

# Discrete-time variable

Diego Trapero

## Table of contents

<b>1</b>	<b>Sampling</b>	<b>2</b>
<b>2</b>	<b>Signal reconstruction</b>	<b>4</b>
<b>3</b>	<b>Sample and hold</b>	<b>4</b>
<b>4</b>	<b>AD Converters</b>	<b>6</b>
4.1	Quantification . . . . .	7
4.2	AD converter types . . . . .	7
4.2.1	Flash ADC . . . . .	7
4.2.2	Half-Flash ADC . . . . .	7
4.2.3	Successive approximation ADC . . . . .	8
4.2.4	Pipeline ADC . . . . .	8
<b>5</b>	<b>AD Times</b>	<b>9</b>
<b>6</b>	<b>DA Converters</b>	<b>9</b>
6.1	Codification . . . . .	10
6.2	AD converter types . . . . .	10
6.2.1	Weighted resistor DAC . . . . .	10
6.2.2	R-2R Ladder . . . . .	11
<b>7</b>	<b>Z transform</b>	<b>11</b>
<b>8</b>	<b>Digital filters</b>	<b>11</b>
8.1	IIR filters . . . . .	11
8.2	FIR filters . . . . .	11
<b>9</b>	<b>Reference</b>	<b>11</b>
<b>10</b>	<b>More</b>	<b>11</b>

# 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  $n$ ,  $n = 0, 1, 2, 3, \dots$  #

- Sampling period,  $T_s$  is the time interval between samples.
- Sampling frequency  $f_s$  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 frequency 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 frequency 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 bandwidth, that contains no frequencies higher than  $f_{max}$ , it can be perfectly reconstructed from a sampling frequency 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 frequency

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 frequency of a high frequency component of a signal, subtract  $f_s$  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 have 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 which 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 frequency is not known, but an histogram of the signal is available, we can determine the sampling frequency 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 antialiasing filter can be used before sampling to attenuate any frequencies that may cause aliasing.

The filters represented in the figure are 1. An ideal (infinite order) filter, that only let frequencies in the range  $[0, f_M]$  pass (the useful part of the signal). 2. A filter that let pass frequencies, that let frequencies between  $[0, f_N]$  pass (the complete alias-free zone). 3. A filter that let pass frequencies between 0 and  $f_s - f_M$ . This filter let pass all the frequencies 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 Antialiasing 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  $f_s - f_M$  frequency 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.

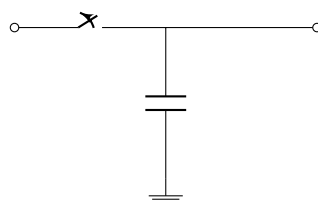
In the worst case, a signal with amplitude  $V_{ref}$

**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.

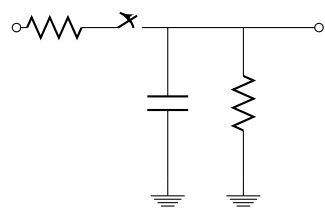
## 2 Signal reconstruction

## 3 Sample and hold

Ideal S/H model



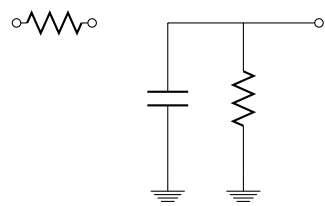
**Real S/H model**



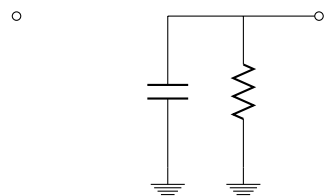
**Ideal S/H operation**

**Real S/H operation**

**Sample phase**



**Hold phase**



Transition between S/H

S/H complete cycle

S/H circuits

## 4 AD Converters

## 4.1 Quantification

## 4.2 AD converter types

### 4.2.1 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.

### 4.2.2 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 comparison 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

#### 4.2.3 Successive approximation ADC

The successive 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 outputting 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 receives 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 completely converted.

#### 4.2.4 Pipeline ADC



## 5 AD Times

### Concurrent times

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

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

### Sequential times

$$t = t_1 + t_2 + t_3 + \dots$$

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

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

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

### Times

- $t_{MUX}$ , MUX commutation 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

## 6 DA Converters

## 6.1 Codification

## 6.2 AD converter types

### 6.2.1 Weighted resistor DAC

## 7 Z transform

## 8 Digital filters

### Difference equation

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

### 8.1 IIR filters

### 8.2 FIR filters

## 9 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

## 10 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  $\omega_c$ , as follows

$$H(s) = \frac{G_0}{B_n(a)} \text{ , where } a = \frac{s}{\omega_c} \text{ .}$$

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](#)