Digital Processing with Focus onto Neutron Detection

SNRI-V INFN, Padova

Gabriele Pasquali
(pasquali@fi.infn.it)

Università di Firenze
and INFN-Sezione di Firenze

25-26 October 2016

Outline

Digital Processing with Focus onto Neutron Detection

Focus: Key issues behind Pulse Shape Discrimination with digitized signals
Aim: understanding few basic facts (not a comprehensive treatment)

Outline:

- First lesson:

Outline

Digital Processing with Focus onto Neutron Detection

Focus: Key issues behind Pulse Shape Discrimination with digitized signals
Aim: understanding few basic facts (not a comprehensive treatment)

Outline:

- First lesson:
  - Introduction: general framework, PSD examples

Outline

Digital Processing with Focus onto Neutron Detection

Focus: Key issues behind Pulse Shape Discrimination with digitized signals
Aim: understanding few basic facts (not a comprehensive treatment)

Outline:

- First lesson:
  - Introduction: general framework, PSD examples
  - ADC as additional noise source

Outline

Digital Processing with Focus onto Neutron Detection

Focus: Key issues behind Pulse Shape Discrimination with digitized signals

Aim: understanding few basic facts (not a comprehensive treatment)

Outline:

- First lesson:
  - Introduction: general framework, PSD examples
  - ADC as additional noise source
  - Signal reconstruction from samples (interpolation)

Outline

Digital Processing with Focus onto Neutron Detection

Focus: Key issues behind Pulse Shape Discrimination with digitized signals

Aim: understanding few basic facts (not a comprehensive treatment)

Outline:

- First lesson:
  - Introduction: general framework, PSD examples
  - ADC as additional noise source
  - Signal reconstruction from samples (interpolation)
- Second lesson:

Outline

Digital Processing with Focus onto Neutron Detection

Focus: Key issues behind Pulse Shape Discrimination with digitized signals
Aim: understanding few basic facts (not a comprehensive treatment)

Outline:

- First lesson:
  - Introduction: general framework, PSD examples
  - ADC as additional noise source
  - Signal reconstruction from samples (interpolation)
- Second lesson:
  - Timing as a study case (role of interpolation *noise*)

Outline

Digital Processing with Focus onto Neutron Detection

Focus: Key issues behind Pulse Shape Discrimination with digitized signals
Aim: understanding few basic facts (not a comprehensive treatment)

Outline:

- First lesson:
  - Introduction: general framework, PSD examples
  - ADC as additional noise source
  - Signal reconstruction from samples (interpolation)
- Second lesson:
  - Timing as a study case (role of interpolation *noise*)
  - Application to n/γ PSD

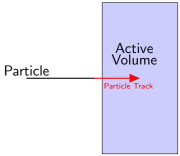
Introduction: sensor, read-out, digitizer

Detection cell: sensor and read-out circuit

- active volume

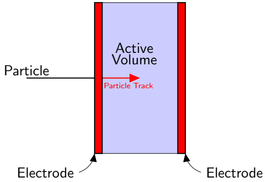
Detection cell: sensor and read-out circuit

- active volume
- energy absorbed $\Longrightarrow$ info carriers



Detection cell: sensor and read-out circuit

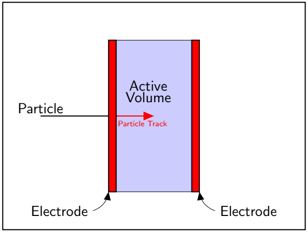
- active volume
- energy absorbed $\Longrightarrow$ info carriers
- carrier collecting device (electrodes, photo-something)



Detection cell: sensor and read-out circuit

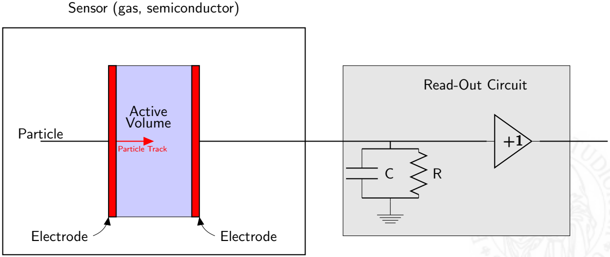
- active volume
- energy absorbed $\Longrightarrow$ info carriers
- carrier collecting device (electrodes, photo-something)
- *Sensor* (t_c: carrier collection time)

Sensor (gas, semiconductor)



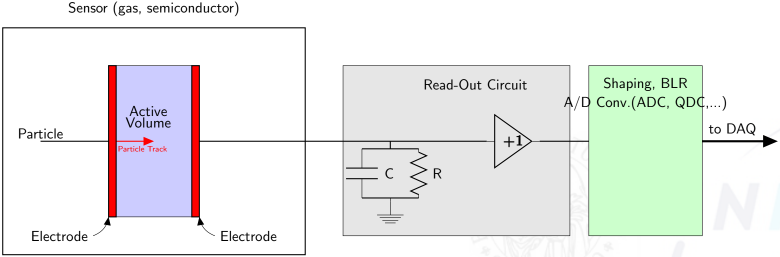
Detection cell: sensor and read-out circuit

- active volume
- energy absorbed $\Longrightarrow$ info carriers
- carrier collecting device (electrodes, photo-something)
- *Sensor* (t_c: carrier collection time)
- read-out circuit (RC: readout time constant)



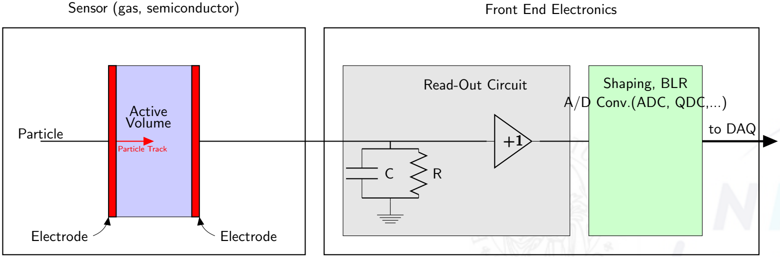
Detection cell: sensor and read-out circuit

- active volume
- energy absorbed $\Longrightarrow$ info carriers
- carrier collecting device (electrodes, photo-something)
- *Sensor* (t_c: carrier collection time)
- read-out circuit (RC: readout time constant)
- signal processing (e.g. filtering for SNR optimization)
- convert amplitude to digital



Detection cell: sensor and read-out circuit

- active volume
- energy absorbed $\Longrightarrow$ info carriers
- carrier collecting device (electrodes, photo-something)
- *Sensor* (t_c: carrier collection time)
- read-out circuit (RC: readout time constant)
- signal processing (e.g. filtering for SNR optimization)
- convert amplitude to digital
- read-out + signal processing = Front End Electronics (FEE)



Carrier collection and Read-Out characteristic times

- current pulse from sensor: total duration t_c (*collection time*)

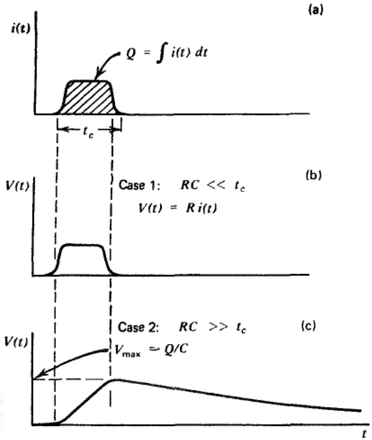


Figure 4.1 from Ref. [Knoll00].

Carrier collection and Read-Out characteristic times

- current pulse from sensor: total duration t_c (*collection time*)
- read-out signal shape: depends on read-out time constant RC

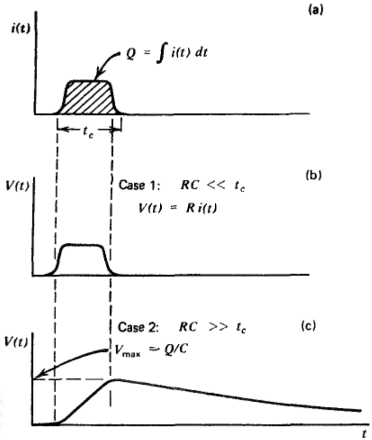


Figure 4.1 from Ref. [Knoll00].

Carrier collection and Read-Out characteristic times

- current pulse from sensor: total duration t_c (*collection time*)
- read-out signal shape: depends on read-out time constant RC
- $RC \ll t_c$: voltage across $R \| C$
  $V(t) \sim R\, i(t)$

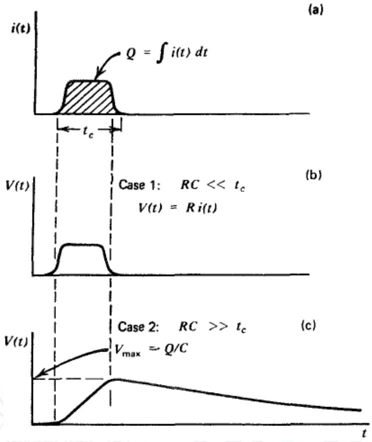


Figure 4.1 from Ref. [Knoll00].

Carrier collection and Read-Out characteristic times

- current pulse from sensor: total duration t_c (*collection time*)
- read-out signal shape: depends on read-out time constant RC
- $RC \ll t_c$: voltage across $R \| C$ $V(t) \sim R\, i(t)$
- $RC \gg t_c$: charge first integrated on C (max voltage $\sim Q/C$ reached after t_c), then C is discharged exponentially $V(t) \propto e^{-t/RC}$

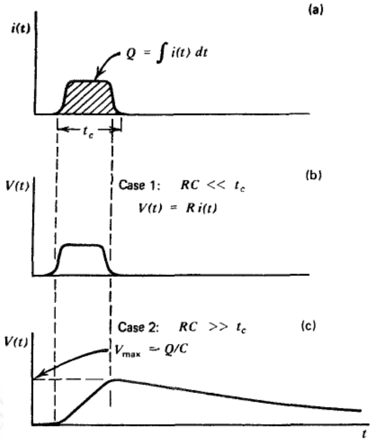


Figure 4.1 from Ref. [Knoll00].

Carrier collection and Read-Out characteristic times

- current pulse from sensor: total duration t_c (*collection time*)
- read-out signal shape: depends on read-out time constant RC
- $RC \ll t_c$: voltage across $R \| C$
  $V(t) \sim R\,i(t)$
- $RC \gg t_c$: charge first integrated on C (max voltage $\sim Q/C$ reached after t_c), then C is discharged exponentially
  $V(t) \propto e^{-t/RC}$
- t_c: few ns (fast scintillators, microchannel plates) $\longrightarrow$ 10-100 ns (semiconductors) $\longrightarrow$ μs (ionization chambers)

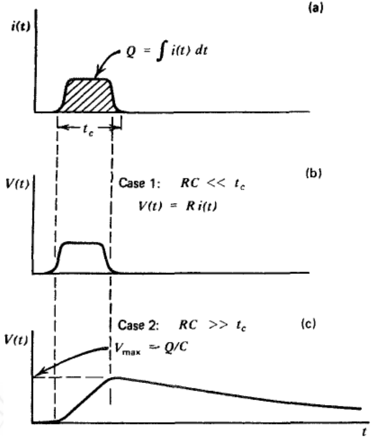
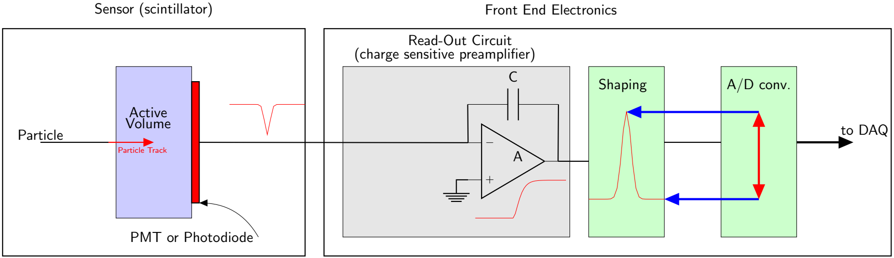
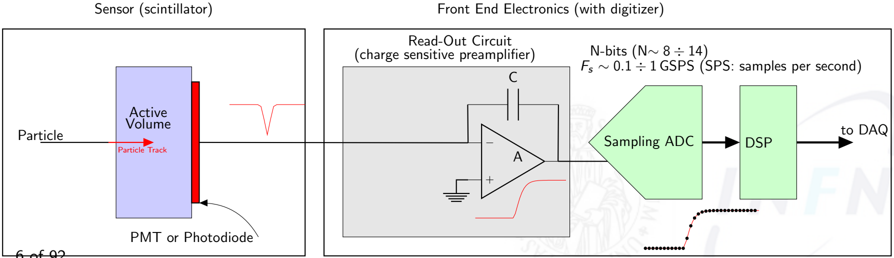


Figure 4.1 from Ref. [Knoll00].

Analog and digitizing chains compared





Advantages of digitizers

- better stability with respect to analog circuits
- flexibility (processing just a matter of calculation)
- easy pulse shape analysis implementation
- predictable and reduced dead time
- easy implementation of pile-up rejection
- processing not possible with analog can be implemented

How's information coded?

Deposited Energy:

- number of generated information carriers (ion pairs, e-h pairs, scintillation photons) $\Longrightarrow$ total produced charge Q;
- retrieved as $\int i(t)\mathrm{d}t$ ($RC \ll t_c \rightarrow$ current signal) or...
- maximum amplitude of $V(t)$ ($RC \gg t_c \rightarrow$ charge signal);

Time of interaction:

- time at which some signal feature (e.g. threshold crossing) occurs (*time mark*);
- unavoidable delay between interaction time and time mark;

How's information coded?

Point of gamma interaction (Hyperpure Ge detectors, e.g. AGATA):

- needed to reconstruct full gamma energy when multiple interaction occur before absorption;
- need segmented electrodes to identify a subvolume [Akkoyun12];
- within subvolume, interaction point obtained from all signal shapes (comparing with waveform database) with 1 mm resolution.

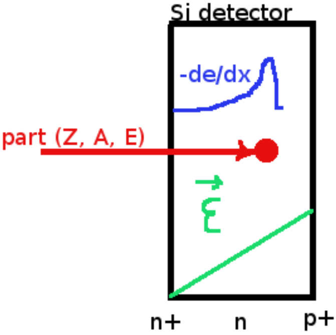
Radiation type (neutron? gamma? charged fragment?):

- usually coded into time evolution of signals;
- technique goes under PSA or PSD names (Pulse Shape Analysis, Pulse Shape Discrimination);
- often obtained from correlations (e.g. PSD param vs energy);
- collection time, relative amplitude fast/slow components.

From signal shape to radiation type: Si detector

Pulse Shape Discrimination example: nuclear fragment stopped in Si detector (FAZIA collaboration).

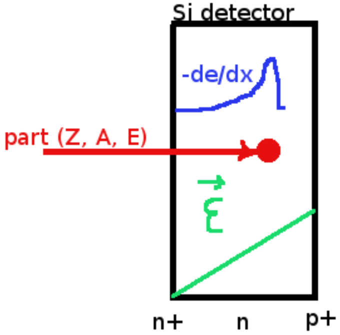
- collection time and maximum value of current depend on (Z, A) and E



From signal shape to radiation type: Si detector

Pulse Shape Discrimination example: nuclear fragment stopped in Si detector (FAZIA collaboration).

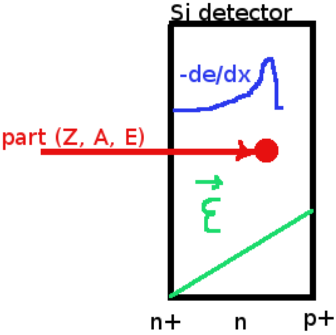
- collection time and maximum value of current depend on (Z, A) and E
- physical process involves:



From signal shape to radiation type: Si detector

Pulse Shape Discrimination example: nuclear fragment stopped in Si detector (FAZIA collaboration).

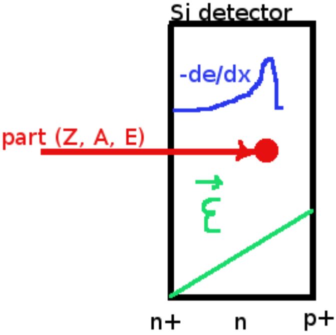
- collection time and maximum value of current depend on (Z, A) and E
- physical process involves:
  - ionization vs depth (Bragg curve);



From signal shape to radiation type: Si detector

Pulse Shape Discrimination example: nuclear fragment stopped in Si detector (FAZIA collaboration).

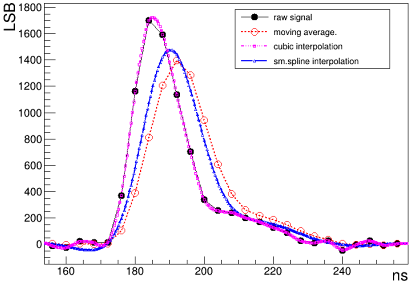
- collection time and maximum value of current depend on (Z, A) and E
- physical process involves:
  - ionization vs depth (Bragg curve);
  - electron-hole plasma erosion time;



From signal shape to radiation type: Si detector

Pulse Shape Discrimination example: nuclear fragment stopped in Si detector (FAZIA collaboration).

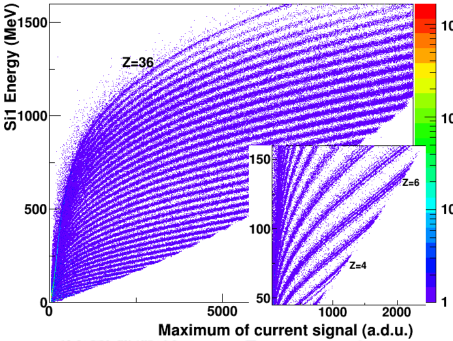
- collection time and maximum value of current depend on (Z, A) and E
- physical process involves:
  - ionization vs depth (Bragg curve);
  - electron-hole plasma erosion time;
- e.g., current signal max: study of noise reduction/interpolation



From signal shape to radiation type: Si detector

Pulse Shape Discrimination example: nuclear fragment stopped in Si detector (FAZIA collaboration).

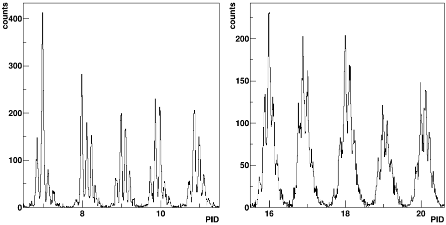
- collection time and maximum value of current depend on (Z, A) and E
- physical process involves:
  - ionization vs depth (Bragg curve);
  - electron-hole plasma erosion time;
- e.g., current signal max: study of noise reduction/interpolation
- figure: energy vs Imax recent data FAZIA collab. (G.Pastore et al., to be published)



From signal shape to radiation type: Si detector

Pulse Shape Discrimination example: nuclear fragment stopped in Si detector (FAZIA collaboration).

- collection time and maximum value of current depend on (Z, A) and E
- physical process involves:
  - ionization vs depth (Bragg curve);
  - electron-hole plasma erosion time;
- e.g., current signal max: study of noise reduction/interpolation
- figure: energy vs Imax recent data FAZIA collab. (G.Pastore et al., to be published)
- resulting isotopic identification (PID spectrum)



t-domain vs f-domain

- information coding is best understood in the time domain (t-domain);

t-domain vs f-domain

- information coding is best understood in the time domain (t-domain);
- amplitude, rise-time, shape, are all t-domain features;

t-domain vs f-domain

- information coding is best understood in the time domain (t-domain);
- amplitude, rise-time, shape, are all t-domain features;
- sometimes frequency domain (f-domain) useful (e.g. to understand processing with filters, noise behaviour, to exploit Fourier T., etc.);

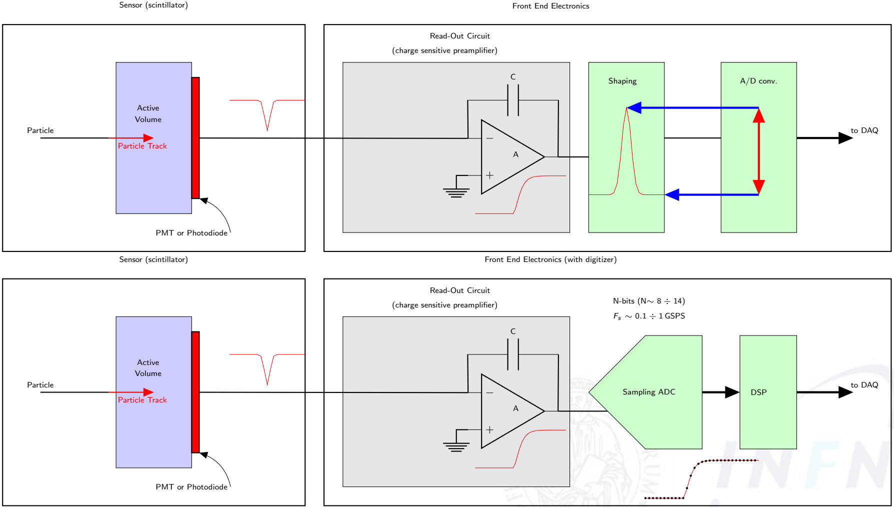
t-domain vs f-domain

- information coding is best understood in the time domain (t-domain);
- amplitude, rise-time, shape, are all t-domain features;
- sometimes frequency domain (f-domain) useful (e.g. to understand processing with filters, noise behaviour, to exploit Fourier T., etc.);
- we will mainly deal with t-domain issues, using f-domain when needed for better understanding.

Noise from the ADC

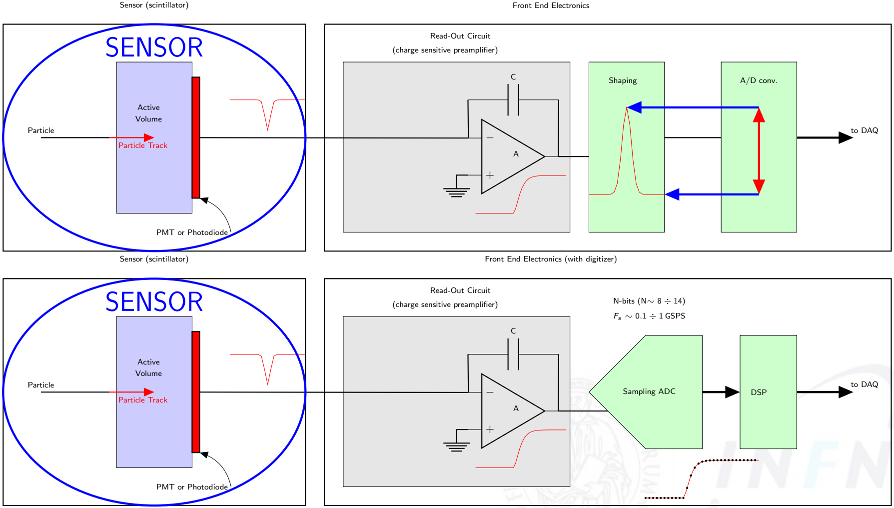
First Message: Sampling ADC as noise source

NOISE SOURCES EVIDENCED BY ELLIPSES!



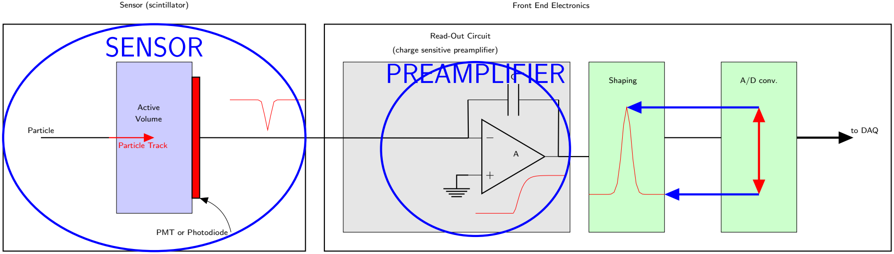
First Message: Sampling ADC as noise source

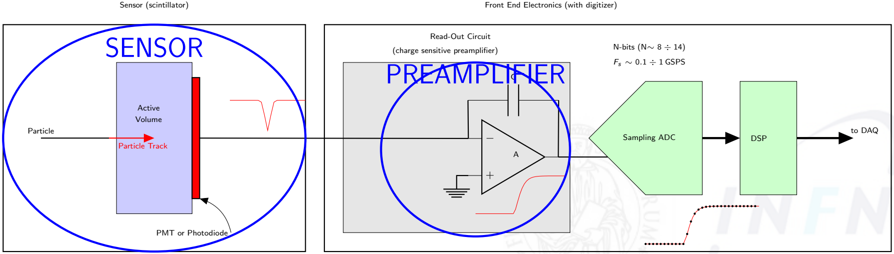
NOISE SOURCES EVIDENCED BY ELLIPSES!



First Message: Sampling ADC as noise source

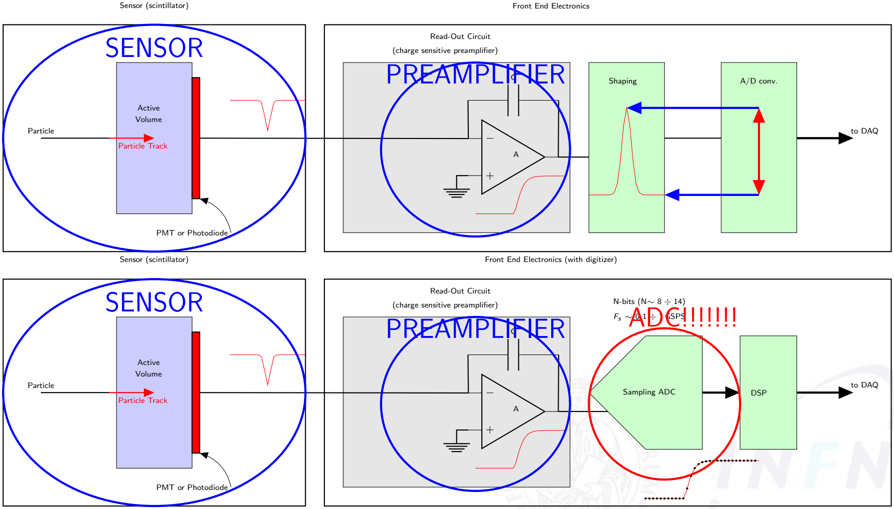
NOISE SOURCES EVIDENCED BY ELLIPSES!



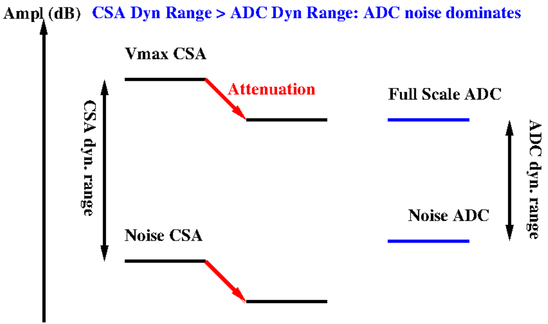


First Message: Sampling ADC as noise source

NOISE SOURCES EVIDENCED BY ELLIPSES!

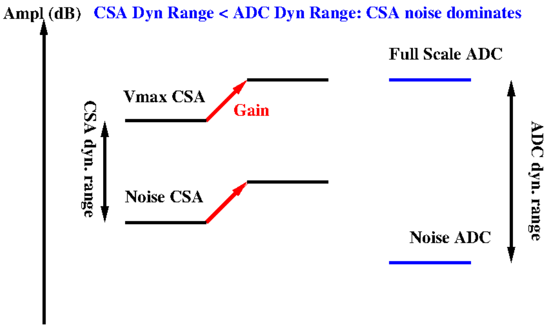


First Message: ADC noise and dynamic range



Take ADC noise into account for low noise energy measurements (e.g. X-ray detectors) or shape related measurements (e.g. for threshold crossing and signal maximum you can't average over many samples). Choose ADC based on the amount of noise already present in your input signal! For a noisy detector, low noise ADC not needed. Useful to compare Read-Out vs ADC dynamic range ($\approx \sqrt{12}\,2^{\mathrm{ENOB}}$):

First Message: ADC noise and dynamic range



Take ADC noise into account for low noise energy measurements (e.g. X-ray detectors) or shape related measurements (e.g. for threshold crossing and signal maximum you can't average over many samples). Choose ADC based on the amount of noise already present in your input signal! For a noisy detector, low noise ADC not needed. Useful to compare Read-Out vs ADC dynamic range ($\approx \sqrt{12}\, 2^{\mathrm{ENOB}}$):

Sampling ADC: performed operations

A sampling-ADC:

- measures amplitude at the analog input;

Sampling ADC: performed operations

A sampling-ADC:

- measures amplitude at the analog input;
- produces N-bits digital numbers (N called *resolution*);

Sampling ADC: performed operations

A sampling-ADC:

- measures amplitude at the analog input;
- produces N-bits digital numbers (N called *resolution*);
- 2^N possible output levels $\Longrightarrow$ some approximation needed

Sampling ADC: performed operations

A sampling-ADC:

- measures amplitude at the analog input;
- produces N-bits digital numbers (N called *resolution*);
- 2^N possible output levels $\Longrightarrow$ some approximation needed
- measurements most often separated by constant time interval (*sampling period*: T_s)

Sampling ADC: performed operations

A sampling-ADC:

- measures amplitude at the analog input;
- produces N-bits digital numbers (N called *resolution*);
- 2^N possible output levels $\Longrightarrow$ some approximation needed
- measurements most often separated by constant time interval (*sampling period*: T_s)
- $F_s = 1/T_s$ (in S/s, samples/second) called *sampling frequency*

Sampling ADC: performed operations

A sampling-ADC:

- measures amplitude at the analog input;
- produces N-bits digital numbers (N called *resolution*);
- 2^N possible output levels $\Longrightarrow$ some approximation needed
- measurements most often separated by constant time interval (*sampling period*: T_s)
- $F_s = 1/T_s$ (in S/s, samples/second) called *sampling frequency*
- in summary:

Sampling ADC: performed operations

A sampling-ADC:

- measures amplitude at the analog input;
- produces N-bits digital numbers (N called *resolution*);
- 2^N possible output levels $\Longrightarrow$ some approximation needed
- measurements most often separated by constant time interval (*sampling period*: T_s)
- $F_s = 1/T_s$ (in S/s, samples/second) called *sampling frequency*
- in summary:
  1. sampling: discretization of *independent* variable (e.g. time);

Sampling ADC: performed operations

A sampling-ADC:

- measures amplitude at the analog input;
- produces N-bits digital numbers (N called *resolution*);
- 2^N possible output levels $\Longrightarrow$ some approximation needed
- measurements most often separated by constant time interval (*sampling period*: T_s)
- $F_s = 1/T_s$ (in S/s, samples/second) called *sampling frequency*
- in summary:
  1. sampling: discretization of *independent* variable (e.g. time);
  2. quantization: discretization of *dependent* variable (e.g. amplitude);

ADC: math model of sampling

- from continuous (analog) signal $x_c(t)$ to sequence $x[n]$ $(n \in \mathbb{Z})$

$$x[n] = \mathcal{Q}\{x_c(n\,T_s)\}$$

where $T_s \equiv$ sampling period, $F_s = 1/T_s$ sampling frequency and $\mathcal{Q}$ is "amplitude quantization" operator

ADC: math model of sampling

- from continuous (analog) signal $x_c(t)$ to sequence $x[n]$ ($n \in \mathbb{Z}$)

$$x[n] \,=\, \mathcal{Q}\left\{x_c(n\,T_s)\right\}$$

  where $T_s \equiv$ sampling period, $F_s = 1/T_s$ sampling frequency and $\mathcal{Q}$ is "amplitude quantization" operator

- Can we recognize the two main tasks operated by an ADC?

ADC: math model of sampling

- from continuous (analog) signal $x_c(t)$ to sequence $x[n]$ ($n \in \mathbb{Z}$)

$$x[n] = \mathcal{Q}\{x_c(n\,T_s)\}$$

  where $T_s \equiv$ sampling period, $F_s = 1/T_s$ sampling frequency and $\mathcal{Q}$ is "amplitude quantization" operator
- Can we recognize the two main tasks operated by an ADC?
  1. sampling: discretization of independent var. $\rightarrow$ that's the $x_c(n\,T_s)$;

ADC: math model of sampling

- from continuous (analog) signal $x_c(t)$ to sequence $x[n]$ ($n \in \mathbb{Z}$)

$$x[n] = \mathcal{Q}\{x_c(n\,T_s)\}$$

  where $T_s \equiv$ sampling period, $F_s = 1/T_s$ sampling frequency and $\mathcal{Q}$ is "amplitude quantization" operator
- Can we recognize the two main tasks operated by an ADC?
  1. sampling: discretization of independent var. $\rightarrow$ that's the $x_c(n\,T_s)$;
  2. quantization: discretization of dependent var. $\rightarrow$ approx $x_c(n\,T_s)$ to the closest (or immediately below) admittable value $\mathcal{Q}\{x_c(n\,T_s)\}$

ADC: math model of sampling

- from continuous (analog) signal $x_c(t)$ to sequence $x[n]$ $(n \in \mathbb{Z})$

$$x[n] = \mathcal{Q}\{x_c(n\,T_s)\}$$

  where $T_s \equiv$ sampling period, $F_s = 1/T_s$ sampling frequency and $\mathcal{Q}$ is "amplitude quantization" operator
- Can we recognize the two main tasks operated by an ADC?
  1. sampling: discretization of independent var. $\rightarrow$ that's the $x_c(n\,T_s)$;
  2. quantization: discretization of dependent var. $\rightarrow$ approx $x_c(n\,T_s)$ to the closest (or immediately below) admittable value $\mathcal{Q}\{x_c(n\,T_s)\}$
- Widespread adopted convention:

ADC: math model of sampling

- from continuous (analog) signal $x_c(t)$ to sequence $x[n]$ $(n \in \mathbb{Z})$

$$x[n] = \mathcal{Q}\{x_c(n\,T_s)\}$$

where $T_s \equiv$ sampling period, $F_s = 1/T_s$ sampling frequency and $\mathcal{Q}$ is "amplitude quantization" operator

- Can we recognize the two main tasks operated by an ADC?
  1. sampling: discretization of independent var. $\rightarrow$ that's the $x_c(n\,T_s)$;
  2. quantization: discretization of dependent var. $\rightarrow$ approx $x_c(n\,T_s)$ to the closest (or immediately below) admittable value $\mathcal{Q}\{x_c(n\,T_s)\}$

- Widespread adopted convention:
  1. $x(t)$ continuous-time signal (lower case "x", parentheses)

ADC: math model of sampling

- from continuous (analog) signal $x_c(t)$ to sequence $x[n]$ ($n \in \mathbb{Z}$)

$$x[n] \;=\; \mathcal{Q}\left\{x_c(n\,T_s)\right\}$$

where $T_s \equiv$ sampling period, $F_s = 1/T_s$ sampling frequency and $\mathcal{Q}$ is "amplitude quantization" operator

- Can we recognize the two main tasks operated by an ADC?
  1. sampling: discretization of independent var. $\rightarrow$ that's the $x_c(n\,T_s)$;
  2. quantization: discretization of dependent var. $\rightarrow$ approx $x_c(n\,T_s)$ to the closest (or immediately below) admittable value $\mathcal{Q}\left\{x_c(n\,T_s)\right\}$

- Widespread adopted convention:
  1. $x(t)$ continuous-time signal (lower case "x", parentheses)
  2. $x[n]$ discrete-time signal (lower case "x", square brackets).

Quantization and noise...

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

- we start from point 2);

Quantization and noise...

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

- we start from point 2);
- N-bit ADC $\Longrightarrow 2^{\mathrm{N}}$ possible values $(0 \div 2^{\mathrm{N}} - 1)$;

Quantization and noise...

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

- we start from point 2);
- N-bit ADC $\Longrightarrow 2^{\mathrm{N}}$ possible values $(0 \div 2^{\mathrm{N}} - 1)$;
- quantized values $\neq$ "exact values":
  $e[n] = x_c(n\,T_s) - \mathcal{Q}\{x_c(n\,T_s)\} \neq 0$

Quantization and noise...

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

- we start from point 2);
- N-bit ADC $\Longrightarrow 2^{\mathrm{N}}$ possible values $(0 \div 2^{\mathrm{N}} - 1)$;
- quantized values $\neq$ "exact values":
  $e[n] = x_c(n\,T_s) - \mathcal{Q}\left\{x_c(n\,T_s)\right\} \neq 0$
- $e[n]$ (quantization error) varies from sample to sample;

Quantization and noise...

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

- we start from point 2);
- N-bit ADC $\Longrightarrow 2^{\mathrm{N}}$ possible values $(0 \div 2^{\mathrm{N}} - 1)$;
- quantized values $\neq$ "exact values":
  $e[n] = x_c(n\,T_s) - \mathcal{Q}\left\{x_c(n\,T_s)\right\} \neq 0$
- $e[n]$ (quantization error) varies from sample to sample;
- quantized levels "close enough" $+$ complex signal (e.g. speech) $\Longrightarrow$ difference fluctuates randomly from sample to sample [Oppenheim2010];

Quantization and noise...

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

- we start from point 2);
- N-bit ADC $\Longrightarrow 2^{\mathrm{N}}$ possible values $(0 \div 2^{\mathrm{N}} - 1)$;
- quantized values $\neq$ "exact values":
  $e[n] = x_c(n\,T_s) - \mathcal{Q}\left\{x_c(n\,T_s)\right\} \neq 0$
- $e[n]$ (quantization error) varies from sample to sample;
- quantized levels "close enough" + complex signal (e.g. speech) $\Longrightarrow$ difference fluctuates randomly from sample to sample [Oppenheim2010];
- also true for simple signals + large BW noise (detector pulse!);

Quantization and noise...

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

- we start from point 2);
- N-bit ADC $\Longrightarrow 2^{\mathrm{N}}$ possible values $(0 \div 2^{\mathrm{N}} - 1)$;
- quantized values $\neq$ "exact values":
  $e[n] = x_c(n\,T_s) - \mathcal{Q}\left\{x_c(n\,T_s)\right\} \neq 0$
- $e[n]$ (quantization error) varies from sample to sample;
- quantized levels "close enough" + complex signal (e.g. speech) $\Longrightarrow$ difference fluctuates randomly from sample to sample [Oppenheim2010];
- also true for simple signals + large BW noise (detector pulse!);
- quant. noise model: "white" noise of variance $\sigma_Q^2 = \dfrac{1}{12}\left(\dfrac{R}{2^{\mathrm{N}}}\right)^2$

Quantization and noise...

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

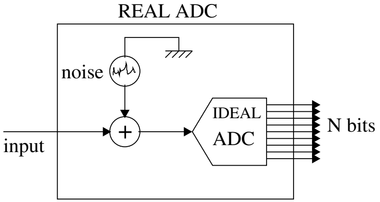
- we start from point 2);
- N-bit ADC $\Longrightarrow 2^{\mathrm{N}}$ possible values ($0 \div 2^{\mathrm{N}} - 1$);
- quantized values $\neq$ "exact values":
  $e[n] = x_c(n\,T_s) - \mathcal{Q}\{x_c(n\,T_s)\} \neq 0$
- $e[n]$ (quantization error) varies from sample to sample;
- quantized levels "close enough" + complex signal (e.g. speech) $\Longrightarrow$ difference fluctuates randomly from sample to sample [Oppenheim2010];
- also true for simple signals + large BW noise (detector pulse!);
- quant. noise model: "white" noise of variance $\sigma_Q^2 = \frac{1}{12}\left(\frac{R}{2^{\mathrm{N}}}\right)^2$
- R is ADC range in Volt (take $R = 2^{\mathrm{N}}$ to get the equivalent in bits);

Real ADC and noise...

- thermal and shot noise in internal ADC circuit $\Longrightarrow$ a real ADC will add much more than quantization noise

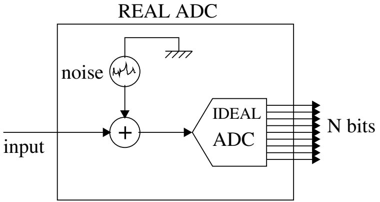
Real ADC and noise...

- thermal and shot noise in internal ADC circuit $\Longrightarrow$ a real ADC will add much more than quantization noise
- a real ADC can be modelled as an "ideal" ADC plus a noise generator adding noise to the input (see figure);



Real ADC and noise...

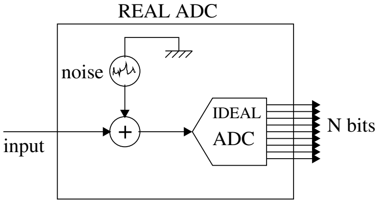
- thermal and shot noise in internal ADC circuit $\Longrightarrow$ a real ADC will add much more than quantization noise
- a real ADC can be modelled as an "ideal" ADC plus a noise generator adding noise to the input (see figure);



- we can include quantization noise into the generator and assume no need for quantization in the "ideal" ADC;

Real ADC and noise...

- thermal and shot noise in internal ADC circuit $\Longrightarrow$ a real ADC will add much more than quantization noise
- a real ADC can be modelled as an "ideal" ADC plus a noise generator adding noise to the input (see figure);



- we can include quantization noise into the generator and assume no need for quantization in the "ideal" ADC;
- real ADC noise has variance $\sigma^2_{eff} > \sigma^2_Q$

ENOB and noise (1)

- quantization noise, "white" noise, variance $\sigma_Q^2 = \frac{1}{12}\left(\frac{R}{2^{\mathrm{N}}}\right)^2$;

ENOB and noise (1)

- quantization noise, "white" noise, variance $\sigma_Q^2 = \frac{1}{12}\left(\frac{R}{2^{\mathrm{N}}}\right)^2$;
- "real" ADC: added noise has variance $\sigma_{eff}^2 > \sigma_Q^2$

ENOB and noise (1)

- quantization noise, "white" noise, variance $\sigma_Q^2 = \frac{1}{12}\left(\frac{R}{2^{\mathrm{N}}}\right)^2$;
- "real" ADC: added noise has variance $\sigma_{eff}^2 > \sigma_Q^2$
- manufacturers quote effective-number-of-bits (ENOB)

ENOB and noise (1)

- quantization noise, “white” noise, variance $\sigma_Q^2 = \frac{1}{12}\left(\frac{R}{2^{\mathrm{N}}}\right)^2$;
- “real” ADC: added noise has variance $\sigma_{eff}^2 > \sigma_Q^2$
- manufacturers quote effective-number-of-bits (ENOB)
- ENOB is the number you need instead of N in $\sigma_Q^2 = \frac{1}{12}\left(\frac{R}{2^{\mathrm{N}}}\right)^2$

  to get σ_{eff}^2, i.e. $\sigma_{eff}^2 = \frac{1}{12}\left(\frac{R}{2^{\mathrm{ENOB}}}\right)^2$

$$\mathrm{ENOB} = log_2\left(\frac{R}{\sqrt{12}\,\sigma_{eff}}\right)$$

ENOB and noise (1)

- quantization noise, "white" noise, variance $\sigma_Q^2 = \frac{1}{12}\left(\frac{R}{2^{\mathrm{N}}}\right)^2$;
- "real" ADC: added noise has variance $\sigma_{eff}^2 > \sigma_Q^2$
- manufacturers quote effective-number-of-bits (ENOB)
- ENOB is the number you need instead of N in $\sigma_Q^2 = \frac{1}{12}\left(\frac{R}{2^{\mathrm{N}}}\right)^2$

  to get σ_{eff}^2, i.e. $\sigma_{eff}^2 = \frac{1}{12}\left(\frac{R}{2^{\mathrm{ENOB}}}\right)^2$

$$\mathrm{ENOB} = log_2\left(\frac{R}{\sqrt{12}\,\sigma_{eff}}\right)$$

- The "textbook" ENOB definition is actually $\mathrm{ENOB} = \frac{SNR(dB) - 1.76}{6.02}$ where *SNR* is the signal-to-noise ratio including effects of harmonic distortion, clock jitter etc. My definition is equivalent to this one provided that σ_{eff} is the dominant contribution to *SNR*, which is usually the case in nuclear physics.

ENOB and noise (2)

- dynamic range: 1 bit lost each doubling of σ_{eff} ($R/\sigma_{eff} \propto 2^{\mathrm{ENOB}}$);

ENOB and noise (2)

- dynamic range: 1 bit lost each doubling of σ_{eff} ($R/\sigma_{eff} \propto 2^{\mathrm{ENOB}}$);
- in bits ($R = 2^{\mathrm{N}}$) we get $\sigma_{eff} \propto 2^{\mathrm{N}-\mathrm{ENOB}} \Longrightarrow \mathrm{N} - \mathrm{ENOB}$ controls how much noise added by ADC;

ENOB and noise (2)

- dynamic range: 1 bit lost each doubling of σ_{eff} ($R/\sigma_{eff} \propto 2^{\mathrm{ENOB}}$);
- in bits ($R = 2^{\mathrm{N}}$) we get $\sigma_{eff} \propto 2^{\mathrm{N}-\mathrm{ENOB}} \Longrightarrow \mathrm{N} - \mathrm{ENOB}$ controls how much noise added by ADC;
- greater $\mathrm{N} - \mathrm{ENOB} \Longrightarrow$ more ADC noise;

ENOB and noise (2)

- dynamic range: 1 bit lost each doubling of σ_{eff} ($R/\sigma_{eff} \propto 2^{\mathrm{ENOB}}$);
- in bits ($R = 2^{\mathrm{N}}$) we get $\sigma_{eff} \propto 2^{\mathrm{N}-\mathrm{ENOB}} \implies \mathrm{N} - \mathrm{ENOB}$ controls how much noise added by ADC;
- greater $\mathrm{N} - \mathrm{ENOB} \implies$ more ADC noise;
- typical value $\mathrm{N} - \mathrm{ENOB} = 1 \div 2$

ENOB and noise (2)

- dynamic range: 1 bit lost each doubling of σ_{eff} ($R/\sigma_{eff} \propto 2^{\mathrm{ENOB}}$);
- in bits ($R = 2^{\mathrm{N}}$) we get $\sigma_{eff} \propto 2^{\mathrm{N}-\mathrm{ENOB}} \Longrightarrow \mathrm{N} - \mathrm{ENOB}$ controls how much noise added by ADC;
- greater $\mathrm{N} - \mathrm{ENOB} \Longrightarrow$ more ADC noise;
- typical value $\mathrm{N} - \mathrm{ENOB} = 1 \div 2$
- comparing ADC noise to noise before ADC: express R in Volts (e.g. $R = 2\mathrm{V}$, N$= 14$, ENOB$= 12 \Longrightarrow \sigma_{eff} = 1.15\,\mathrm{LSB} = 140\,\mu\mathrm{V}$ where 1 LSB$= R/2^{\mathrm{N}}$).

ENOB and noise (2)

- dynamic range: 1 bit lost each doubling of σ_{eff} ($R/\sigma_{eff} \propto 2^{\mathrm{ENOB}}$);
- in bits ($R = 2^{\mathrm{N}}$) we get $\sigma_{eff} \propto 2^{\mathrm{N-ENOB}} \Longrightarrow \mathrm{N} - \mathrm{ENOB}$ controls how much noise added by ADC;
- greater $\mathrm{N} - \mathrm{ENOB} \Longrightarrow$ more ADC noise;
- typical value $\mathrm{N} - \mathrm{ENOB} = 1 \div 2$
- comparing ADC noise to noise before ADC: express R in Volts (e.g. $R = 2$V, N$= 14$, ENOB$= 12 \Longrightarrow \sigma_{eff} = 1.15\,\mathrm{LSB} = 140\,\mu$V where 1 LSB$= R/2^{\mathrm{N}}$).
- *ENOB* and *Effective Resolution* (a DC spec) not the same (distortion and quantization noise not included). However...

ENOB and noise (2)

- dynamic range: 1 bit lost each doubling of σ_{eff} ($R/\sigma_{eff} \propto 2^{\mathrm{ENOB}}$);
- in bits ($R = 2^{\mathrm{N}}$) we get $\sigma_{eff} \propto 2^{\mathrm{N}-\mathrm{ENOB}} \Longrightarrow \mathrm{N} - \mathrm{ENOB}$ controls how much noise added by ADC;
- greater $\mathrm{N} - \mathrm{ENOB} \Longrightarrow$ more ADC noise;
- typical value $\mathrm{N} - \mathrm{ENOB} = 1 \div 2$
- comparing ADC noise to noise before ADC: express R in Volts (e.g. $R = 2$V, N$= 14$, ENOB$= 12 \Longrightarrow \sigma_{eff} = 1.15\,\mathrm{LSB} = 140\,\mu$V where 1 LSB$= R/2^{\mathrm{N}}$).
- *ENOB* and *Effective Resolution* (a DC spec) not the same (distortion and quantization noise not included). However...
- **in practice** input ref. noise accounts for most SINAD, i.e. determines most of $N - ENOB$

ENOB and noise (2)

- dynamic range: 1 bit lost each doubling of σ_{eff} ($R/\sigma_{eff} \propto 2^{\mathrm{ENOB}}$);
- in bits ($R = 2^{\mathrm{N}}$) we get $\sigma_{eff} \propto 2^{\mathrm{N}-\mathrm{ENOB}} \Longrightarrow \mathrm{N} - \mathrm{ENOB}$ controls how much noise added by ADC;
- greater $\mathrm{N} - \mathrm{ENOB} \Longrightarrow$ more ADC noise;
- typical value $\mathrm{N} - \mathrm{ENOB} = 1 \div 2$
- comparing ADC noise to noise before ADC: express R in Volts (e.g. $R = 2$V, N$= 14$, ENOB$= 12 \Longrightarrow \sigma_{eff} = 1.15\,\mathrm{LSB} = 140\,\mu$V where 1 LSB$= R/2^{\mathrm{N}}$).
- *ENOB* and *Effective Resolution* (a DC spec) not the same (distortion and quantization noise not included). However...
- **in practice** input ref. noise accounts for most SINAD, i.e. determines most of $N - ENOB$
- actual effective resolution and *ENOB* usually differ by a few 0.1 LBS

Shannon, signal reconstruction, interpolation

ADC: questions about sampling

- Is some info lost in the process?

ADC: questions about sampling

- Is some info lost in the process?
- If info is lost, when and how?

ADC: questions about sampling

- Is some info lost in the process?
- If info is lost, when and how?
- Under which conditions is the loss acceptable?

ADC: questions about sampling

- Is some info lost in the process?
- If info is lost, when and how?
- Under which conditions is the loss acceptable?
- no final answer (ask questions each time);

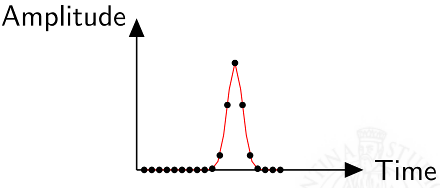
ADC: questions about sampling

- Is some info lost in the process?
- If info is lost, when and how?
- Under which conditions is the loss acceptable?
- no final answer (ask questions each time);
- useful to grasp the general aspect of the problem.

Discrete-Time and “proper” sampling

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

Let’s come to point 1), time-discretization: $x[n] = x_c(n\,T_s)$
Intuition $\Longrightarrow$ maximum acceptable sampling period must exists:
Starting from “close” samples...

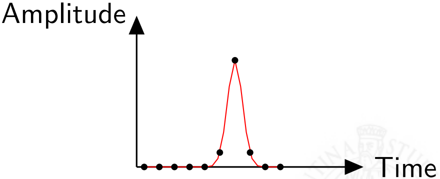


...we can still recognize signal shape from samples...

Discrete-Time and “proper” sampling

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

Let’s come to point 1), time-discretization: $x[n] = x_c(n\,T_s)$
Intuition $\Longrightarrow$ maximum acceptable sampling period must exists:
...we increase the sampling period T_s...

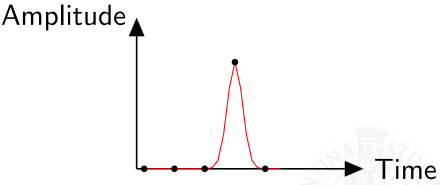


...now just a few samples taken...

Discrete-Time and "proper" sampling

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

Let's come to point 1), time-discretization: $x[n] = x_c(n\,T_s)$
Intuition $\Longrightarrow$ maximum acceptable sampling period must exists:
...still more "space" between samples...

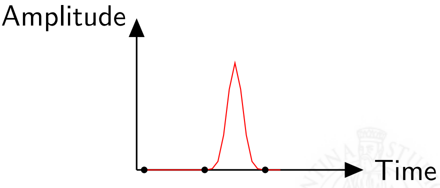


...so that we take just one sample (by chance) during the signal...

Discrete-Time and "proper" sampling

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

Let's come to point 1), time-discretization: $x[n] = x_c(n\,T_s)$
Intuition $\Longrightarrow$ maximum acceptable sampling period must exists:
...and when $T_s \geq$ signal duration...



... signal not even recognized as such!

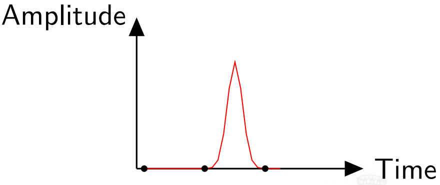
Discrete-Time and "proper" sampling

1. sampling: discretization of *independent* variable;
2. quantization: discretization of *dependent* variable;

Let's come to point 1), time-discretization: $x[n] = x_c(n\,T_s)$

Intuition $\Longrightarrow$ maximum acceptable sampling period must exists:

...and when $T_s \geq$ signal duration...



... signal not even recognized as such!

Need for recipe: what is maximum T_s for which I can get back the original (continuous) signal? How to get it back from samples?

Why signal reconstruction? E.g. timing!

- Timing: extracting a “time mark” from a signal, e.g. with a leading edge discriminator (LED);

Why signal reconstruction? E.g. timing!

- Timing: extracting a "time mark" from a signal, e.g. with a leading edge discriminator (LED);
- LED: device emitting a logic "true" signal when input voltage crosses a fixed threshold (e.g. oscilloscope trigger)

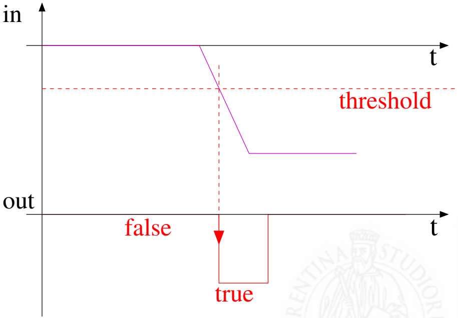


Figure 1: Leading edge discriminator

Timing with digitized signals

- A LED “fires” when signal assumes a given value: e.g. LED looks for t_{x}: $s(t_{\mathrm{x}}) = V_{\mathrm{threshold}}$

Timing with digitized signals

- A LED “fires” when signal assumes a given value: e.g. LED looks for t_x: $s(t_\mathrm{x}) = V_\mathrm{threshold}$
- what about timing with a digitized signal $s[n]$ where $s[n] = s(n\,T_s)$?

Timing with digitized signals

- A LED "fires" when signal assumes a given value: e.g. LED looks for t_{x}: $s(t_{\mathrm{x}}) = V_{\mathrm{threshold}}$
- what about timing with a digitized signal $s[n]$ where $s[n] = s(n\,T_s)$?
- "digital timing" still means: look for t_x: $s(t_x) = V_{\mathrm{threshold}}$

Timing with digitized signals

- A LED "fires" when signal assumes a given value: e.g. LED looks for t_{x}: $s(t_{\mathrm{x}}) = V_{\mathrm{threshold}}$
- what about timing with a digitized signal $s[n]$ where $s[n] = s(n\,T_s)$?
- "digital timing" still means: look for t_x: $s(t_x) = V_{\mathrm{threshold}}$
- approximating the crossing time to the closest sample could be too rough in many applications (surely if we want sub-nanosecond resolution!)

Timing with digitized signals

- A LED "fires" when signal assumes a given value: e.g. LED looks for t_{x}: $s(t_{\mathrm{x}}) = V_{\mathrm{threshold}}$
- what about timing with a digitized signal $s[n]$ where $s[n] = s(n\,T_s)$?
- "digital timing" still means: look for t_x: $s(t_x) = V_{\mathrm{threshold}}$
- approximating the crossing time to the closest sample could be too rough in many applications (surely if we want sub-nanosecond resolution!)
- we would like to do better!

Timing with digitized signals

- A LED "fires" when signal assumes a given value: e.g. LED looks for t_{x}: $s(t_{\mathrm{x}}) = V_{\mathrm{threshold}}$
- what about timing with a digitized signal $s[n]$ where $s[n] = s(n\,T_s)$?
- "digital timing" still means: look for t_x: $s(t_x) = V_{\mathrm{threshold}}$
- approximating the crossing time to the closest sample could be too rough in many applications (surely if we want sub-nanosecond resolution!)
- we would like to do better!
- the event $s[n] = s(n\,T_s) = V_{\mathrm{threshold}}$ is **very** unlikely;

Timing with digitized signals

- A LED "fires" when signal assumes a given value: e.g. LED looks for t_{x}: $s(t_{\mathrm{x}}) = V_{\mathrm{threshold}}$
- what about timing with a digitized signal $s[n]$ where $s[n] = s(n\,T_s)$?
- "digital timing" still means: look for t_x: $s(t_x) = V_{\mathrm{threshold}}$
- approximating the crossing time to the closest sample could be too rough in many applications (surely if we want sub-nanosecond resolution!)
- we would like to do better!
- the event $s[n] = s(n\,T_s) = V_{\mathrm{threshold}}$ is **very** unlikely;
- threshold crossing usually happens "in between samples";

Timing with digitized signals

- A LED "fires" when signal assumes a given value: e.g. LED looks for t_{x}: $s(t_{\mathrm{x}}) = V_{\mathrm{threshold}}$
- what about timing with a digitized signal $s[n]$ where $s[n] = s(n\,T_s)$?
- "digital timing" still means: look for t_x: $s(t_x) = V_{\mathrm{threshold}}$
- approximating the crossing time to the closest sample could be too rough in many applications (surely if we want sub-nanosecond resolution!)
- we would like to do better!
- the event $s[n] = s(n\,T_s) = V_{\mathrm{threshold}}$ is **very** unlikely;
- threshold crossing usually happens "in between samples";
- knowing the signal "in between samples" we could get the exact crossing time;

A possible recipe: Shannon Theorem

Shannon Theorem

- in order to obtain "proper" sampling (i.e. being able to reconstruct signal $x_c(t)$ from equally spaced samples $x[n]$ exactly)

1. continuous signal must be bandlimited $\Longrightarrow$ no components in f-domain (Fourier transform) with $f > F_{max}$ (Nyquist freq.)
2. samples must be spaced by $T_s \leq \frac{1}{2} T_{max}$ with $T_{max} = \frac{1}{F_{max}}$, i.e. sampling frequency $F_s \geq 2\, F_{max}$

- recipe to get back $x_c(t)$:

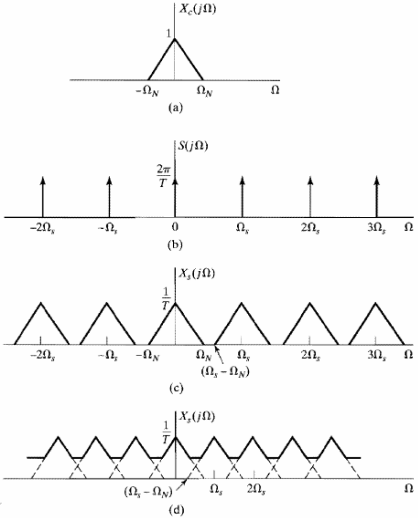
$$x_c(t) = \sum_{n=-\infty}^{+\infty} x[n] \cdot \mathrm{sinc}\left(\frac{t}{T_s} - n\right) \tag{1}$$

where $\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$ (sinc: *cardinal sine* function).

Shannon: periodic frequency spectrum

$$X_s(j\Omega) = \frac{1}{T_s} \sum_{k=-\infty}^{+\infty} X_c(\Omega - k\Omega_s)$$

original $X_c(j\Omega)$ plus ∞ copies shifted by $k\Omega_s$

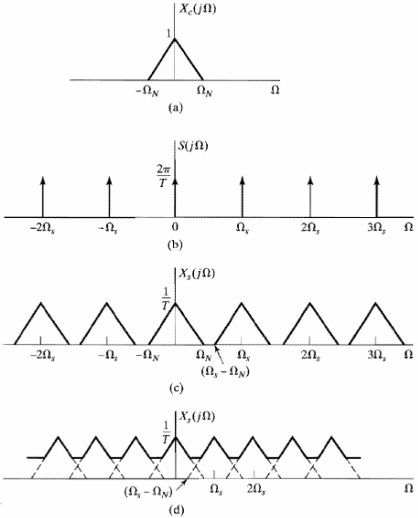


Shannon: periodic frequency spectrum

$$X_s(j\Omega) = \frac{1}{T_s} \sum_{k=-\infty}^{+\infty} X_c(\Omega - k\Omega_s)$$

original $X_c(j\Omega)$ plus ∞ copies shifted by $k\Omega_s$

- To re-construct the original $\mathcal{FT}$: use frequency-selective filter keeping the original and discarding the copies

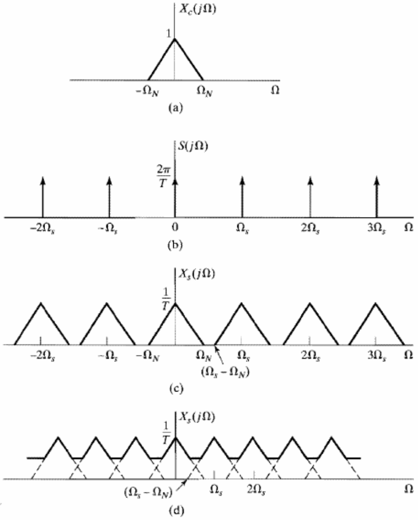


Shannon: periodic frequency spectrum

$$X_s(j\Omega) = \frac{1}{T_s} \sum_{k=-\infty}^{+\infty} X_c(\Omega - k\Omega_s)$$

original $X_c(j\Omega)$ plus ∞ copies shifted by $k\Omega_s$

- To re-construct the original $\mathcal{FT}$: use frequency-selective filter keeping the original and discarding the copies
- use inverse $\mathcal{FT}$ to obtain $x_c(t)$



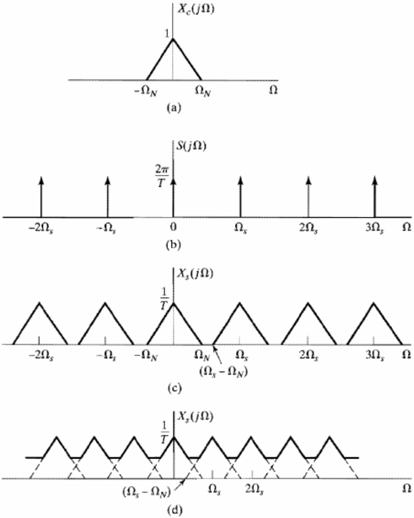
Shannon: periodic frequency spectrum

$$X_s(j\Omega) = \frac{1}{T_s} \sum_{k=-\infty}^{+\infty} X_c(\Omega - k\Omega_s)$$

original $X_c(j\Omega)$ plus ∞ copies shifted by $k\Omega_s$

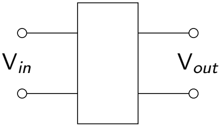
- To re-construct the original $\mathcal{FT}$: use frequency-selective filter keeping the original and discarding the copies
- use inverse $\mathcal{FT}$ to obtain $x_c(t)$
- copies must NOT overlap $\Longrightarrow$ if Ω_N is maximum frequency in $x_c(t)$ then we want

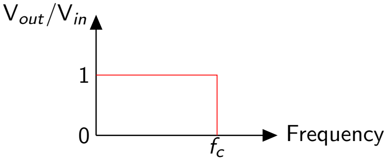
$$\Omega_s - \Omega_N \geq \Omega_N \Longrightarrow \Omega_s \geq 2\Omega_N$$



Shannon Theorem: the sinc function

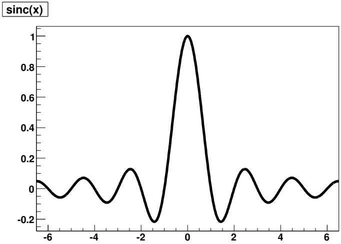
The sinc is the inverse FT of (the response of) the perfect reconstruction filter: the brick wall filter:



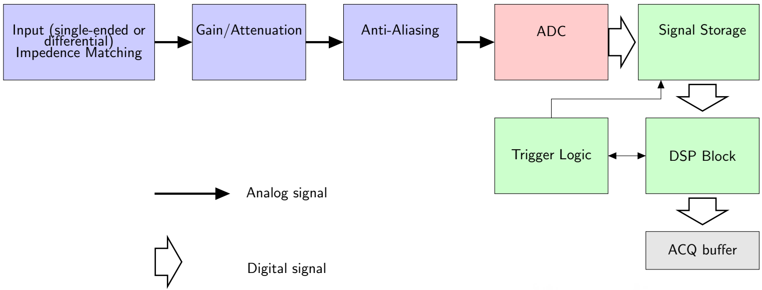


The sinc function:

1. extends to $\pm\infty$;
2. it is $= 1$ in $x = 0$;
3. it is $= 0$ for integer x.



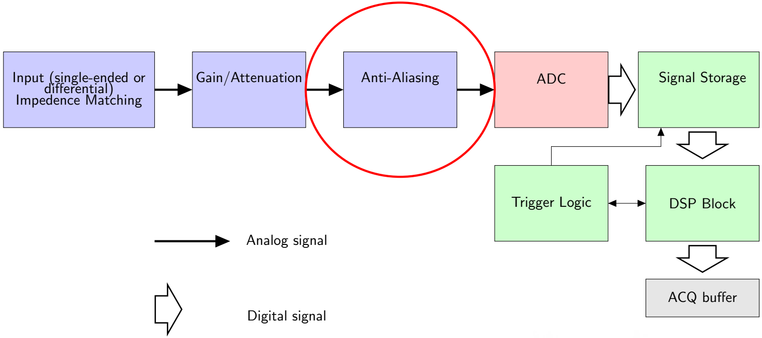
Block diagram of a digitizer



- input stage: can be single ended or differential, matches transmission line characteristic impedance
- gain: adjust overall dynamic range, variable gain to change max non-saturating amplitude (ADC range is fixed)
- antialias: LPF needed to cope with Sampling Theorem, more about it later
- ADC: produces one binary value every T_s (sampling period, $F_s = 1/T_s$ sampling frequency)
- Storage: here we keep digital signal until we have processed it as needed
- DSP and ACQ buffer: signal elaboration, information extraction (on-board μprocessor or FPGA)
- Trigger Logic: often inside the same FPGA used for storage and elaboration

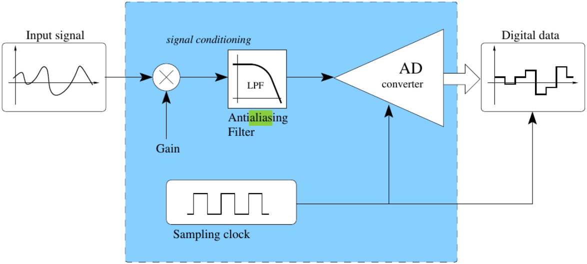
Block diagram of a digitizer

SOME WORDS ABOUT THE ANTIALIASING FILTER...



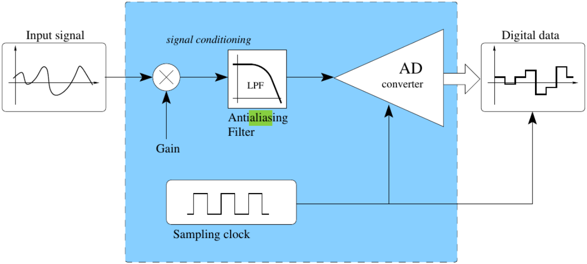
- input stage: can be single ended or differential, matches transmission line characteristic impedance
- gain: adjust overall dynamic range, variable gain to change max non-saturating amplitude (ADC range is fixed)
- antialias: LPF needed to cope with Sampling Theorem, more about it later
- ADC: produces one binary value every T_s (sampling period, $F_s = 1/T_s$ sampling frequency)
- Storage: here we keep digital signal until we have processed it as needed
- DSP and ACQ buffer: signal elaboration, information extraction (on-board μprocessor or FPGA)
- Trigger Logic: often inside the same FPGA used for storage and elaboration

Antialiasing filter



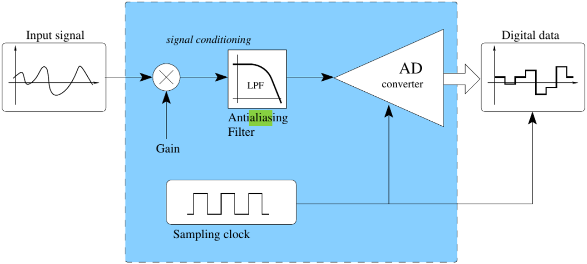
- low pass filter: to limit the bandwidth before sampling;

Antialiasing filter



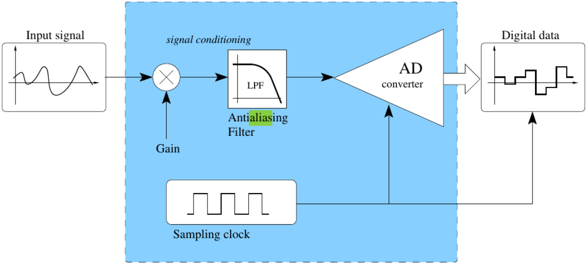
- low pass filter: to limit the bandwidth before sampling;
- a component at $F_s/2 + \Delta f$ appears at $F_s/2 - \Delta f$ after sampling (*aliasing*);

Antialiasing filter



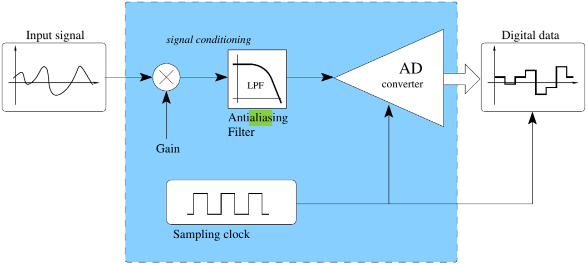
- low pass filter: to limit the bandwidth before sampling;
- a component at $F_s/2 + \Delta f$ appears at $F_s/2 - \Delta f$ after sampling (*aliasing*);
- importance of good t-domain response (e.g. step response);

Antialiasing filter



- low pass filter: to limit the bandwidth before sampling;
- a component at $F_s/2 + \Delta f$ appears at $F_s/2 - \Delta f$ after sampling (*aliasing*);
- importance of good t-domain response (e.g. step response);
- a real antialias filter can't eliminate $f > F_s/2$ completely;

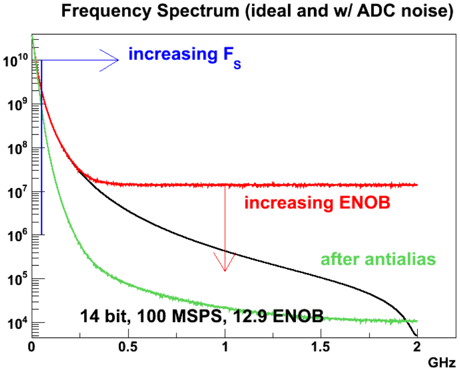
Antialiasing filter



- low pass filter: to limit the bandwidth before sampling;
- a component at $F_s/2 + \Delta f$ appears at $F_s/2 - \Delta f$ after sampling (*aliasing*);
- importance of good t-domain response (e.g. step response);
- a real antialias filter can't eliminate $f > F_s/2$ completely;
- aliasing will worsen the *SNR* of the digitizer (must be always compared to the *SNR* of the original signal);

Pulse f-spectrum, ideal, w/ noise, w/ anti-alias

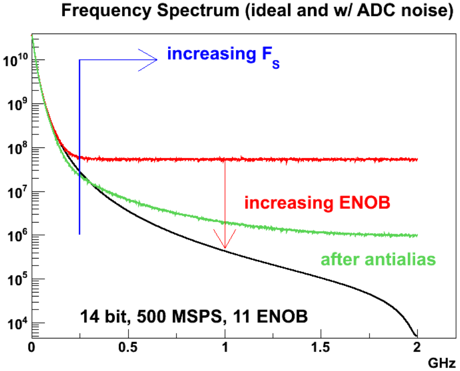
Example: Analog f-spectrum (i.e. no mirror images) of PMT pulse lasting about 50 ns (black curve). Detector signals have vanishing frequency content at high freq. (easier job for antialias filter).



In red: ADC noise added. In green: with antialias filter.

Pulse f-spectrum, ideal, w/ noise, w/ anti-alias

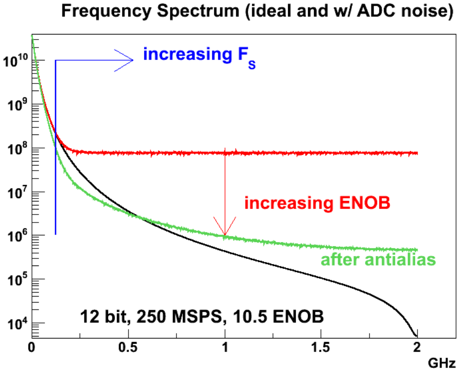
Example: Analog f-spectrum (i.e. no mirror images) of PMT pulse lasting about 50 ns (black curve). Detector signals have vanishing frequency content at high freq. (easier job for antialias filter).



Only 11 ENOB (though N=14 bit resol): noise > signal near Nyquist.

Pulse f-spectrum, ideal, w/ noise, w/ anti-alias

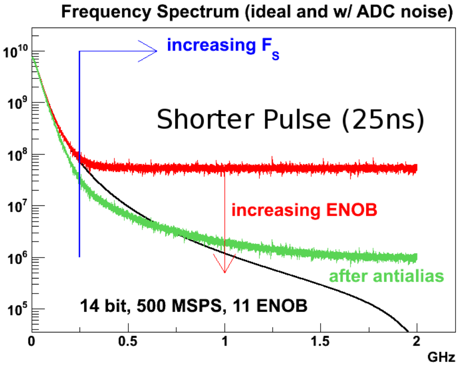
Example: Analog f-spectrum (i.e. no mirror images) of PMT pulse lasting about 50 ns (black curve). Detector signals have vanishing frequency content at high freq. (easier job for antialias filter).



Only 12 bit resol. However 10.5 ENOB $\Longrightarrow$ reasonable SNR.

Pulse f-spectrum, ideal, w/ noise, w/ anti-alias

Example: Analog f-spectrum (i.e. no mirror images) of PMT pulse lasting about 50 ns (black curve). Detector signals have vanishing frequency content at high freq. (easier job for antialias filter).



Shorter pulse! More high frequency content. Now 500 MSPS probably better!

Shannon Theorem: additional remarks/caveats

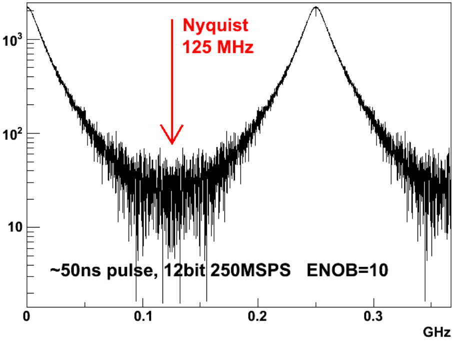
- equation (1) is a particular instance of *interpolation formula*

$$f(t) = \sum_{n=-\infty}^{+\infty} c_n \cdot g\left(\frac{t}{T_s} - n\right) \tag{2}$$

  where $g(t)$ is called *interpolation kernel*;
- "interpolation is the process of estimating the intermediate values of a continuous event from discrete samples" [Keys1981].
- usually $c_n = x[n]$, i.e. eq.(2) is a discrete convolution $(x * g)$
- you want to get back the original signal...
- ...in real life you never sample the *original* signal (artifacts added by input circuit, antialias filter, ADC noise,...)
- "original" signal already has fluctuations/noise (e.g. statistical fluctuations in carrier production, read-out circuit noise,...)
- interpolation-induced deviations of the same order of the ones already present would not be much of a problem...

Interpolation and frequency domain

In discrete-time, as in continuous time, t-domain convolution of signals $\equiv$ product of their Fourier Transforms. Convolution with sinc removes all frequencies above Nyquist frequency ($F_s/2$).



Spectrum of 50 ns pulse+ADC noise sampled at 12 bit 250 MSPS.

Alternative interpolation kernels...

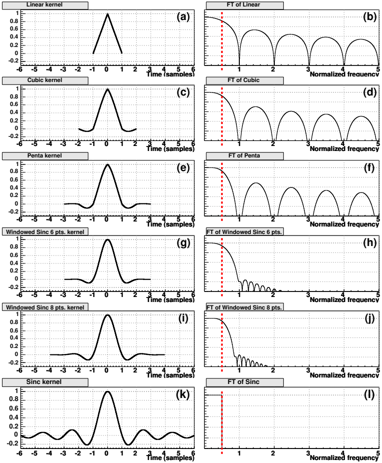


Figure 2: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- Alternative kernels (left) with their Fourier Transforms (right)

Alternative interpolation kernels...

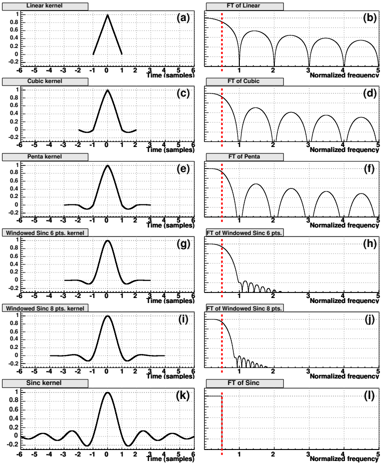


Figure 2: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- Alternative kernels (left) with their Fourier Transforms (right)
- sinc (k) is the only bandlimited kernel!

Alternative interpolation kernels...

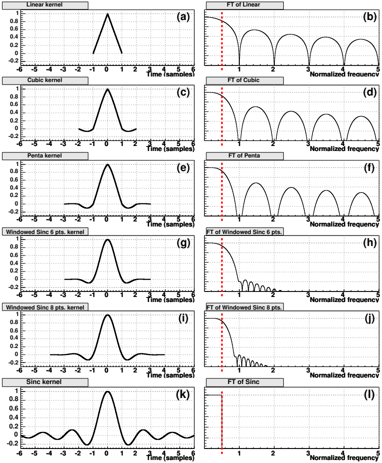


Figure 2: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- Alternative kernels (left) with their Fourier Transforms (right)
- sinc (k) is the only bandlimited kernel!
- $\mathrm{sinc}(t)$ best kernel for BW limited signals and f-domain

Alternative interpolation kernels...

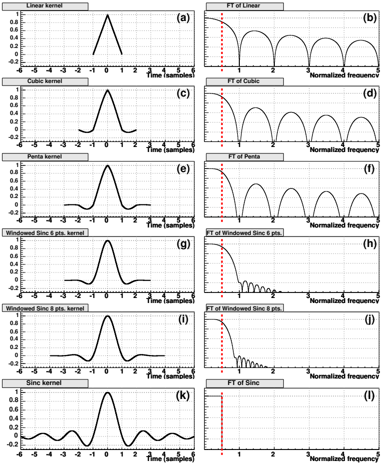


Figure 2: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- Alternative kernels (left) with their Fourier Transforms (right)
- sinc (k) is the only bandlimited kernel!
- $\text{sinc}(t)$ best kernel for BW limited signals and f-domain
- other kernel comparable or better for detector signals and t-domain

Alternative interpolation kernels...

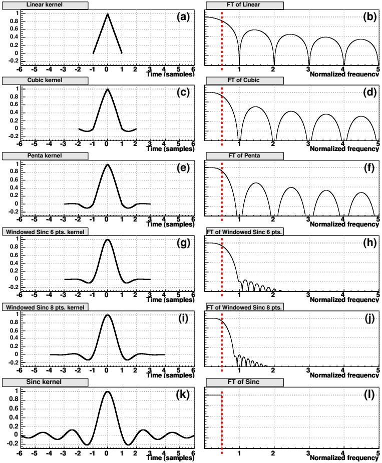


Figure 2: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- Alternative kernels (left) with their Fourier Transforms (right)
- sinc (k) is the only bandlimited kernel!
- $\mathrm{sinc}(t)$ best kernel for BW limited signals and f-domain
- other kernel comparable or better for detector signals and t-domain
- sinc interpolation can't be calculated exactly anyway:

Alternative interpolation kernels...

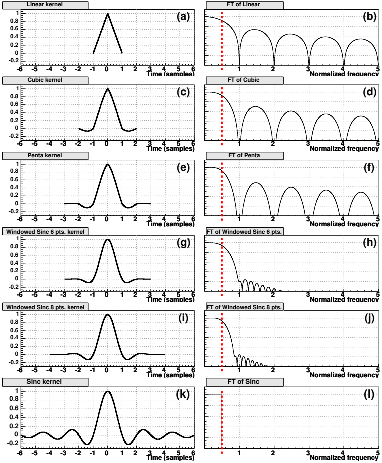


Figure 2: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- Alternative kernels (left) with their Fourier Transforms (right)
- sinc (k) is the only bandlimited kernel!
- $\mathrm{sinc}(t)$ best kernel for BW limited signals and f-domain
- other kernel comparable or better for detector signals and t-domain
- sinc interpolation can't be calculated exactly anyway:
  1. needs ∞ number of samples (sinc not limited in t-domain);

Alternative interpolation kernels...

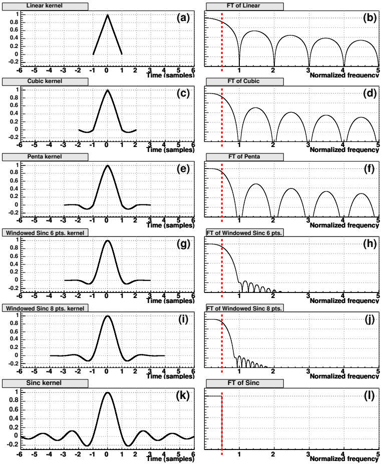


Figure 2: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- Alternative kernels (left) with their Fourier Transforms (right)
- sinc (k) is the only bandlimited kernel!
- $\mathrm{sinc}(t)$ best kernel for BW limited signals and f-domain
- other kernel comparable or better for detector signals and t-domain
- sinc interpolation can't be calculated exactly anyway:
  1. needs ∞ number of samples (sinc not limited in t-domain);
  2. contribution from "distant" samples $\leq$ finite numerical precision; (bad: no exact calc.; good: $\#$ of terms $< \infty$);

Interpolation: alternative kernels

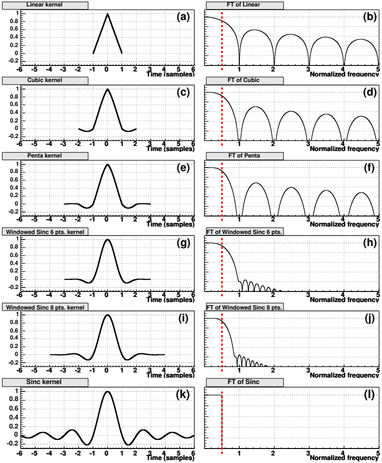


Figure 3: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- What happens with the alternatives?

Interpolation: alternative kernels

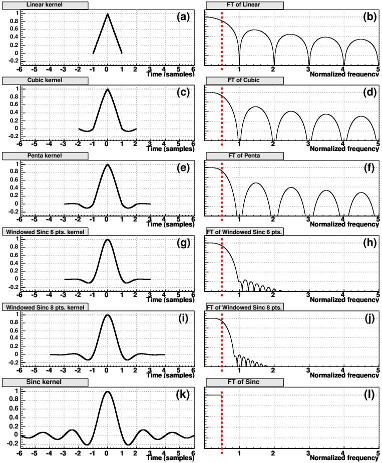


Figure 3: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- What happens with the alternatives?
- 1) not bandwidth limited;

Interpolation: alternative kernels

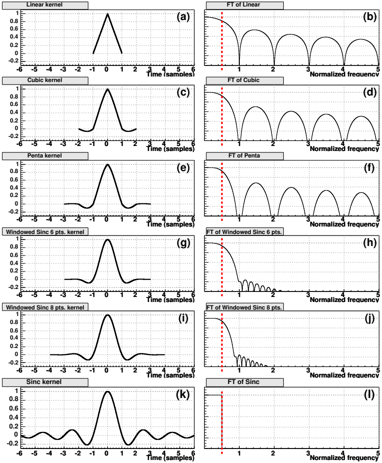


Figure 3: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- What happens with the alternatives?
- 1) not bandwidth limited;
- 2) they also attenuate in-band (below Nyquist).

Interpolation: alternative kernels

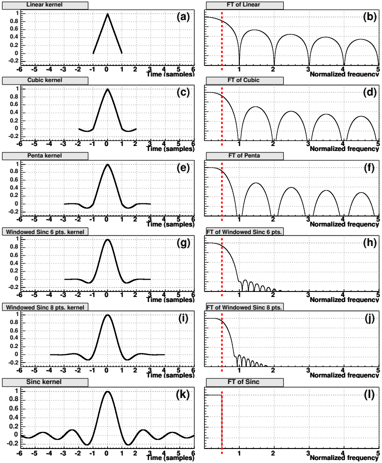


Figure 3: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- What happens with the alternatives?
- 1) not bandwidth limited;
- 2) they also attenuate in-band (below Nyquist).
- windowed sinc (g, i): sinc $\times$ bell shaped function to trim borders and get finite length

Interpolation: alternative kernels

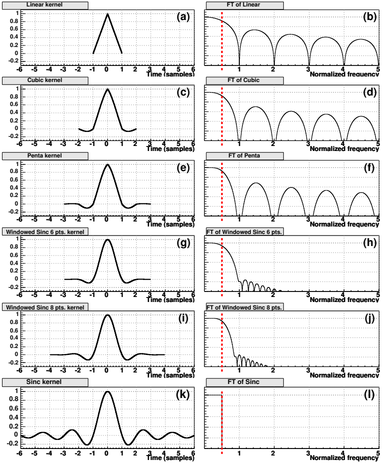


Figure 3: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- What happens with the alternatives?
- 1) not bandwidth limited;
- 2) they also attenuate in-band (below Nyquist).
- windowed sinc (g, i): sinc × bell shaped function to trim borders and get finite length
- other kernels (a, c, e) much more extended in f-domain

Interpolation: alternative kernels

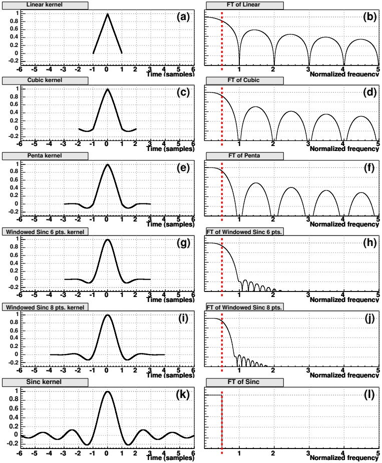


Figure 3: Interpolation kernels and their Fourier Transforms (from [Bardelli2005])

- What happens with the alternatives?
- 1) not bandwidth limited;
- 2) they also attenuate in-band (below Nyquist).
- windowed sinc (g, i): sinc $\times$ bell shaped function to trim borders and get finite length
- other kernels (a, c, e) much more extended in f-domain
- linear (tent) kernel (a) amounts to linear interpolation (connects points with segments)

Interpolation: alternative kernels...

Other interpolation kernels (from [Keys1981]):

- kernel b) corresponds to a) in previous slide;
- kernel d) is [Keys1981] version of a cubic kernel;

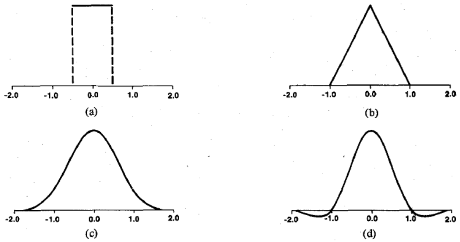


Fig. 1. Interpolation kernels. (a) Nearest-neighbor. (b) Linear interpolation. (c) Cubic spline. (d) Cubic convolution.

- a), a.k.a. "box interpolation" kernel, produces a stepped result;
- c) is cubic box-spline (a case for which $c_n \neq x[n]$)

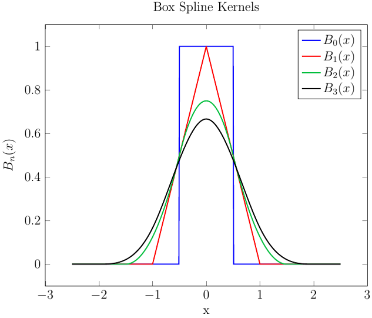
Generic interpolation and cubic splines

Starting from samples $x[n] = x(n\,T_s)$, look for $f(t)$ in

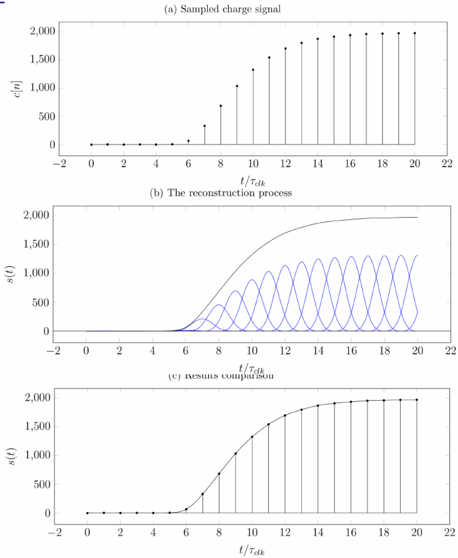
$$K(g,\ T_s) = \left\{ f(t) \mid f(t) = \sum_{m=-\infty}^{+\infty} c_m\, g(t/T_s - m) \right\}$$

Find member of K passing through the samples (i.e. find c_m):
$\sum_{m=-\infty}^{+\infty} c_m\, g(n-m) = x[n] \quad \forall n \in \mathbb{Z}$

E.g.: $g(t)$ could be a box-spline (a.k.a. B-spline):
B-spline kernels obtained from box function by multiple convolution with itself (Picture from [Ottanelli2016]).

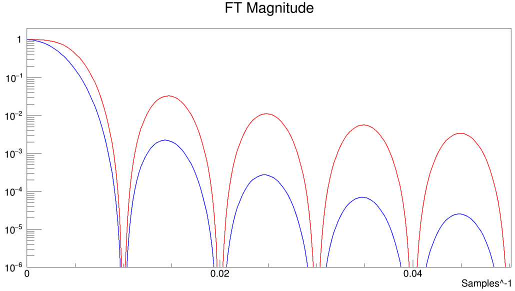


Interpolation at work (from [Ottanelli2016])



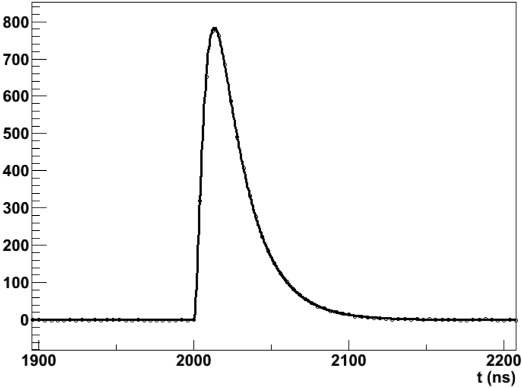
Cubic kernels compared

Cubic kernel c) of Fig.3 [Bardelli2004]
Cubic B-spline [Hou1978]



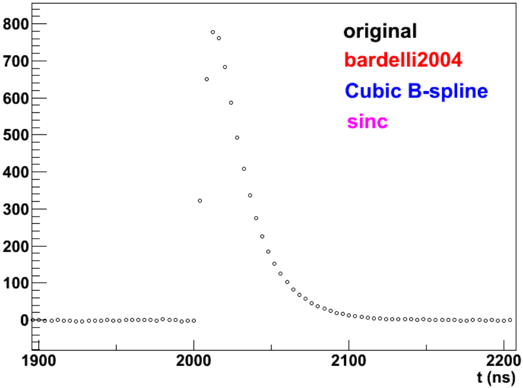
Faster and simpler: [Bardelli2004] (c_m are just signal samples)
More powerful: cubic B-splines (c_m cumbersome to calculate but... a fast approximated calc possible using IIR or FIR filters [Unser1993, Ottanelli2016]!)

Interpolating a detector pulse



Original signal (note the "sudden" start) and its samples.
Sampling includes ADC noise (12 bit, 250 MSPS, 10 ENOB).

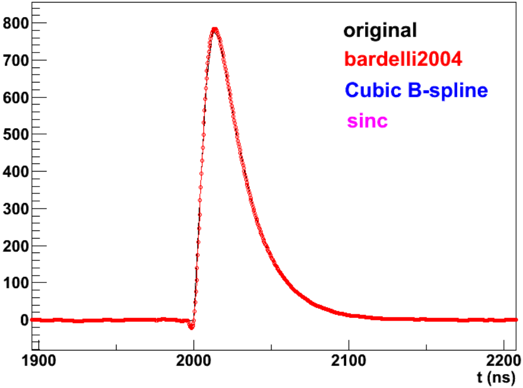
Interpolating a detector pulse



This is what we know after sampling.

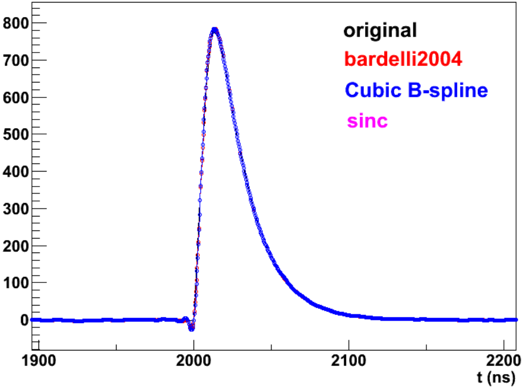
Perfect reconstruction not possible (noise, aliasing,...).

Interpolating a detector pulse



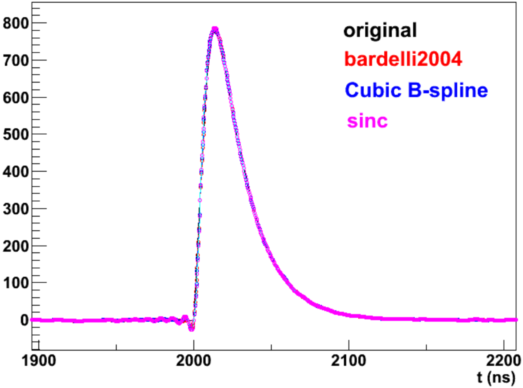
Cubic interpolation from [Bardelli2004], $\times 10$ oversampling.
Exploits the unique polynomial through 4 consecutive samples.

Interpolating a detector pulse



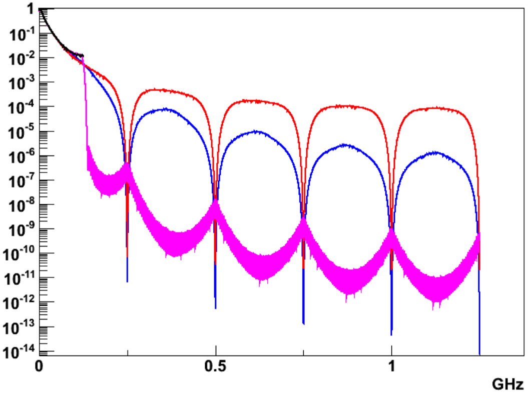
Now we show Cubic Spline interpolation, $\times 10$ oversampling.
Note different behaviour where signal varies rapidly.

Interpolating a detector pulse



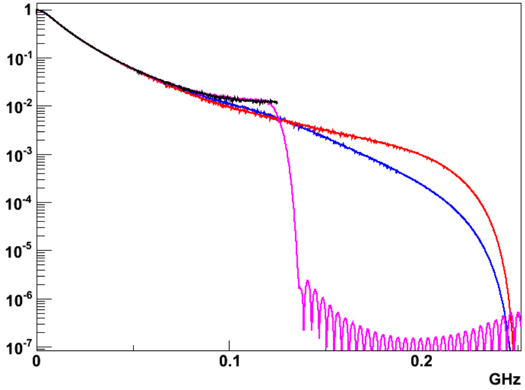
...and finally sinc interpolation. Note: all similar where signal varies slowly. Artifact and different behaviour at fast transition. Why?

Interpolating a detector pulse



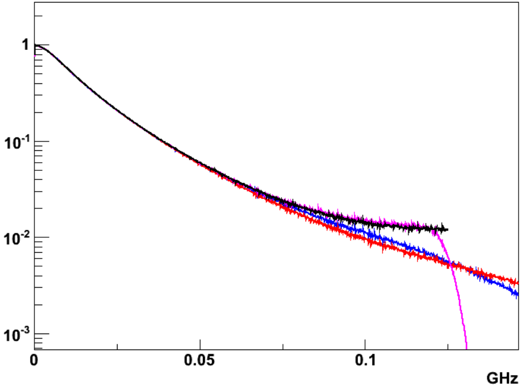
Original spectrum (black) ends at original Nyquist freq. Interpolated signal spectrum extended beyond Nyquist (black is periodic!).

Interpolating a detector pulse



New signal spectrum distorted below Nyquist. Sinc interpolation: less in-band attenuation and less artifacts beyond Nyquist (good?)

Interpolating a detector pulse



Expanded view: frequencies below Nyquist ($F_s/2 = 125\,\mathrm{MHz}$). Sinc interpolation preserves amplitude where noise could dominate.

Interpolation error and sampling clock phase

- N.B. in what follows we assume noiseless signals!

Interpolation error and sampling clock phase

- N.B. in what follows we assume noiseless signals!
- not under Shannon $\Longrightarrow$ interpolated signal $\neq$ original signal

Interpolation error and sampling clock phase

- N.B. in what follows we assume noiseless signals!
- not under Shannon $\Longrightarrow$ interpolated signal $\neq$ original signal
- we could define $e(t) = S(t) - f(t)$ as *interpolation error*

Interpolation error and sampling clock phase

- N.B. in what follows we assume noiseless signals!
- not under Shannon $\Longrightarrow$ interpolated signal $\neq$ original signal
- we could define $e(t) = S(t) - f(t)$ as *interpolation error*
- $e(t)$ depends on the position of the interpolation nodes!

Interpolation error and sampling clock phase

- N.B. in what follows we assume noiseless signals!
- not under Shannon $\Longrightarrow$ interpolated signal $\neq$ original signal
- we could define $e(t) = S(t) - f(t)$ as *interpolation error*
- $e(t)$ depends on the position of the interpolation nodes!
- in other words: reconstruction will be different if original signal is sampled at different points

Interpolation error and sampling clock phase

- N.B. in what follows we assume noiseless signals!
- not under Shannon $\Longrightarrow$ interpolated signal $\neq$ original signal
- we could define $e(t) = S(t) - f(t)$ as *interpolation error*
- $e(t)$ depends on the position of the interpolation nodes!
- in other words: reconstruction will be different if original signal is sampled at different points
- this is what happens in digitizing detector signals: sampling clock phase is random with respect to detector signal

Interpolation error and sampling clock phase

- N.B. in what follows we assume noiseless signals!
- not under Shannon $\Longrightarrow$ interpolated signal $\neq$ original signal
- we could define $e(t) = S(t) - f(t)$ as *interpolation error*
- $e(t)$ depends on the position of the interpolation nodes!
- in other words: reconstruction will be different if original signal is sampled at different points
- this is what happens in digitizing detector signals: sampling clock phase is random with respect to detector signal
- suppose you digitize the same signal shape many times, looking at reconstructed amplitude at a fixed time (t_0 after signal start): each time you find a slightly different amplitude

Interpolation error and sampling clock phase

- N.B. in what follows we assume noiseless signals!
- not under Shannon $\Longrightarrow$ interpolated signal $\neq$ original signal
- we could define $e(t) = S(t) - f(t)$ as *interpolation error*
- $e(t)$ depends on the position of the interpolation nodes!
- in other words: reconstruction will be different if original signal is sampled at different points
- this is what happens in digitizing detector signals: sampling clock phase is random with respect to detector signal
- suppose you digitize the same signal shape many times, looking at reconstructed amplitude at a fixed time (t_0 after signal start): each time you find a slightly different amplitude
- it's a kind of noise: *interpolation noise*!

Simulation: interpolation "noise"

- sampling of noiseless signal $S(t)$ adding a random time shift $0 < \delta < T_s$, i.e. $S[n] = S(n\,T_s + \delta)$

Simulation: interpolation "noise"

- sampling of noiseless signal $S(t)$ adding a random time shift $0 < \delta < T_s$, i.e. $S[n] = S(n\,T_s + \delta)$
- each signal has its δ (same for all samples)

Simulation: interpolation "noise"

- sampling of noiseless signal $S(t)$ adding a random time shift $0 < \delta < T_s$, i.e. $S[n] = S(n\,T_s + \delta)$
- each signal has its δ (same for all samples)
- simulation: generate a set of $S_\delta[n]$

Simulation: interpolation "noise"

- sampling of noiseless signal $S(t)$ adding a random time shift $0 < \delta < T_s$, i.e. $S[n] = S(n\,T_s + \delta)$
- each signal has its δ (same for all samples)
- simulation: generate a set of $S_\delta[n]$
- for each $S_\delta[n]$ obtain interpolated $S_\delta(t)$ ($g(t)$ is kernel):

$$S_\delta(t) = \sum_{n=-\infty}^{+\infty} S_\delta[n] \cdot g\left(\frac{t}{T_s} - n\right)$$

Simulation: interpolation "noise"

- sampling of noiseless signal $S(t)$ adding a random time shift $0 < \delta < T_s$, i.e. $S[n] = S(n\,T_s + \delta)$
- each signal has its δ (same for all samples)
- simulation: generate a set of $S_\delta[n]$
- for each $S_\delta[n]$ obtain interpolated $S_\delta(t)$ ($g(t)$ is kernel):

$$S_\delta(t) = \sum_{n=-\infty}^{+\infty} S_\delta[n] \cdot g\left(\frac{t}{T_s} - n\right)$$

- perfect reconstruction $\Longrightarrow S(t-\delta) = S_\delta(t)$ ($\equiv S(t) = S_\delta(t+\delta)$)

Simulation: interpolation "noise"

- sampling of noiseless signal $S(t)$ adding a random time shift $0 < \delta < T_s$, i.e. $S[n] = S(n\,T_s + \delta)$
- each signal has its δ (same for all samples)
- simulation: generate a set of $S_\delta[n]$
- for each $S_\delta[n]$ obtain interpolated $S_\delta(t)$ ($g(t)$ is kernel):

$$S_\delta(t) = \sum_{n=-\infty}^{+\infty} S_\delta[n] \cdot g\left(\frac{t}{T_s} - n\right)$$

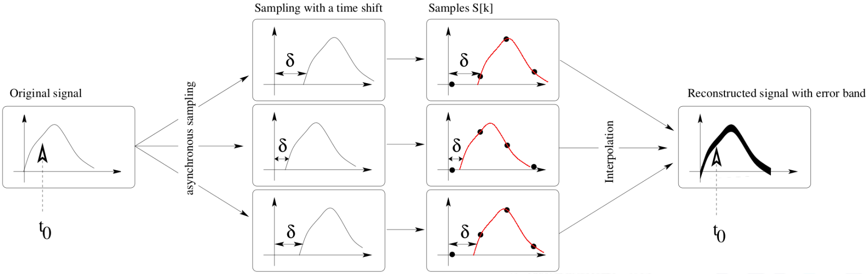
- perfect reconstruction $\Longrightarrow S(t-\delta) = S_\delta(t)$ ($\equiv S(t) = S_\delta(t+\delta)$)
- amplitude variance at fixed $t = t_0$ (with respect to true value $S(t_0) = S_0$):

$$\sigma^2 = \frac{1}{T_s}\int_0^{T_s} \left[S_0 - S_\delta(t_0+\delta)\right]^2 \,\mathrm{d}\delta$$

Interpolation "noise" simulation: a picture

Variance formula with explicit kernel dependence [Bardelli2005]:

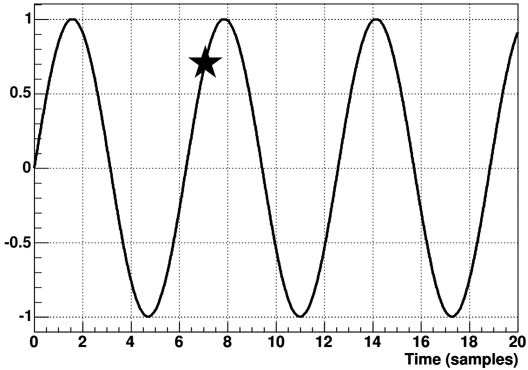
$$\sigma^2 = \frac{1}{T_s}\int_0^{T_s}\left[S_0 - \sum_{n=-\infty}^{+\infty} S(kT_s-\delta)\cdot g\left(\frac{t_0+\delta}{T_s}-n\right)\right]^2 \mathrm{d}\delta$$



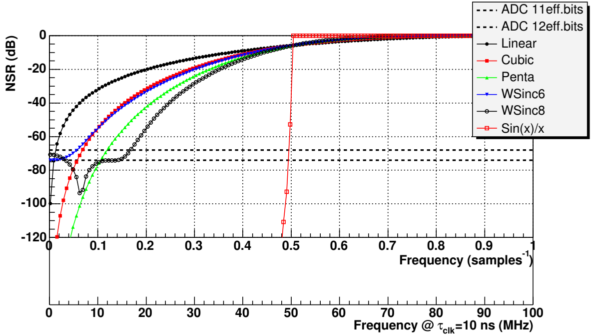
N.B. generally, red (reconstructed) signals have different shapes!

Sine simulation: select point as t_0

Simulation for sinusoidal signal: we will study amplitude fluctuations at fixed t_0, point marked with $\star$, also trying different kernels.

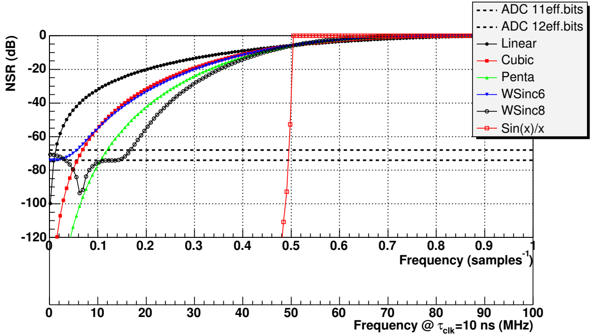


Sine simulation for $T_s = 10\,\text{ns}$: fluctuations at t_0



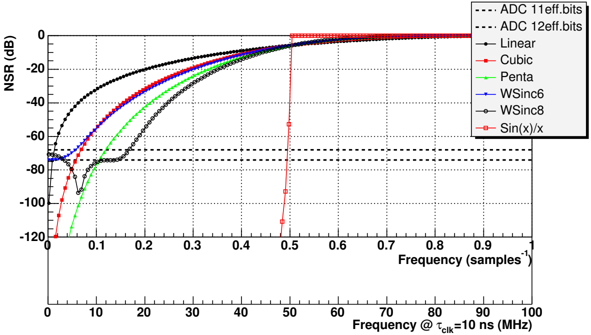
- sine is BW limited $\Longrightarrow$ sinc is the best kernel!

Sine simulation for $T_s = 10\,\text{ns}$: fluctuations at t_0



- sine is BW limited $\Longrightarrow$ sinc is the best kernel!
- note the reference ADC noise levels (two ADC's, different ENOB);

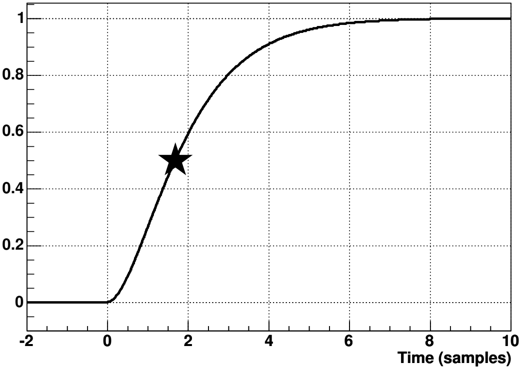
Sine simulation for $T_s = 10\,\text{ns}$: fluctuations at t_0



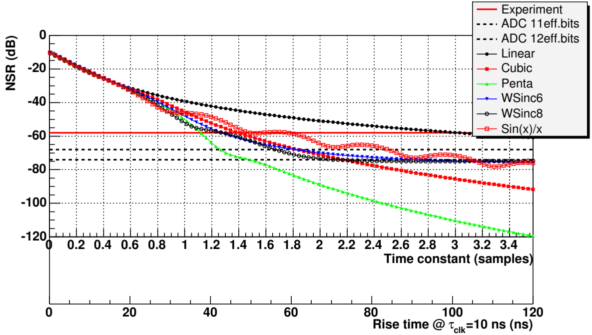
- sine is BW limited $\Longrightarrow$ sinc is the best kernel!
- note the reference ADC noise levels (two ADC's, different ENOB);
- when interpolation noise$<$ADC noise, we can forget about it

Preamp simulation: select point as t_0

Simulation for charge preamp signal: we will study amplitude fluctuations at fixed t_0, point marked with $\star$, also trying different kernels.

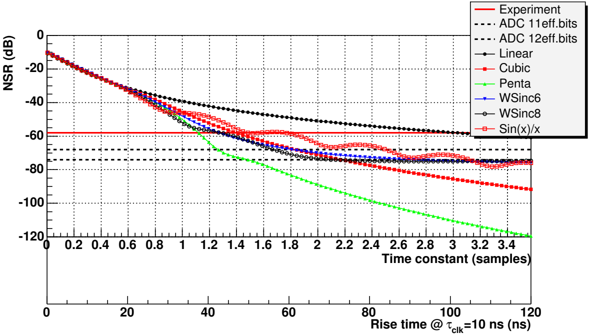


Preamp simulation for $T_s = 10\,\text{ns}$: fluctuations at t_0



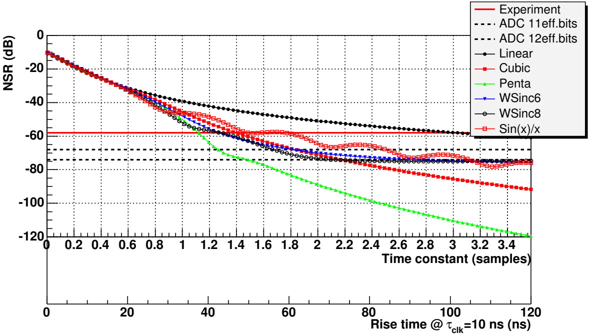
- three reference noise levels (added measured noise in actual FEE);

Preamp simulation for $T_s = 10\,\text{ns}$: fluctuations at t_0



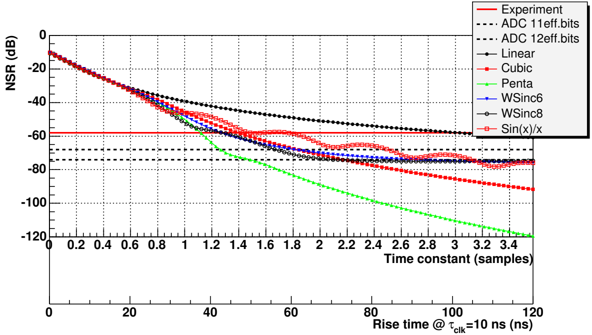
- three reference noise levels (added measured noise in actual FEE);
- short rise-time $\Longrightarrow$ interpolation noise dominates

Preamp simulation for $T_s = 10\,\text{ns}$: fluctuations at t_0



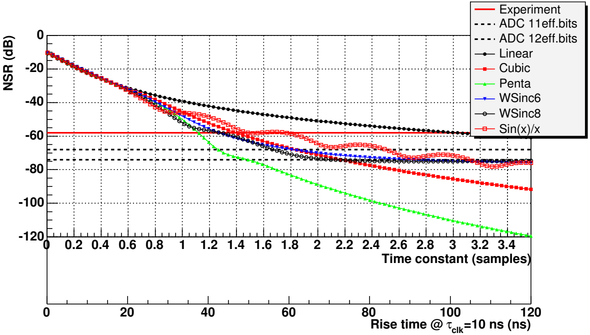
- three reference noise levels (added measured noise in actual FEE);
- short rise-time $\Longrightarrow$ interpolation noise dominates
- $>$ 4 samples in leading edge ($> 40\,\text{ns}$): penta $<$ experimental noise

Preamp simulation for $T_s = 10\,\mathrm{ns}$: fluctuations at t_0



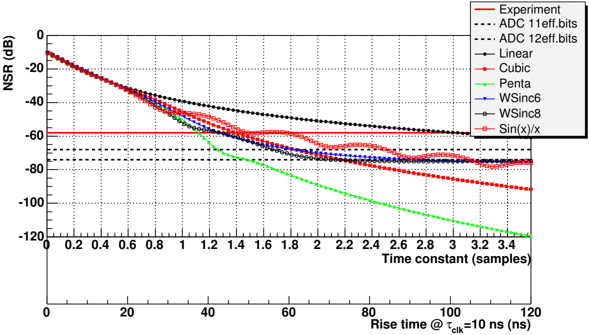
- three reference noise levels (added measured noise in actual FEE);
- short rise-time $\Longrightarrow$ interpolation noise dominates
- $>$ 4 samples in leading edge ($> 40\,\mathrm{ns}$): penta $<$ experimental noise
- linear?...bah!...front is not linear!

Preamp simulation for $T_s = 10\,\text{ns}$: fluctuations at t_0



- three reference noise levels (added measured noise in actual FEE);
- short rise-time $\Longrightarrow$ interpolation noise dominates
- $>$ 4 samples in leading edge ($> 40\,\text{ns}$): penta $<$ experimental noise
- linear?...bah!...front is not linear!
- sinc? no better...why? no "beyond Nyquist" components in sinc

Preamp simulation for $T_s = 10\,\text{ns}$: fluctuations at t_0



- three reference noise levels (added measured noise in actual FEE);
- short rise-time $\Longrightarrow$ interpolation noise dominates
- $>$ 4 samples in leading edge ($>$ 40 ns): penta $<$ experimental noise
- linear?...bah!...front is not linear!
- sinc? no better...why? no "beyond Nyquist" components in sinc
- cubic interpolation: good performance/complexity ratio

End of First Lesson (2016-10-25)