furthermore, the polar coordinates of a point X(r,ϕ) in z-space are given by

$$r = |z| = e^{-\sigma T},$$
$$\Phi =< z = \omega T, \quad \ldots. (5)$$
$$\sigma = -\ln r$$

Using equations (3)-(5), it is possible to map a point in Laplace space into the z-space. Now this latter domain is usually represented by the unit circle, so-called because it has a radius of 1.

Point on unit circle	Frequency (Hz)	Frequency, ω	Angle, ωT, where $T = 0.1$
A	1.25	7.854	$\pi/4$
B	2.5	15.708	$\pi/2$
C	3.75	23.562	$3\pi/4$
D	5	31.416	π
E	6.25 (3.75)	23.562	$-3\pi/4$

Figure 1. Frequency points on the unit circle denoted by angle, where T=0.1

As with the Laplace space, the horizontal axis is real and the vertical axis is imaginary. Any point on the circumference on the unit circle denotes a frequency, the angle of which is given by ωT. For example, assume we are sampling a signal at 10Hz, that is T=0.1s, and we wish to represent frequencies at 1.25, 2.5, 3.75, 5 and 6.25Hz. These are indicated by points A-E in figure below.

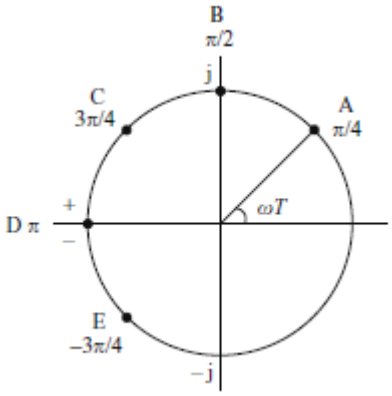


Figure 2 The unit circle of z-space with five frequency points located on its circumference. The frequency of any given point is determined from the angle ωT that it substends to the origin, where T is the sample period.

From our knowledge of sampling, we know that the highest frequency that can faithfully be represented by a signal sampled at f Hz is f/2 Hz. Therefore with the unit circle, the Nyquist point is denoted by π. As a result, although point E represents a frequency of 6.25Hz and therefore an angle of 5π/4, in fact if the system in our example attempted to digitize it, would appear as a frequency of 3.75Hz (i.e. as point C), with an angle of -3π/4.

This brings us to an important distinction between the s-plane and the z-plane. The s-plane maps continuous signals, systems or functions, and therefore not associated with aliasing; in other words, there is no Nyquist point to take into account. In contrast, the unit circle is essentially a periodic mapping technique that describes discrete signals sampled at a finite frequency. Therefore, frequencies that proceed past the Nyquist point simply continue to rotate around the unit circle. This has significant consequences for filter design methodologies that exploit mapping of s-space into equivalent z-space. To convert a coordinate from polar to Cartesian form we use the following formula:

$$re = e^{\sigma T}\cos(\omega T)$$
$$im = e^{\sigma T}\sin(\omega T)$$

If a point moves upward along the vertical axis it represents an increase in frequency. In contrast, the z-plane represents an increase in frequency by the same point moving in counter clockwise direction around the unit circle.

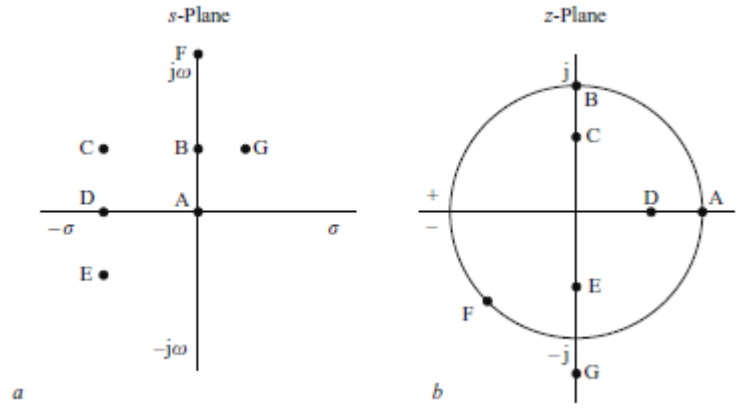


Figure 3 Locations on the s-plane (a) and their corresponding positions on the z-plane (b)

Point	*s*-Plane, Cartesian coordinates		*z*-Plane, polar coordinates		*z*-Plane, Cartesian coordinates	
	σ	ω	$\lvert z\rvert$	$\angle z$	Real	Imaginary
A	0	0	1	0	1	0
B	0	15.707	1	$\pi/2$	0	1
C	−5	15.707	0.606	$\pi/2$	0	0.606
D	−5	0	0.606	0	0.606	0
E	−5	−15.707	0.606	$-\pi/2$	0	−0.606
F	0	39.27	1	$-3\pi/4$	−0.707	−0.707
G	2.5	15.707	1.649	$\pi/2$	0	1.284

Table 1 Points on the s-plane and their equivalent positions in the z-plane, for T=0.1

Referring to Figure 3(a), for example, Point A which is located at the origins of both axes in the s-plane (i.e. zero σ and ω) is located, on the z-plane, on the horizontal (real) axis with a magnitude of 1 (b). Furthermore, a point in the s-plane which is non-zero but purely imaginary, such as point B, will lie at position B on the unit circle. Points C – E are similarly remapped into the z-plane using the appropriate conversion formulae. Now consider point F. Again this is purely imaginary, with a frequency of 6.25Hz, or 39.27 radians. In the s-plane, this simply appears at a position higher up the frequency axis. However, in the discrete case, in the context of this example where T=0.1s, this frequency is aliased downwards. From this we can conclude that although the s-plane always maps points uniquely, the z-plane does not necessarily do so.

Another, and crucial, property of the unit circle is the manner in which it represents instability. We saw in the previous chapter that any system with a pole located on the right-hand side of the vertical axis would be unstable since its output would grow exponentially, theoretically indefinitely (in practice, of course, this cannot happen, since the system would be powered with a voltage of finite magnitude). Point G in (a) shows such a pole. When this pole is mapped to the z-plane, we find that it lies outside the unit circle. Therefore, the right-hand side of the s-space is equivalent to the area outside the unit circle in z-space. Hence, any discrete system that bears a pole with a magnitude greater than 1 will be unstable.

THE Z TRANSFORM AS A POWER SERIES AND DELAY OPERATOR

Imagine we have a causal signal x[n], given by x[n] = 5, -2, 3, 7, -1 for n=0,…,4. Then the z-transform is given by

$$X(z) = \sum_{n=0}^{4} x[n]z^{-n} = x[0]z^0 + x[1]z^{-1} + x[2]z^{-2} + x[3]z^{-3} + x[4]z^{-4}$$

$$X(z) = 5 - 2z^{-1} + 3z^{-2} + 7z^{-3} - z^{-4}$$

Two comments about the expansion of the equation above worth noting as this stage. First, it is possible, at least in theory, to obtain the original discrete time signal from its z-transform merely by extracting the numerical coefficients of each power of z. It must be said, however, that if the inverse transform is required, it is not normally calculated in this manner. As a rule, we would resort to tables of standard inverse transforms and use partial fractions, much as we did using the Laplace transform to solve differential equations. The second observation is that z is essentially a time shift operator. Multiplication by z advances a signal point by one sample interval, and multiplication by z^{-1} delays it by the same amount. Since we have already established that $z=e^{sT}$, where T is the sample interval, we see that multiplication by a complex exponential causes a displacement of the signal in time. This can also be seen in the time-shifting of a Fourier transformed signal being equivalent to the original signal multiplied by a complex exponential.

There is a very simple say to demonstrate the time-shifting property of the z-transform; simply apply it to an impulse function and a delayed impulse function.

Example 1:

Find the z-transform of the following:

(a) The impulse function δ[n];

(b) The weighted, delayed impulse function Aδ[n-3].

Solution

(a) For the isolated impulse function, the z transform is given by:

$$X(z) = \sum_{n=0}^{\infty} x[n]z^{-n} = (1)z^0 + (0)z^{-1} + (0)z^{-2} \ldots$$

Therefore, X(z)=1

(b) For the weighted delayed impulse function, the z-transform is:

$$X(z) = \sum_{n=0}^{\infty} x[n]z^{-n} = (0)z^0 + (0)z^{-1} + (0)z^{-2} + Az^{-3} + (0)z^{-4} \ldots$$

Therefore $X(z)=Az^{-3}$.

Incidentally, part (b) of the solution above also confirms that the z-transform, like the other transforms we have discussed, is a linear operator, therefore

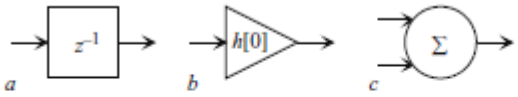
$$\sum_{n=0}^{\infty} kx[n]z^{-1} = k\sum_{n=0}^{\infty} x[n]z^{-1}$$

DIGITAL FILTERS, DIAGRAMS AND Z-TRANSFER FUNCTION

DIGITAL FILTER PROCESSING BLOCKS

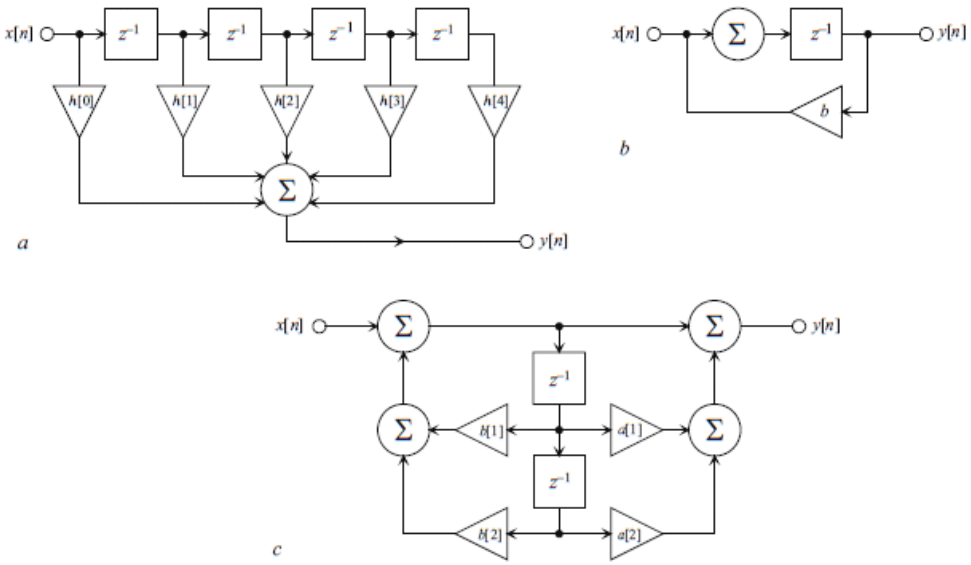
Two main types of linear digital filters operate in the time domain: the FIR and the IIR. No matter how complex such filters become, or how tortuous their impulse responses, there are only ever three operations that a processor uses to execute them: time shift, multiplication and addition. Because these operations are pivotal in the realm of DSP, all real-time DSP devices are optimized to execute them as fast as possible.

A digital filter may be represented as an equation, sometimes called a difference equation, or it may also be depicted as a diagram, the visual nature of which often aids in the understanding of the flow of signals and coefficients, and hence how the filter operates. Filter block diagrams, as they are called, portray three key operations shown by the figure blow. A square enclosing the symbol z^{-1} (or sometimes T) denotes that a delay operation is applied to the incoming signal (a). A triangle enclosing a signal is multiplied by the term or value (b). Finally, the addition or summation is represented by two or more input signals applied to a circle with capital sigma symbol (the + sign is also widely used).



Filter block operations: (a) delay input by one sample interval; (b) multiply input by term; (c) sum inputs

As the name suggests, the finite impulse response filter employs a filter response of finite length, and the output is obtained simply by convolving this with the incoming signal. Consequently, there is no feedback in the system. Such filters, also commonly called transversal filters or feed-forward filters, are typically visualized using the diagram below. This example depicts a five-point FIR structure. In contrast, if the filter is of the IIR type, it must include a feedback. Figure (b) shows a very simple IIR filter that feeds back a fraction b to produce a new output value. This filter is purely recursive, since it does not involve any convolution with multiple input signal values, that is, it has not transversal path. Most IIR filters, however, comprise a recursive and non-recursive part, one such being shown in (c). the right-hand side of the diagram denotes the transversal or feed-forward section, involving coefficients a[1] and a[2], and the left hand side represents the recursive part, with coefficients b[1] and b[2].



(a) FIR, (b) purely recursive and (c) typical IIR filter block diagrams. The final form shown is also a canonic biquad IIR filter stage, or Direct Form II representation (see text)

In general, with this kind of IIR filter, it is considered that the signal is first processed by the recursive section and the processed result is fed back into the transversal, or FIR section. This is also known as a Direct Form II implementation.

DIFFERENCE EQUATIONS AND THE Z-TRANSFER FUNCTION

The convolution equations or difference formula for non-recursive FIR and recursive IIR systems are summarized as follows. The non-recursive FIR discrete-time equation is given by:

$$y[n] = \sum_{k=0}^{M-1} h[[k]x[n-k] \quad \ldots \quad (1)$$

In contrast, the IIR version involves a feedback structure and is therefore given by:

$$y[n] = \sum_{k=0}^{\infty} k[k]x[n-k] = \sum_{k=0}^{M} a[k]x[n-k] - \sum_{k=1}^{N} b[k]y[n-k] \quad \ldots \quad (2)$$

In the case of Equation (1), the impulse response is time limited, that is, it comprises M coefficients, or taps. However, equation (2) reveals that although there is a finite number (M+1) of transversal coefficients and a finite number N of recursive coefficients, because of feedback the impulse response is infinite, or eternal.

As with the Fourier transform and the Laplace transform, the principle of duality operates between the time-domain and the z-domain. Therefore, just as the output signal y[n] can be calculated by convolving the input signal x[n] with the impulse response h[n], so too the z-transform of the output signal is obtained by multiplying the z-transform of x[n] and h[n], that is,

$$Y(z) = H(z)X(z) = Z\{h[n] * x[n]\} \quad \ldots \quad (3)$$

Now the z-transform of h[n] is known as the z-transfer function of the system or sometimes just the transfer function, and is written H(z). it is a most important property and a critical goal, particularly in the design of IIR filters. Rearranging equation (3), we find that in general, the transfer function is given by

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{n=0}^{\infty} h[n]z^{-n} \quad \ldots \quad (4)$$

If we are dealing with an FIR filter, then the transfer function is simply a single polynomial of z^{-1}, that is,

$$H(z) = \sum_{n=0}^{M-1} h[n]z^{-n} = h[0] + h[1]z^{-1} + h[2]z^{-2} + \cdots + h[M-1]z^{-(M-1)} \quad \ldots. \quad (5)$$

In contrast, the transfer function of an IIR filter involves the ratio of two polynomials (Ifeachor andJarvis * 1993) thus:

$$H(z) = \frac{a[0] + a[1]z^{-1} + \cdots + a[m]z^{-m}}{1 + b[1]z^{-1} + \cdots + a[m]z^{-m}} = \frac{\sum_{m=0}^{M} a[m]z^{-m}}{1 + \sum_{n=1}^{N} b[n]z^{-n}} \quad \ldots\ldots \quad (6)$$

Equation (6) has two particular features of interest. The first is that the numerator polynomial represents the non-recursive, or FIR part of the filter, and the denominator polynomial represents the recursive part. This can easily be verified with respect to equation (5); if there is no feedback, then the denominator becomes and we are left with a single polynomial. The second feature of interest is the change in sign (+) in the denominator, respecting the y[n] coefficients, b[n]; in Equation (2), the y[n] terms are shown as negative. This may readily be understood using the following reasoning. Say we have a recursive system, given by

$$y[n] = a[0]x[n] + a[1]x[n-1] + a[2]x[n-2] \quad \ldots \quad (7)$$
$$-b[1]y[n-1] - b[2]y[n-2] - b[3]y[n-3]$$

Rearranging to group y[n] and x[n] yields

$$y[n] + b[1]y[n-1] + b[2]y[n-2] + b[3]y[n-3] \quad \ldots (8)$$
$$= a[0]x[n] + a[1]x[n-1] + a[2]x[n-2]$$

Taking the z-transform of both sides (and using the law of linearity), we obtain

$$Y(z) + b[1]Y(z)z^{-1} + b[2]Y(z)z^{-2} + b[3]Y(z)z^{-3} \quad \ldots\ldots \quad (9)$$
$$= a[0]X(z) + a[1]X(z)z^{-1} + a[2]X(z)z^{-2},$$

That is

$$Y(z)(1 + b[1]z^{-1} + b[2]z^{-2} + b[3]z^{-3})$$
$$= X(x)(a[0] + a[1]z^{-1} + a[2]z^{-2}) \quad \ldots \quad (10)$$

So finally,

$$H(z) = \frac{Y(z)}{X(x)} = \frac{a[0] + a[1]z^{-1} + a[2]z^{-2}}{1 + b[1]z^{-1} + b[2]z^{-2} + b[3]z^{-3}} = \frac{\sum_{m=0}^{3} a[m]z^{-m}}{1 + \sum_{n=1}^{3} b[n]z^{-n}} \quad \ldots \quad (11)$$

It may seem like a trivial point, but the change in sign of the denominator polynomial is something that must not be overlooked when moving from the transfer function to expressing the filter in terms of its difference equation.

POLES AND ZEROS OF DIGITAL FILTERS

As with the Laplacian transfer function, the z-transfer function has poles and zeros, which play an important part in filter design. Again, the zeros are the values of z for which the transfer function becomes zero, and are in essence the roots of the numerator polynomial. The number of zeros therefore corresponds to its order. Similarly, the poles represent the roots of the denominator polynomial, that is, they are the values of z for which the transfer function becomes infinite (Ackroyd, 1973). As long as the poles of a filter lie within the unit circle, it will be stable. In order to avoid confusion with the z operator, we will use the letter α to denot a zero and β to denote a pole.

We have already stated that FIR filters are unconditionally stable; thinking about them in terms of poles and zeros allows us to see why. For example, say we have a filter given by h[n]=0.5, 1, 0.5, for n = 0, … , 2. This will have a transfer function of

$$H(z) = 0.5 + z^{-1} + 0.5z^{-2} \quad \ldots. \quad (12)$$

Clearly, this has two zeros, that is α=-1 twice, but only a single pole given by β=0. The pole-zero diagram for this filter is shown in Figure 4 below, together with its frequency response.

In fact no matter how many terms an FIR filter has, it only ever has a single pole, always located at the origin, that is β=0. This provides us with an alternative understanding as to the reason why the FIR cannot become unstable.

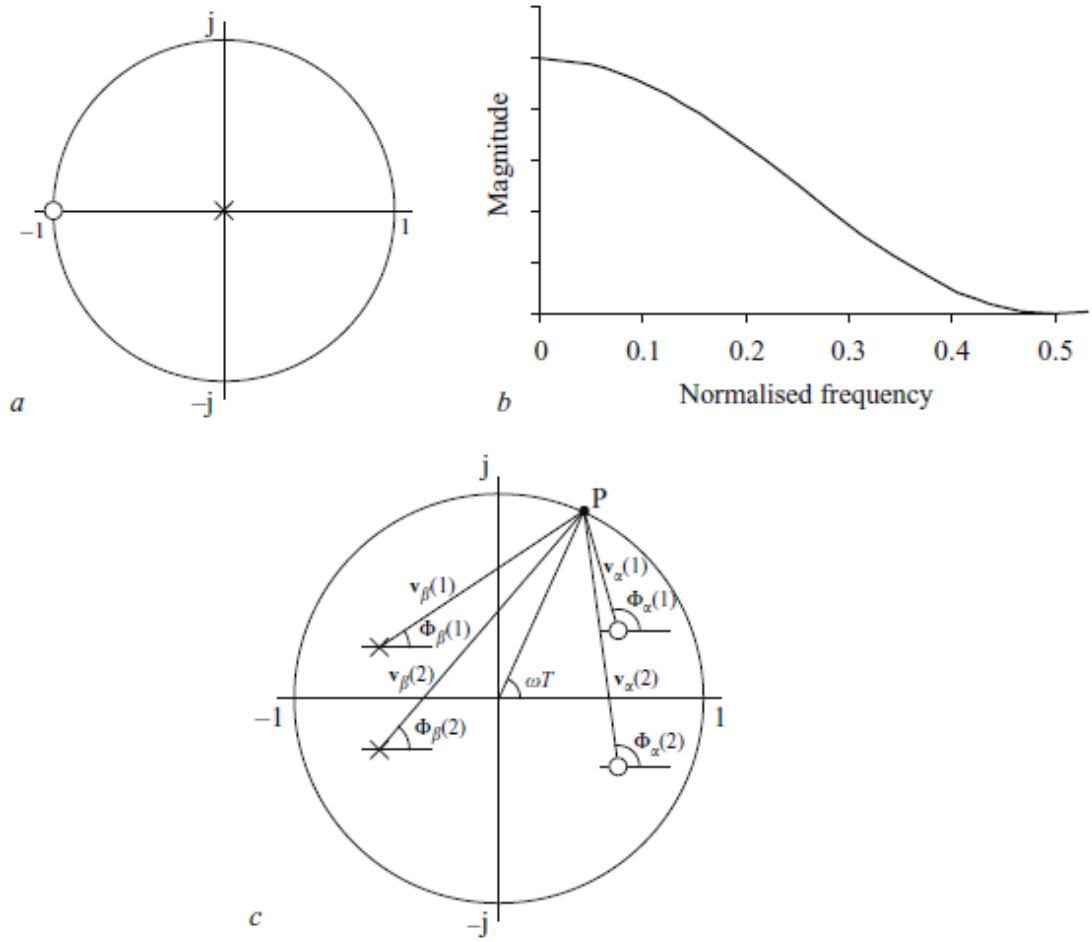


Figure 4 (a) Pole-zero plot and (b) frequency response of a Hanning filter with an impulse response given by 0.5, 1.0, 0.5 (c) Obtaining the magnitude and phase of the frequency response using the pole-zero plot.

IIR filters can have poles anywhere within the unit circle; as we approach a pole magnitude of the filter's output increases, and as we approach a zero, it falls. Using all the poles and zeros within the unit circle it is possible to obtain the frequency response (in terms of both magnitude and phase) of the transfer function(Lynn & Fuerst, 1998). To calculate the response at a given frequency say ω_1, we proceed as follows. First, locate the point P on the circumference of the unit circle that corresponds to this frequency. Now draw vectors from each of the poles to this point. These vectors we will call $v_\beta(1),\ldots, v_\beta(M)$. Next, produce similar vectors for zeros, $v_\alpha(1),\ldots, v_\alpha(N)$. The magnitude of the frequency response at the frequency ω_1 is given by the product of the lengths of all the zero vector divided by the product of the lengths of all the pole vectors, tht is,

$$|H(z)| = \frac{\prod_{n=1}^{N} V_\alpha(n)}{\prod_{m=1}^{M} V_\beta(m)} \quad \ldots \quad (13)$$

This is illustrated in Figure 4(c) above. As with the pole-zero diagram of the Laplace transform, to get the right answers using this method, we have to use radians, not hertz, when calculating the lengths of the pole and zero vectors. To obtain the phase of the frequency response at the frequency ω_1, we sum the phases of all the zero vectors, sum the phases of the pole vectors and subtract the second sum from the first, that is,

$$\Phi(z) = \sum_{n=1}^{N} \Phi_\alpha(n) - \sum_{m=1}^{M} \Phi_\beta(m) \quad \ldots. \quad (14)$$

This is also illustrated in Figure 4(c). an alternative method of obtaining the frequency response is to replace z^{-1} with the term $e^{-j\omega T}$ wherever it appears in the transfer function; a brief inspection of equation (1) shows that if we do this then we are simply taking the Fourier transform of the impulse response. Although both the pole-zero method and the substitution method is simply to stimulate the difference equation made with an impulse, and compute the DFT of the resulting function (which is of course the impulse response). Philosophically, one could argue that IIR filters preclude the use of this method, since their output is eternal. However, in reality even the highest-order designs degenerate outputs that fall to negligible levels after a few hundred or so points, giving rise to insignificant errors in the transform.

Figure 5 below for example, shows a recursive filter whose transfer function is given by

$$H(z) = \frac{1}{1 - 0.5z^{-1}}$$

And whose difference equation is

$$y[n] = x[n] + 0.5y[n-1]$$

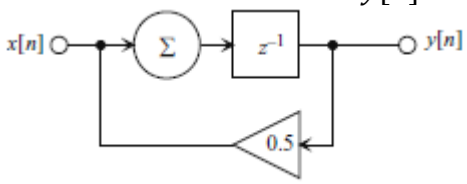


Figure 5 Simple recursive filter with a feedback gain of 0.5

Clearly this has an impulse response given by h[n]=1,0.5,0.25,0.125, …. It is therefore a simple matter to compute that $h[100]=7.889\text{x}10^{-31}$.

FACTORED FORMULATION

If we know the poles and zeros of a filter, we can express it directly in diagram form (Direct Form II) or as a difference equation. For example, a filter whose transfer function is given by

$$H(z) = \frac{1 - a[2]z^{-2}}{1 + b[1]z^{-1} - b[2]z^{-2} + b[4]z^{-4}} \quad \ldots \quad (15)$$

Has a difference equation given by

$$y[n] = x[n] - a[2]x[n-2] - b[1]y[n-1] + b[2]y[n-2] - b[4]y[n-4] \ldots (16)$$

Which diagrammatic form is shown in Figure 6 below.

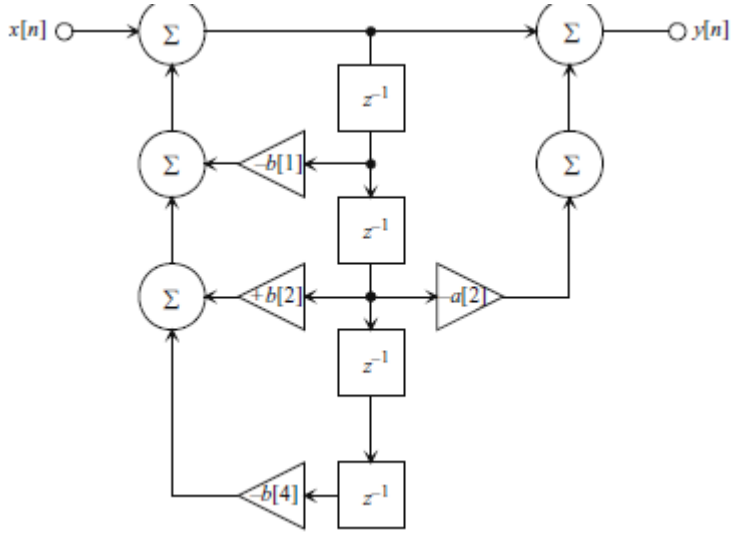


Figure 6 IIR filter diagram of Equation (22). Design of high-order IIR filters is best carried out by cascading low-order sections.

However, many filter designers choose to reconfigure high order filters as a series of cascaded low-order sections so, usually of a second order. This is known as biquad design, and there are good reasons for adopting this approach. First, because the filter involves low negative values of z, the dangers associated with instability are minimized. Additionally, it is possible to derive a general-purpose algorithm that can process each and every stage of the filter, obviating the necessity of changing the code each time the filter is re-designed. Biquad design commences by using the poles and zeros to express the transfer function in factored form. Given an nth order filter with n poles and n zeros, the factored form of the transfer function may be written as a series product of biquad sections, that is,

$$H(z) = \prod_{n=0}^{N-1} \frac{z-\alpha_n}{z-\beta_n}$$
$$= \frac{(z-\alpha_0)(z-\alpha_1)}{(z-\beta_0)(z-\beta_1)} \times \frac{(z-\alpha_2)(z-\alpha_3)}{(z-\beta_2)(z-\beta_3)} \times \dots$$
$$\times \frac{(z-\alpha_{N-2})(z-\alpha_{N-1})}{(z-\beta_{N-2})(z-\beta_{N-1})} \quad \dots. \qquad (17)$$

Now it is often the case when design IIR filters that the poles and zeros are represented as conjugate pairs, that is, $a \pm jb$. If we make this reasonable assumption here, then for each biquad section given by

$$H(z) = \frac{(z-\alpha_0)(z-\alpha_1)}{(z-\beta_0)(z-\beta_1)}, \quad \dots\dots \qquad (18)$$

The complex values of the zeros and poles are

$$\alpha_0 = a_0 + jb_0,$$
$$\alpha_0 = a_0 - jb_0,$$
$$\beta_0 = a_1 + jb_1,$$
$$\beta_1 = a_1 - jb_1, \dots \quad (19)$$

By making the simplification

$$\varepsilon_0 = a_0^2 + b_0^2,$$

$$\varepsilon_1 = a_1^2 + b_1^2, \ldots \quad (20)$$

Then each second order stage is given by

…

And the difference equation for each second order stage is therefore

$$y[n] = x[n] - 2a_0x[n-1] + \varepsilon_0x[n-2] + 2a_1y[n-1] - \varepsilon_1y[n-2]. \ldots. \quad (22)$$

The nice thing about Equation (22) is that to use it, we simply insert different coefficient values, based on the poles and zeros of our high-order filter. If, for example, we started with a tenth order design, the algorithm would simply employ a five-stage loop, within which we would be embedded our second order difference equation. This brings us neatly to our next subject – designing IIR filters using pole-zero placement.

DIFFERENCE EQUATIONS IN SIMULATION OF CIRCUITS

FIRST ORDER CASE

The differential equation is essentially an analytical tool, enabling in this case dynamic characterization of some simple circuits. We have also demonstrated that we can express particular solutions in program form, to map the system's step response or impulse responses; obviously enough, this is termed an analytical approach. However, in its present computational form, there is a problem with it, which is this: not only is the differential equation required, so too is the form of the particular solution. This is because each solution has a different algebraic expression, which is then coded as necessary. But we do not know the form of the solution until we solve the equation manually, using the component values or boundary conditions – it has to be told what the form of the solution is, and then the values of the constants A, B, α and β. In short, we have to solve the differential equation and explicitly encode the various algebraic possibilities.

It would instead be much better if we could write a program that simply used some numerical (rather than analytical) representation of the differential equation that employed the values of L, C and R directly, without the operator having to solve the equation first. This is indeed possible and very easy to achieve. Differencing is affected by subtracting form each signal value the value of its predecessor, that is,

$$x'[n] = y[n] = x[n] - x[n-1] \quad \ldots \quad (1)$$

This equation subtly omits the fact that there is a periodicity to the equation that assumes that the period T=1s. A modification to equation (1) to include periodicity is given as

$$x'[n] = y[n] = \frac{x[n] - x[n-1]}{T} \quad \ldots \quad (2)$$

The smaller T is, the more closely the difference equation will model the analytical case. We can now extend the differencing process to higher order derivatives; the second differential is simply the differential of the first differential, that is,

$$x''[n] = y'[n] = \frac{y[n] - y[n-1]}{T} \quad \ldots \quad (3)$$

Substituting the right-hand side of equation (1) into equation (3) gives

$$x''[n] = y'[n] = \frac{1}{T}\left(\frac{x[n] - x[n-1]}{T} - \frac{x[n-1] - x[n-2]}{T}\right) \quad \ldots (4)$$

That is

$$x''[n] = \frac{x[n] - 2x[n-1] - x[n-2]}{T^2} \quad \ldots \quad (5)$$

By now you probably get the idea that differencing to any order is straightforward, simply involving additions, subtractions and multiplications (the heart of DSP0. Next let us apply what we have learned to the first order differential equation above, which described the behaviour of the low-pass RC filter. We will manipulate the equation in a slightly different way, to make it easier to get the output, $v_2(t)$. The 1st order equation for an RC circuit is given as

$$v_2(t) = v_1(t) - RC\frac{dv_2(t)}{dt} \quad \ldots \quad (6)$$

We can convert to a differential form as follows:

$$v_2[n] = v_1[n] - \frac{RC}{T}(v_2[n] - v_2[n-1]) \quad \ldots \quad (7)$$

Grouping the $v_2[n]$ terms we get

$$v_2[n]\left(1 + \frac{RC}{T}\right) = v_1[n] + \frac{RC}{T}v_2[n-1] \quad \ldots\ldots \quad (8)$$

Which ultimately yeilds

$$v_2[n] = \frac{T}{T + RC}v_1[n] + \frac{RC}{T + RC}v_2[n-1] \quad \ldots\ldots \quad (9)$$

This is now in the standard form for a first order difference equation.

SECOND ORDER CASE

Once again, the voltage $v_1(t)$ across its inputs is given by

$$v_1(t) = \frac{L}{R}.\frac{dv_2(t)}{dt} + \frac{1}{RC}\int v_2(t)dt + v_2(t) \quad \ldots \quad (10)$$

Again we differentiate to get rid of the integral:

$$\frac{dv_1(t)}{dt} = \frac{L}{R}.\frac{d^2v_2(t)}{dt^2} + \frac{1}{RC}v_2(t) + \frac{dv_2(t)}{dt} \quad \ldots\ldots \quad (11)$$

Finally, make $v_2(t)$ the subject of the equation, that is,

$$v_2(t) = RC\left(\frac{dv_1(t)}{dt} - \frac{dv_2(t)}{dt}\right) - LC.\frac{d^2v_2(t)}{dt^2} \quad \ldots\ldots \quad (12)$$

Expressed in difference form, this is

$$v_2[n] = \frac{RC}{T}[(v_1[n] - v_1[n-1]) - (v_2[n] - v_2[n-1])] - \frac{LC}{T^2}[v_2[n] - 2v_2[n-1] + v_2[n-2]] \quad \ldots\ldots \quad (13)$$

Grouping again the $v_2[n]$ terms gives

$$v[2]\left(1 + \frac{RC}{T} + \frac{LC}{T^2}\right) = \frac{RC}{T}(v_1[n] - v_1[n-1]) + \left(\frac{RC}{T} + \frac{2LC}{T^2}\right)v_2[n-1] - \frac{LC}{T^2}v_2[n-2] \quad \ldots \quad (14)$$

COMPLEX IMPEDANCE, DIFFERENTIAL AND DIFFERENCE EQUATIONS IN DSP

CONVOLUTION AND CORRELATION

If the impulse response h(t) of a linear system is known, then it may be convolved with any input signal x(t) to produce the system's response, denoted as y(t).

The most common purposes of convolution is for filtering. Filters are linear systems and the digital equivalents are achieved using convolution operations. When we use an electronic analog filter with an electrical signal, the precise effect the circuit has on the incoming signal may be both understood an modelled using the convolution integral.

A close relative of convolution is the operation of correlation. It comes in two varieties, auto-correlation and cross-correlation. As its name suggest, auto-correlation involves the correlation of a signal with itself. In contrast, cross-correlation is performed when two different signals are correlated with one another. The reason correlation is closely related to convolution is because, from the mathematical perspective, the operations are almost identical, except for the time-reversal of one of the signals using convolution. Mechanically, the algorithms are the same. Although from a mathematical perspective auto-correlation has some interesting properties, in practice cross-correlation is more often applied in the context of DSP. The concept of cross-correlation is a significant one in respect of signal processing generally but not as pivotal as convolution. The reason why is straightforward: whereas convolution, together with the impulse response, represents a fundamental mechanism for understanding, modelling and simulating systems, cross-correlation is in general restricted to signal enhancement or noise suppression.

The convolution integral is given by

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \quad \ldots\ldots \quad (1)$$

Looking carefully at this equation it is saying that we are multiplying an incoming signal x(t) with an impulse response h(τ) over the range ±∞ and summing (integrating) over the same period. This is sometimes called the bilateral form of the convolution integral, because of the limits of integration. Now for causal systems, which start from some initial point in time, we use the unilateral version, given by

$$y(t) = \int_{0}^{\infty} h(\tau)x(t-\tau)\,d\tau \quad \ldots \quad (2)$$

Now let us make the conversion to discrete space over a finite range. Equation (2) is recast as

$$y[n] = \sum_{k=0}^{M-1} h[k]x[n-k] \quad \ldots \quad (3)$$

Equation (3) may be understood as follows: to obtain a new value of the output signal y[n], we multiply M values of the signal x[n] with M values of the impulse

response h[k] and sum the result. To obtain the next value that is, y[n+1], we increment the index n by one position and repeat the process. The general signal flow pattern for convolution is illustrated in the Figure 7 below, which shows a system with five points in the impulse response h[k].

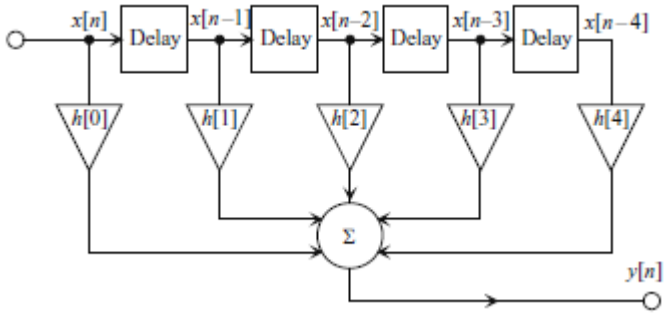


Figure 7 Convolution equivalent also known as five-point Finite Infinite Response (FIR) filter.

Incidentally, you will often see the equation for convolution using short hand notation, that is,

$$y[n] = h[n] * x[n] \quad \ldots. \quad (4)$$

Where the symbol * is termed the convolution operator. But – you may be asking – why does any of this work? Can we prove that this series of arithmetic operations really does yield the right answer? For instance, say we had obtained, though some means, the impulse response of an analog filter, and expressed it as a series of numbers, and also digitized some arbitrary input. If we used equation (3), would the output numbers correspond to the output signal from the filter, as viewed in an oscilloscope screen? Indeed they would. To understand why, we need to look in a little more detail at the nature of discrete signals and by employing some of the properties of linear systems in our subsequent analysis.

USING IMPULSE FUNCTIONS TO REPRESENT DISCRETE FUNCTIONS.

Although it might seem strange when we first encountered the idea, all discrete signals may be represented as a series of shifted, weighted impulse functions (Lynn, 1986).

The discrete time impulse function δ[n], stating that it had a value of zero for all values of n, except when n = 0, in which case δ[n]=n. If we delay the impulse function by k intervals, we may write this new function as δ[n-k]. In addition, we may scale or weight the impulse function by some value, for example a. In this case, the weighted impulse function is given by aδ[n-k].

Now think of a signal, x[n], shown Figure 8 below.

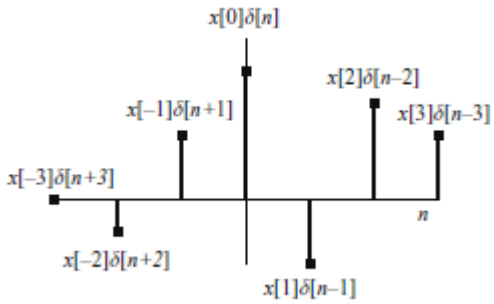


Figure 8 A signal x[n] shown as a series of weighted, shifted impulses

When n is zero, then clearly it has a value of x[0]. This can be thought of as a weighting function, which we apply to an impulse. Therefore, we may rewrite the signal at this instant as x[0]δ[n]. Similarly, when n=1, we can write this as x[1]δ[n-1], and so on. Therefore, the unilateral description of this process (for causal signals) may be written as

$$x[n] = x[0]\delta[n] + x[1]\delta[n-1] + x[2]\delta[n-2] + \cdots = \sum_{k=0}^{\infty} +x[k]\delta[n-k] \ldots \quad (5)$$

More generally, the bilateral version of equation (4) is given by

$$x[n] = \sum_{k=-\infty}^{\infty} +x[k]\delta[n-k] \ldots \quad (6)$$

Equations (5) and (6) must be true, since the function δ[n-k] is only ever equal to 1 when n=k, otherwise it is zero.

DESCRIPTION OF CONVOLUTION

Using Linear Superposition

If a linear system produced y1[n] in response to x1[n], and y2[n] in respsonse to x2[n], then if we add the inputs and then apply them, the system produces a correspondingly summed version of the two original outputs. Moreover, if the inputs are weighted, then the outputs will be weighted by the same amount. These manifestations of superposition and proportionality, are encapsulated in the relationship written as

$$\sum_{k=0}^{\infty} a_k x_k[n] \rightarrow h[n] \rightarrow \sum_{k=0}^{\infty} a_k y_k[n] \quad \ldots \quad (7)$$

Where h[n] represents the linear system (i.e. its impulse response). Now we are getting to the heart of the matter of convolutions, for it involves 3 principles.

1. Any input signal may be described as a series of weighted, shifted impulse functions.
2. Linear systems are both superpositional and proportional.

As an example, imagine we had a simple linear system whose impulse response h[n] was

$$h[0] = 4, \qquad h[1] = 2, \qquad h[2] = 1, \qquad h[3] = 0.5 \qquad \ldots \quad (8)$$

We want to use convolution to predict its output in response to the input signal x[k] given by

$$x[0] = 2, \quad x[1] = 1, \quad x[2] = -1, \quad x[3] = -2 \quad x[4] = 3 \quad \ldots \quad (9)$$

Each signal value x[k] is weighted, shifted impulse function. Thus the system's response when k=0 (i.e. when x[0]=2) will be 8, 4, 2, 1, that is, this is an impulse response weighted by 2. When k=1, the system's response to x[1] will be 4, 2, 1, 0.5.

However, this response will occur one time interval later, since k has advanced one unit. Since the system is still responding to the first value, the overall response to the inputs x[0] and x[1] is obtained by summing the individual responses, with a delay applied to the second response. This process is repeated for each of the input signal values. Grouping the individual responses together and shifting appropriately, we obtained when summed:

8	4	2	1					
	4	2	1	0.5				
		−4	−2	−1	−0.5			
			−8	−4	−2	−1		
				12	6	3	1.5	
$y[n] =$	8	8	0	−8	7.5	3.5	2	1.5	$n = 0, \ldots, 7$

In this example we clearly see how impulse representation, superposition and proportionality are applied in the mechanics of the convolution process.

Using Time Reversal of the Impulse Response

It is worth noting that discrete-time convolution may also be described by the equation

$$y[n] = \sum_{k=0}^{4} x[k]h[n-k] \quad \ldots \quad (10)$$

If you compare this equation with equation (3), you will notice that the positions of x[k] and h[n] have been switched. This does not matter, because the principle of commutativity states that the order of operations in a linear system may be rearranged without affecting the outcome. In this particular case, if we were to swap the position of x[k] and h[n] in equation (10) we would write

$$y[n] = \sum_{k=0}^{3} h[k]x[n-k] \quad \ldots \quad (11)$$

Since k ranges from 0 to 3. In general, however, we can state that

$$\sum_{k=0}^{\infty} x[k]h[n-k] = \sum_{k=0}^{\infty} h[k]x[n-k] \quad \ldots \quad (12)$$

Whether we use equations (3) or (10) makes no difference, since the first case we are time-reversing the signal and in the second we are time reversing the impulse response. Similarly, we can describe the mechanics of convolution in a manner that is slightly different to that of the above scheme. It involves time-

reversal of the impulse response, with its final value aligned with the first value of the signal, which is slightly different to that of the above scheme. It involves time-reversal of the impulse response, with its final value aligned with the first value of the signal, that is, x[0], and the rest of the impulse response extending back in time. We now multiply each value of the impulse response with its corresponding value of the signal. When we have performed this for all values of the impulse response, we sum the products, and the result of this sum represents one new value of the output signal y[n]. Next, the entire time reversed version of the impulse response is shifted one place to the right that is, delayed in time by one sampling interval and the one product summation is repeated to obtain the next value of y[n], and so on. Using the above values for x[n] and h[k], to calculate y[0], the initial position of the signals x[n] and h[-k] would be:

$x[n] =$ (0) (0) (0) 2 1 −1 −2 3

$h[-k] =$ 0.5 1 2 4

This gives y[0] = 0.5 x 0 + 1 x 0 + 2 x 0 + 4 x 2 = 8

Note the first three values of x[n] are in parenthesis, since the values for x[-3] to x[-1] have not been defined. By convention, they are said to be zero. To obtain y[1], the signal positions are

$x[n] =$ (0) (0) (0) 2 1 −1 −2 3

$h[-k] =$ 0.5 1 2 4

This gives y[1]= 0.5 x 0 + 1 x 0 + 2 x 2 + 4 x 1 = 8

If we continue this operation, then we will obtain the same sequence as before, that is, 8,8,0,-8,7.5,3.5,2,1.5. This confirms that the convolution operation may be effected by sliding the time-reversed impulse response through the input signal and accumulating the products at each time interval to obtain the new output point.

REFERENCES

Ackroyd, M. H. (1973). *Digital filters*. London: Butterworths. Retrieved from

http://hud.summon.serialssolutions.com/2.0.0/link/0/eLvHCXMwbV1N

SwMxEB2sQhGKuH7EVoX-gd3uJtvN5twPCupN0FtJNo3upSBd_78z2axo2-

PkkBnCkJfM5L0ACJ6k8d6eoKleobVCNBVTV6RSpdaWqSmySnHJ7X9hpl_Z

7a7w9vltA7Wq0g11NiemNhPu5dN70JOSrl3Zm_CUcwRB303t5HWCTbqA

OM8f9FhewikxCiI42WyvoP8SutnXcDGvP-

jPjrGrqWe9u4HhcvE6W8U4wTrUVdate34LA02P0beNJ61ZBmcOM2fDaD
dn6IlB_109z8vV06w1o85Mdp5ZlXw1DMHDJ15cJPIOxrrIK2UN4oub5q40
eIjJrVOyMqTfJtwQosNQRscG7-E8U1K0VYOHENujX4ofDfJrAA

Lynn, P. A. (1986). *Electronic signals and systems*. Basingstoke: Macmillan Education. Retrieved from http://hud.summon.serialssolutions.com/2.0.0/link/0/eLvHCXMwbV3JDoIwEJ24XEw8SNSIW_gBFVqh9GwgfoCJR9I1evEi_r9TBGLQ3trDZJous3TeKwAl-3DXuRNiHuI5Mugea4UerpCKx1YYlZBUp5Z2iJla2u0m8XZ76RpapUTpXjYP8i4PDkHvsHt9xtzejq60eo7Fhj4PJTW9TtOP0HqgnC_rkU9g4BAFHvTMYwrbrP14JnDFE7j8AUbzwYdS-TkDP88up_MOhRR1bqX4qEDmMBauIP1RVsA1vYCAJJRxy0KDNv-oJBWMKGOptTpmWlLug_craPlvcAWjiKd13L-GoUXFzKaazBucOGAT

Lynn, P. A., & Fuerst, W. (1998). *Introductory digital signal processing with computer applications*. Chichester: Wiley. Retrieved from http://hud.summon.serialssolutions.com/2.0.0/link/0/eLvHCXMwbV3JTsMwEB1RkBAVB8JiyiL5B5KmdoudI0JUINQbEohLZTs29FKWhgN_z4yTVGE5RRNFySia5CUz7z0DSJHl6a93QumDNhcBnyUdxvSJGjD2UlgrrZ3k_qcx09p2u228vXyWjbTKmYomm0O7sEOJ2KhUD3qqIDLf6EFGybkaIcRiqbb2Om2M6IHn6aDHdA82SVGQwIZf7kO_4wB4AJe3xBQnz9XXjy9eLp5pDQ9OpArcvNUkfjyQU7uUu2YFBt6dOh_CYHp9f3WT4lXnTTNmXucsjmDXEIN9WUWlW8lgK2C5eUYQwDA9BtuPxZ2ePYlZHSZtmK2iHCt7rxgiTqzWdJKNj4E7GwqhrVNO08zEWFfk3ohSu-Ak_igNIPmbysl_O09hp9beUavhrMntPN6_bzpNgss

Stroud, K. A., & Booth, D. J. (2011). *Advanced engineering mathematics*. Basingstoke: Palgrave Macmillan. Retrieved from http://hud.summon.serialssolutions.com/2.0.0/link/0/eLvHCXMwY2AwNtIz0EUrE4yBrdLUFAvLlLRU46S0xOSUFNOkVANTiyRDo2RjyCZ4pIOZ4MduwwbeMkpToFurkhNLQDOb-kmZSfoWZmYWhmbMDMzmlqDFflbhoEsbQG1qI3NgO9wMerwOjA-qPYDmINUeboIMLKAdBUIMTKl5wgzcvvCDUotFGBQcoXPwCqmIgwEVchFKRBkk3VxDnD10gYbGQ8da4iFOMhJjYAF23lMlGBSA9adpqmmagWmyiZlJUnKyRaqliWGqhXlKIkipiaUkgxCmfilsgtIMXJDxTBCSYWBNAybPVFmwnwD70WER