Discrete-time variable

Diego Trapero

Table of contents

1 Sampling

Signals

Value	Time Continuous	Time discrete
Value continuous	Analog	Sampled signals (DV/DT)
Value discrete	DV/CT (rare)	Digital

Sampling. Sampling is the reduction of a continuous time signal to a discrete time signal by taking values at different instants of time.

Sampling at a constant rate: for functions that vary with time, let $s(t)$ be a continuous function (or signal) to be sampled, and let sampling be performed by measuring the value of the continuous function every T units of time, which is called the sampling interval. Thus, the sampled function is given by the sequence $s(nT)$, for integer values of n, $n = 0, 1, 2, 3, ...$ #

$$s = \{s(0), s(T), s(2T), s(3T), ...\}$$

- Sampling period, T_s is the time interval between samples.
- Sampling frequency f_s is the is the number of samples per time unit, or the sampling rate. It is the reciprocal of the period, $f_s = 1/T_s$
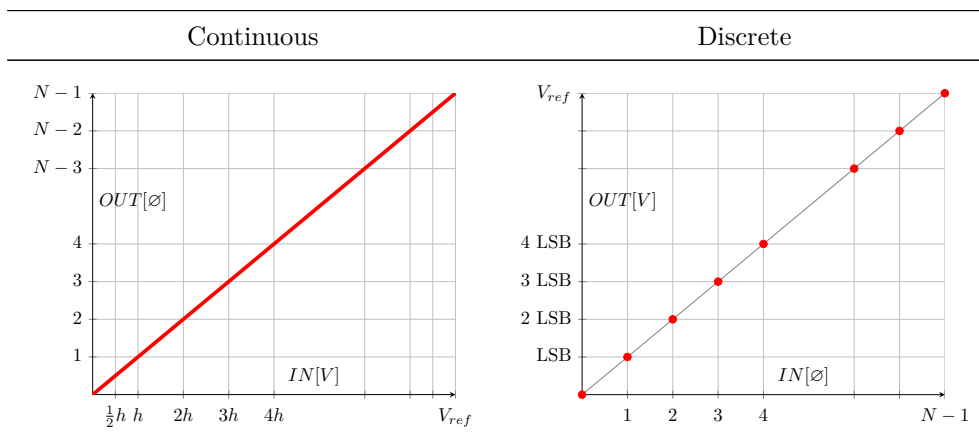
Aliasing Aliasing is the effect that occurs when sampling a continuous signal, whereby frequency components higher than half the sampling frecuency are transformed into lower frequency components. These components, or aliases, (which where a valid part of a signal, or noise) may appear in the band of the signal that we are interested in, thus making the reconstruction of the original signal impossible. From the time-domain point of view, aliasing occurs when we sample a continuous signal at a too low sample rate. The resulting sequence is an alias of the original signal, with lower frequency components.

- From the *time domain point of view*, aliasing occurs when trying to reconstruct a high frequency signal from the samples taken in a not high enough frequency. In this case, there are few low frequency values that can be used to reconstruct many different high frequency signals (all of them would output the same sequence when sampled), aliases of one another.

- From the *frequency domain point of view*, sampling a signal at sampling frequency f_s "folds" any frecuency higher than $f_s/2$ back into the frequency range $0 - f_s/2$. In other words, signal components with frequencies higher than $f_s/2$ will produce the same samples as a signal with some frequency in the range $0, f_s/2$. $f_s/2$ is known as Nyquist frequency or folding frequency. This effect is expressed in the form of the Nyquist theorem.

Nyquist theorem If s(t) is a continuous signal with finite bandwith, that contains no frequencies higher than f_{max}, it can be perfectly reconstructed from a sampling sequency taken at a sampling rate $f_s \geq 2f_{max}$.

- f_{max} is the maximum frequency of the signal
- f_s is the sampling frequency
- $f_s/2 = f_N$ is the Nyquist frecuency

Shannon's enunciation: If a function x(t) contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced 1/(2B) seconds apart.

Aliased frequency of a real world frequency To obtain the aliased frecuency of a high frequency component of a signal, substract f_s repeatedly from the frequency until you have a result in the range $-f_s/2, f_s/2$. Then, take the absolute value. This rule defines different frequency zones that are considered to understand the problem of aliasing:

Choosing the sampling frequency If we hace a signal with a known upper frequency of interest, we can double that frequency to get the Nyquist rate; that is, the sampling rate above wich you must sample in order to avoid aliasing.

- If the maximum frequency of the signal, f_{max} is known, $f_s > 2f_{max}$
- If the maximum frecuency is not known, but an histogram of the signal is available, we can determine the sampling frecuency from the maximum slope point of the signal. This method gives very conservative results, often well above the minimum necessary f_s for the signal.

Antialiasing filters To avoid the problem of aliasing, a low-pass filter called antialising filter can be used before sampling to attenuate any frecuencies that may cause aliasing.

The filters represented in the figure are 1. An ideal (infinite order) filter, that only let frecuencies in the range $[0, f_M]$ pass (the useful part of the signal). 2. A filter that let pass frecuencies, that let frecuencies between $[0, f_N]$ pass (the complete alias-free zone). 3, A filter that let pass frecuencies between 0 and $f_s - f_M$. This filter let pass all the frecuencies before the aliases zones, where aliases may be mapped to the $[0, f_M]$ range (the useful components of the signal). This is the most used filter because it is completely functional while having the minimum order.

**Designing a Antialising Filter for sampling with a ADC**

1. The gain in the $[0, f_M]$ would ideally be 0dB (no gain/attenuation), to let the useful part of the signal pass unaltered. Other constant gain may be considered when increasing or decreasing the amplitude of the signal before the ADC is necessary.

2. All components after the $f_s - f_M$ frecuency must be attenuated before the resolution of the ADC, so they cannot be registered. This will prevent any aliases mapping to the $[0, f_M]$ band.

In the worst case, a signal with amplitude $V_r ef$
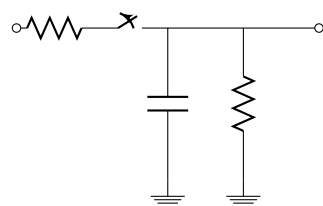
Example For a analog signal with $f_{max} = 1000Hz$, sampled at 5000 Hz, determine the order of the filter necessary to avoid aliasing in a 8 bits ADC.

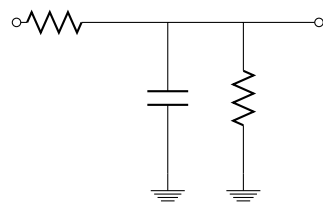# 2   Sample and hold

Ideal S/H model
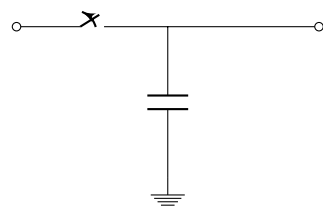
Real S/H model

Ideal S/H operation

Real S/H operation

Sample phase

Hold phase
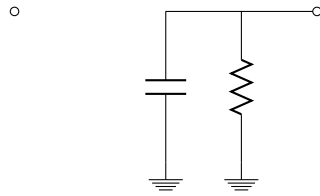
Transition between S/H

S/H complete cycle

S/H circuits

3   AD Converters

3.1   Quantification
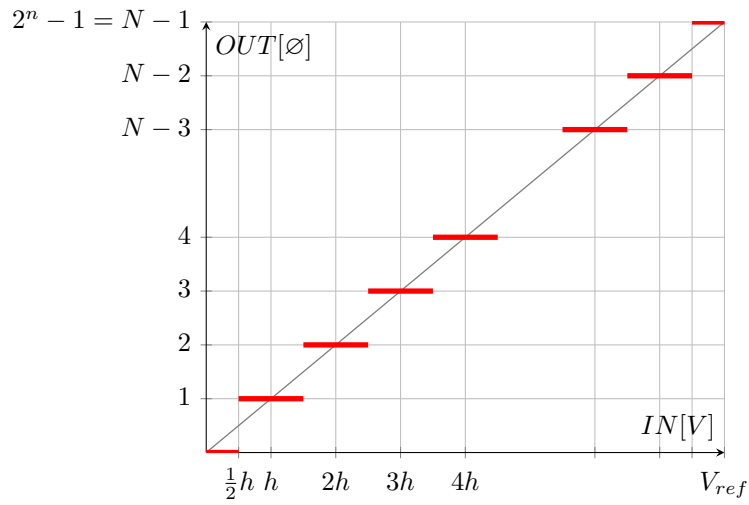
$$\text{LSB}_{[V]} = h = \frac{V_{ref}}{N-1} = \frac{V_{ref}}{2^n - 1}$$

$$\text{LSB}_N = \frac{1}{N-1} = \frac{1}{2^n - 1}$$

Example, 3 bit (8 levels) ADC $V_{ref} = 5V$
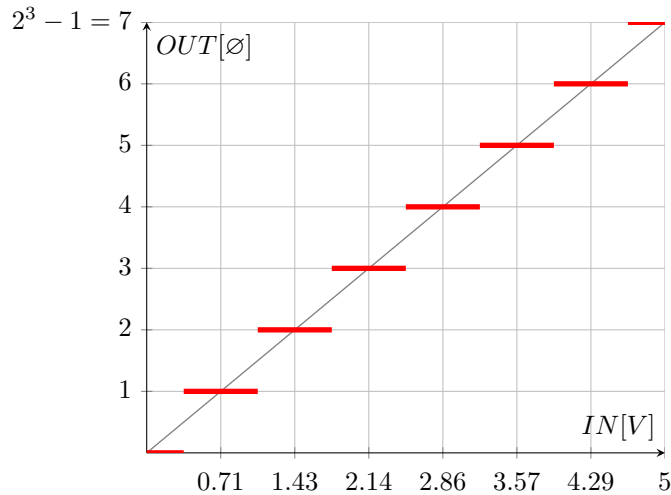
3.2   AD converter types

3.2.1   Flash ADC

The flash converter employs several comparators with fixed reference voltages (obtained with a large voltage divisor) to obtain a thermometric code from the input signal. An encoder can next convert to binary code.

3.2.2 Half-Flash ADC

Flash converter is fast but uses many components (growing exponentially with the number of bits). For avoiding this problem, the half-flash converts $q + p$ bits using two flash-converters of q and p bits.

Half-Flash vs Flash components comparation For 8 bits:

- Flash
    - $2^8 - 1 = 255$ comparators
    - $2^8 = 256$ resistors
- Half-Flash
    - $2 \cdot (2^4 - 1) = 30$ comparators
    - $2 \cdot (2^4) = 32$ resistors

3.2.3 Successive approximation ADC

The succesive approximation ADC is an ADC that can convert an analog signal in a digital code of 2^n levels in n approximations (or tries). It obtains the code bit by bit, starting from the MSB.

The control circuit starts generating the code for 1 MSB and all zeros, and outputing it to the DAC. The output of the DAC is then compared with the analog signal. If the analog signal is greater than the DAC output, the control circuit recives a "greater" signal from the comparator, and knows that the first digit is 1. In the other case, the digit is a 0. The converter then tries to get the next digit from the (holded) analog signal until it is completedly converter.

3.2.4 Pipeline ADC

4   AD Times

Concurrent times

$$t = \max\{t_1, t_2, t_3, ...\}$$

$$t = \sqrt{t_1^2 + t_2^2 + t_3^2 + ...}$$

Sequential times

$$t = t_1 + t_2 + t_3 + ...$$

AD Conversion times An ADC needs its input analog signal to be constant during its convertion time. The minimum time is the following, that determines the sample frecuency:

$$T_{min} = (t_{adquisition} + t_{setling} + t_{ap.delay})_{SH} + t_{conv}$$

$$f_s \leq \frac{1}{T_{min}}$$

Times

- t_{MUX}, MUX conmutation time
- t_{sample}, delay in sampling (until output constant is constant)

$$t_{sample} = t_{switchcloses} + t_{samplingsettling}$$

- t_{hold}, delay in holding.

$$t_{hold} = t_{switchopens} + t_{holdsettling}$$

- t_{PGA}, programmable amplifier settling time.
- t_{conv}, ADC conversion time

5 DA Converters

5.1 Codification
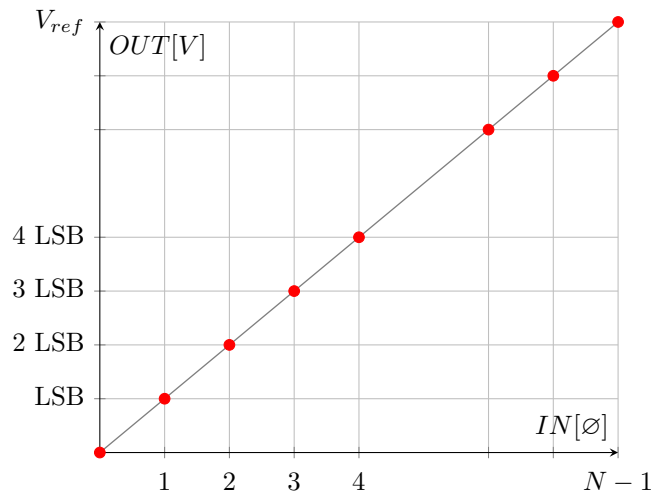
5.2 AD converter types

5.2.1 Weighted resistor DAC

5.2.2 R-2R Ladder

6 ADC / DAC tables

	ADC	DAC
Process	Quantification 	Codification
Static		
Dynamic		

Errors	14	
Types		

7 Z transform

In mathematics and signal processing, the Z-transform converts a time domain signal, which is a sequence of real or complex numbers, into a complex frequency domain representation.

It can be considered as a discrete-time equivalent of the Laplace transform. #

Definition The unilateral Z-transform of a discrete-time signal (sequence) $f(n)$ is the formal power series $F(z)$ defined as

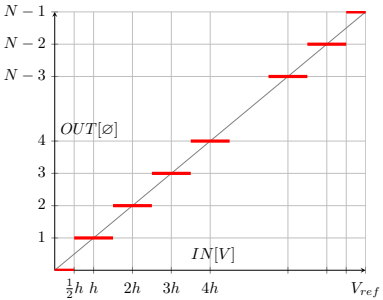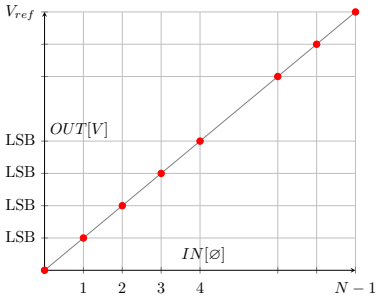$$F(z) = \mathcal{Z}[f(n)](z) = \sum_{n=0}^{\infty} f(n) z^{-n}$$

Property of the Z-transform The following property of the Z-transform make it useful for solving and manipulating difference equations:

$$\mathcal{Z}[f(n-k)](z) = z^{-k} F(z)$$

7.1 Difference equations

$$\begin{aligned} y(n) =& b_0(n) + b_1 x(n-1) + ... + b_z(n-z) \\ & - a_1 y(n-1) - a_2 y(n-2) - ... - a_p y(n-p) \end{aligned}$$

$$y(n) = b_0(n) + b_1 x(n-1) + ... + b_z(n-z) - a_1 y(n-1) - a_2 y(n-2) - ... - a_p y(n-p)$$

Example $y(n) = 3x(n) + 5x(n-1) + 7x(n-2) - 4y(n-1) - 6y(n-2)$

1. Compute $y(n)$ for $x = \{0, 0, 1, 0, 0, 0, 0\}$
2. Calculate the z-domain transfer function, $F(z) = \frac{Y(z)}{X(z)}$

Example Calculate the difference equation for this transfer function

$$F(z) = \frac{Y(z)}{X(z)} = \frac{1 + 2z^{-1} + z^{-2}}{6.9 - 4.6z^{-1} + 1.7z^{-2}}$$

Example Calculate the difference equation for this transfer function

$$F(z) = \frac{Y(z)}{X(z)} = \frac{1 + 2z^{-1}}{2 - z^{-1} + z^{-2}}$$

7.2   Bilinear transform

The bilinear transform (also known as Tustin's method) is used in digital signal processing and discrete-time control theory to transform continuous-time system representations to discrete-time and vice versa. #

The transform preserves stability and maps every point of the frequency response of the continuous-time filter, $H_a(j\omega_a)$ to a corresponding point in the frequency response of the discrete-time filter, $H_d(e^{j\omega_d T})$ although to a somewhat different frequency, as shown in the Frequency warping section below. This means that for every feature that one sees in the frequency response of the analog filter, there is a corresponding feature, with identical gain and phase shift, in the frequency response of the digital filter but, perhaps, at a somewhat different frequency. This is barely noticeable at low frequencies but is quite evident at frequencies close to the Nyquist frequency.

The bilinear transform is a first-order approximation of the natural logarithm function that is an exact mapping of the z-plane to the s-plane. When the Laplace transform is performed on a discrete-time signal (with each element of the discrete-time sequence attached to a correspondingly delayed unit impulse), the result is precisely the Z transform of the discrete-time sequence with the substitution of

$$z = e^{sT}$$
$$= \frac{e^{sT/2}}{e^{-sT/2}}$$
$$\approx \frac{1 + sT/2}{1 - sT/2}$$

where T is the numerical integration step size of the trapezoidal rule used in the bilinear transform derivation. The above bilinear approximation can be solved for s or a similar approximation for $s = (1/T)\ln(z)$ can be performed.

The inverse of this mapping (and its first-order bilinear approximation) is

$$s = \frac{1}{T}\ln(z)$$
$$= \frac{2}{T}\left[\frac{z-1}{z+1} + \frac{1}{3}\left(\frac{z-1}{z+1}\right)^3 + \frac{1}{5}\left(\frac{z-1}{z+1}\right)^5 + \frac{1}{7}\left(\frac{z-1}{z+1}\right)^7 + \cdots \right]$$
$$\approx \frac{2}{T}\frac{z-1}{z+1}$$
$$= \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}}$$

The bilinear transform essentially uses this first order approximation and substitutes into the continuous-time transfer function, $H_a(s)$

$$s \leftarrow \frac{2}{T} \frac{z-1}{z+1}.$$

That is

$$H_d(z) = H_a(s) \Big|_{s=\frac{2}{T} \frac{z-1}{z+1}} = H_a \left(\frac{2}{T} \frac{z-1}{z+1} \right)$$

Example: RC low-pass filter As an example take a simple low-pass RC filter. This continuous-time filter has a transfer function

$$H_a(s) = \frac{1/sC}{R + 1/sC}$$
$$= \frac{1}{1 + RCs}$$

If we wish to implement this filter as a digital filter, we can apply the bilinear transform by substituting for s the formula above; after some reworking, we get the following filter representation:

$$H_d(z)$$
$$= H_a \left(\frac{2}{T} \frac{z-1}{z+1} \right)$$
$$= \frac{1}{1 + RC \left(\frac{2}{T} \frac{z-1}{z+1} \right)}$$
$$= \frac{1 + z}{(1 - 2RC/T) + (1 + 2RC/T)z}$$
$$= \frac{1 + z^{-1}}{(1 + 2RC/T) + (1 - 2RC/T)z^{-1}}$$

The coefficients of the denominator are the 'feed-backward' coefficients and the coefficients of the numerator are the feed-forward coefficients used to implement a real-time digital filter.

Frecuency warping

To determine the frequency response of a continuous-time filter, the transfer function $H_a(s)$ is evaluated at $s = j\omega$ which is on the $j\omega$ axis. Likewise, to determine the frequency response of a discrete-time filter, the transfer function $H_d(z)$ is evaluated at $z = e^{j\omega T}$ which is on the unit circle, $|z| = 1$. When the actual frequency of ω is input to the discrete-time filter designed by use of the bilinear transform, it is desired to know at what frequency, ω_a, for the continuous-time filter that this ω is mapped to.

$$H_d(z) = H_a \left(\frac{2}{T} \frac{z-1}{z+1} \right)$$

$$H_d(e^{j\omega T})$$
$$= H_a \left(\frac{2}{T} \frac{e^{j\omega T} - 1}{e^{j\omega T} + 1} \right)$$
$$= H_a \left(\frac{2}{T} \cdot \frac{e^{j\omega T/2} \left(e^{j\omega T/2} - e^{-j\omega T/2} \right)}{e^{j\omega T/2} \left(e^{j\omega T/2} + e^{-j\omega T/2} \right)} \right)$$
$$= H_a \left(\frac{2}{T} \cdot \frac{\left(e^{j\omega T/2} - e^{-j\omega T/2} \right)}{\left(e^{j\omega T/2} + e^{-j\omega T/2} \right)} \right)$$

$$= H_a \left(j\frac{2}{T} \cdot \frac{\left(e^{j\omega T/2} - e^{-j\omega T/2} \right)/(2j)}{\left(e^{j\omega T/2} + e^{-j\omega T/2} \right)/2} \right)$$

$$= H_a \left(j\frac{2}{T} \cdot \frac{\sin\left(()\,\omega T/2\right)}{\cos\left(()\,\omega T/2\right)} \right)$$

$$= H_a \left(j\frac{2}{T} \cdot \tan()\,(\omega T/2) \right)$$

This shows that every point on the unit circle in the discrete-time filter z-plane, $z = e^{j\omega T}$ is mapped to a point on the $j\omega$ axis on the continuous-time filter s-plane, $s = j\omega_a$. That is, the discrete-time to continuous-time frequency mapping of the bilinear transform is

$$\omega_a = \frac{2}{T}\tan()\left(\omega\frac{T}{2} \right)$$

and the inverse mapping is

$$\omega = \frac{2}{T}\arctan\left(\omega_a\frac{T}{2} \right)$$

The discrete-time filter behaves at frequency ω the same way that the continuous-time filter behaves at frequency $(2/T)\tan(()\,\omega T/2)$. Specifically, the gain and phase shift that the discrete-time filter has at frequency ω is the same gain and phase shift that the continuous-time filter has at frequency $(2/T)\tan(()\,\omega T/2)$. This means that every feature, every "bump" that is visible in the frequency response of the continuous-time filter is also visible in the discrete-time filter, but at a different frequency. For low frequencies (that is, when $\omega \ll 2/T$ or $\omega_a \ll 2/T$), $\omega \approx \omega_a$.

One can see that the entire continuous frequency range

$$-\infty < \omega_a < +\infty$$

is mapped onto the fundamental frequency interval

$$-\frac{\pi}{T} < \omega < +\frac{\pi}{T}.$$

The continuous-time filter frequency $\omega_a = 0$ corresponds to the discrete-time filter frequency $\omega = 0$ and the continuous-time filter frequency $\omega_a = \pm\infty$ correspond to the discrete-time filter frequency $\omega = \pm\pi/T$.

One can also see that there is a nonlinear relationship between ω_a and ω. This effect of the bilinear transform is called frequency warping. The continuous-time filter can be designed to compensate for this frequency warping by setting $\omega_a = \frac{2}{T}\tan()\left(\omega\frac{T}{2} \right)$ for every frequency specification that the designer has control over (such as corner frequency or center frequency). This is called pre-warping the filter design.

When designing a digital filter as an approximation of a continuous time filter, the frequency response (both amplitude and phase) of the digital filter can be made to match the frequency response of the continuous filter at frequency ω_0 if the following transform is substituted into the continuous filter transfer function. This is a modified version of Tustin's transform shown above. However, note that this transform becomes the above transform as $\omega_0 \to 0$. That is to say, the above transform causes the digital filter response to match the analog filter response at DC.

$$s \leftarrow \frac{\omega_0}{\tan()\left(\frac{\omega_0 T}{2}\right)}\frac{z-1}{z+1}$$

The main advantage of the warping phenomenon is the absence of aliasing distortion of the frequency response characteristic, such as observed with Impulse invariance. It is necessary, however, to compensate for the frequency warping by pre-warping the given frequency specifications of the continuous-time system. These pre-warped specifications may then be used in the bilinear transform to obtain the desired discrete-time system.

7.3   Digital filters

Digital filters are classified into one of two basic forms, according to how they respond to a unit impulse:

- *Finite impulse response*, or FIR, filters express each output sample as a weighted sum of the last N input samples, where N is the order of the filter. FIR filters are normally non-recursive, meaning they do not use feedback and as such are inherently stable. A moving average filter or CIC filter are examples of FIR filters that are normally recursive (that use feedback). If the FIR coefficients are symmetrical (often the case), then such a filter is linear phase, so it delays signals of all frequencies equally which is important in many applications. It is also straightforward to avoid overflow in an FIR filter. The main disadvantage is that they may require significantly more processing and memory resources than cleverly designed IIR variants. FIR filters are generally easier to design than IIR filters - the Parks-McClellan filter design algorithm (based on the Remez algorithm) is one suitable method for designing quite good filters semi-automatically.

- *Infinite impulse response*, or IIR, filters are the digital counterpart to analog filters. Such a filter contains internal state, and the output and the next internal state are determined by a linear combination of the previous inputs and outputs (in other words, they use feedback, which FIR filters normally do not). In theory, the impulse response of such a filter never dies out completely, hence the name IIR, though in practice, this is not true given the finite resolution of computer arithmetic. IIR filters normally require less computing resources than an FIR filter of similar performance. However, due to the feedback, high order IIR filters may have problems with instability, arithmetic overflow, and limit cycles, and require careful design to avoid such pitfalls. Additionally, since the phase shift is inherently a non-linear function of frequency, the time delay through such a filter is frequency-dependent, which can be a problem in many situations. 2nd order IIR filters are often called 'biquads' and a common implementation of higher order filters is to cascade biquads. A useful reference for computing biquad coefficients is the RBJ Audio EQ Cookbook. #

7.3.1   IIR filters

Infinite impulse response (IIR) is a property applying to many linear time-invariant systems. Common examples of linear time-invariant systems are most electronic and digital filters. Systems with this property are known as IIR systems or IIR filters, and are distinguished by having an impulse response which does not become exactly zero past a certain point, but continues indefinitely. This is in contrast to a finite impulse response in which the impulse response h(t) does become exactly zero at times t > T for some finite T, thus being of finite duration.

In practice, the impulse response even of IIR systems usually approaches zero and can be neglected past a certain point. However the physical systems which give rise to IIR or FIR (finite impulse response) responses are dissimilar, and therein lies the importance of the distinction. For instance, analog electronic filters composed of resistors, capacitors, and/or inductors (and perhaps linear amplifiers) are generally IIR filters. On the other hand, discrete-time filters (usually digital filters) based on a tapped delay line employing no feedback are necessarily FIR filters. The capacitors (or inductors) in the analog filter have a "memory" and their internal state never completely relaxes following an impulse. But in the latter case, after an impulse has reached the end of the tapped delay line, the system has no further memory of that impulse and has returned to its initial state; its impulse response beyond that point is exactly zero.

Implementation and design Although almost all analog electronic filters are IIR, digital filters may be either IIR or FIR. The presence of feedback in the topology of a discrete-time filter (such as the block diagram shown below) generally creates an IIR response. The z domain transfer function of an IIR filter contains a non-trivial denominator, describing those feedback terms. The transfer function of an FIR filter, on the other hand, has only a numerator as expressed in the general form derived

below. All of the a_i coefficients (feedback terms) are zero and the filter has no finite poles.

The transfer functions pertaining to IIR analog electronic filters have been extensively studied and optimized for their amplitude and phase characteristics. These continuous-time filter functions are described in the Laplace domain. Desired solutions can be transferred to the case of discrete-time filters whose transfer functions are expressed in the z domain, through the use of certain mathematical techniques such as the bilinear transform, impulse invariance, or pole–zero matching method. Thus digital IIR filters can be based on well-known solutions for analog filters such as the Chebyshev filter, Butterworth filter, and Elliptic filter, inheriting the characteristics of those solutions.

Advantages and disadvantages The main advantage digital IIR filters have over FIR filters is their efficiency in implementation, in order to meet a specification in terms of passband, stopband, ripple, and/or roll-off. Such a set of specifications can be accomplished with a lower order (Q in the above formulae) IIR filter than would be required for an FIR filter meeting the same requirements. If implemented in a signal processor, this implies a correspondingly fewer number of calculations per time step; the computational savings is often of a rather large factor.

On the other hand, FIR filters can be easier to design, for instance, to match a particular frequency response requirement. This is particularly true when the requirement is not one of the usual cases (high-pass, low-pass, notch, etc.) which have been studied and optimized for analog filters. Also FIR filters can be easily made to be linear phase (constant group delay vs frequency), a property that is not easily met using IIR filters and then only as an approximation (for instance with the Bessel filter). Another issue regarding digital IIR filters is the potential for limit cycle behavior when idle, due to the feedback system in conjunction with quantization.#

IIR Design Algorithm

1. Take a transfer function in the Laplace domain, $H(s)$.

For Butterworth filters, use $H(s) = \frac{G_0}{B_n(a)}$, where $a = \frac{s}{\omega_c}$ and $B_n(s)$ can be extracted for the following table:

Order	Polynomial $B_n(s)$
1	$(s + 1)$
2	$s^2 + 1.4142s + 1$
3	$(s + 1)(s^2 + s + 1)$
4	$(s^2 + 0.7654s + 1)(s^2 + 1.8478s + 1)$
5	$(s + 1)(s^2 + 0.6180s + 1)(s^2 + 1.6180s + 1)$
6	$(s^2 + 0.5176s + 1)(s^2 + 1.4142s + 1)(s^2 + 1.9319s + 1)$
7	$(s + 1)(s^2 + 0.4450s + 1)(s^2 + 1.2470s + 1)(s^2 + 1.8019s + 1)$
8	$(s^2 + 0.3902s + 1)(s^2 + 1.1111s + 1)(s^2 + 1.6629s + 1)(s^2 + 1.9616s + 1)$

2. Correct the cut-off frecuency of the filter using pre-warping design.

    2.1 The z-domain cut-off frequency of the filter is

$$\omega_z = 2\pi \frac{f_c}{f_s}$$

    2.2 The s-domain frequency we have to use to compensate for warping is

$$\omega_{pw} = \frac{2}{T} \tan\left(\frac{\omega_z}{2}\right)$$

3. Take the bilinear transformation of the transfer function $H(s)$ with the calculated cut-off frequency, ω_{pw}:

$$H_d(z) = H_a(s)\Big|_{s=\frac{2}{T}\frac{z-1}{z+1}} = H_a\left(\frac{2}{T}\frac{z-1}{z+1}\right)$$

Transfer function in s	Transfer function in z	Coefficients
First order $H(s) = \dfrac{1}{as+1}$	$H(s) = \dfrac{1+z^{-1}}{b_0 + b_1 z^{-1}}$	$b_0 = 1 + \dfrac{2a}{T}$ $b_1 = 1 - \dfrac{2a}{T}$
Second order $H(s) = \dfrac{1}{as^2 + bs + 1}$	$H(s) = \dfrac{1 + 2z^{-1} + z^{-2}}{b_0 + b_1 z^{-1} + b_2 z^{-2}}$	$b_0 = \dfrac{4a}{T^2} + \dfrac{2b}{T} + 1$ $b_1 = 2 - \dfrac{8a}{T^2}$ $b_2 = \dfrac{4a}{T^2} - \dfrac{2b}{T} + 1$

4. For implementing the digital filter, transform in a difference equation of the form

$$\begin{aligned} y(n) = &b_0(n) + b_1 x(n-1) + \ldots + b_z(n-z) \\ &- a_1 y(n-1) - a_2 y(n-2) - \ldots - a_p y(n-p) \end{aligned}$$

using the Z-transform property

$$\mathcal{Z}[f(n-k)](z) = z^{-k} F(z)$$

.

8 Reference

- "Digital Signal Processing and the Microcontroller", Dale Grover, John Deller
- "Instrumentación Electrónica", Miguel A. Pérez, Juan C. Álvarez, Juan C. Campo, Fco. Javier Ferrero, Gustavo J. Grillo. Editorial Thomson
- Wikipedia
- Sampling (signal processing), Wikipedia
- Butterworth filter
- Z-transform

9 More

Butterworth filter. Maximum flatness filter. The normalized Butterworth polynomials can be used to determine the transfer function for any low-pass filter cut-off frequency ω_c, as follows

$$H(s) = \frac{G_0}{B_n(a)}$$

where $a = \frac{s}{\omega_c}$.

To four decimal places, Butterworth polynomials are (expressed in quadratic factors)

Order	Polynomial $B_n(s)$
1	$(s+1)$
2	$s^2 + 1.4142s + 1$
3	$(s+1)(s^2 + s + 1)$
4	$(s^2 + 0.7654s + 1)(s^2 + 1.8478s + 1)$
5	$(s+1)(s^2 + 0.6180s + 1)(s^2 + 1.6180s + 1)$
6	$(s^2 + 0.5176s + 1)(s^2 + 1.4142s + 1)(s^2 + 1.9319s + 1)$
7	$(s+1)(s^2 + 0.4450s + 1)(s^2 + 1.2470s + 1)(s^2 + 1.8019s + 1)$
8	$(s^2 + 0.3902s + 1)(s^2 + 1.1111s + 1)(s^2 + 1.6629s + 1)(s^2 + 1.9616s + 1)$

From Butterworth filter, Wikipedia