Discrete-time variable

Diego Trapero

03/12/2013 draft

Table of contents

1 Sampling **2**

2 Signal reconstruction **4**

3 Sample and hold **4**

4 AD Converters **4**

4.1 AD converter types 4

4.1.1 Flash ADC 5

4.1.2 Half-Flash ADC 5

4.1.3 Successive approximation ADC 6

4.1.4 Pipeline ADC 6

5 DA Converters **6**

6 Z transform **6**

7 Digital filters **7**

7.1 IIR filters 7

7.2 FIR filters 7

8 Reference **7**

9 More **7**

1 Sampling

Signals

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, $n = 0, 1, 2, 3, ...$ #

- 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_ref

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 Signal reconstruction

3 Sample and hold

4 AD Converters

4.1 AD converter types

4.1.1 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.

4.1.2 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

4.1.3 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.

4.1.4 Pipeline ADC

5 DA Converters

6 Z transform

7 Digital filters

Difference equation

$$y(n) = b_0(n) + b_1x(n-1) + ... + b_z(n-z) - a_1y(n-1) - a_2y(n-2) - ... - a_py(n-p)$$

7.1 IIR filters

7.2 FIR filters

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, Wikipedia

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