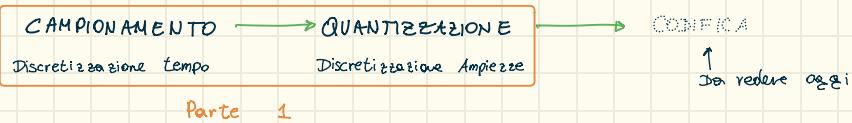


RECAP Nella prima parte

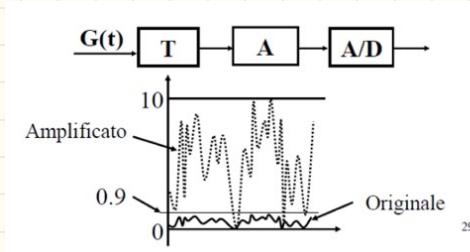


RISOLUZIONE DEL QUANTIZZATORE

Potremmo avere un trasduttore che ha in uscita una grandezza elettrica (per semplicità diciamo che è già una tensione) **molto molto piccola**. Di conseguenza questa grandezza potrebbe essere così piccola da essere minore dell' **LSB** (ovvero l'intervallo di un livello di quantizzazione). Otteniamo come risultato che i 16 (ipotetici) bit del convertitore sono per 15 (most significant) bit zero, mentre l'ultimo bit cambia frequentemente (flicker) tra zero ed uno.

Ovviamente di questo tipo di comportamento non ce ne facciamo nulla.

Quello che si fa è **amplificare il segnale** prima di arrivare al quantizzatore in modo da prendere tutto il FS (fondo scala).



ES: Abbiamo $FS = FS_{MAX} - FS_{MIN} = 10V$
con un Segnale di Ampiezza Max di $G = 0.9V$

Abbiamo da convert. a 8bit

$$\Rightarrow Q = \frac{10V}{2^8} = 39mV \Rightarrow \frac{0.0039V}{0.9V} \cdot 100 = 4.3\% \text{ del segnale}$$

! ALTO !

⇒ SOLUZIONE Amplifichiamo fino ad ottenere un valore max del segnale in input pari a $G = 10V$

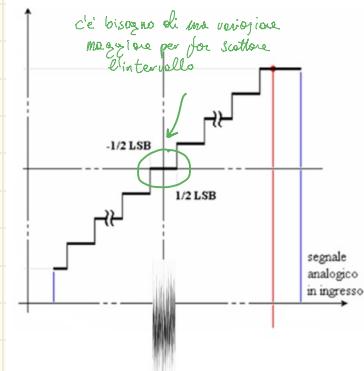
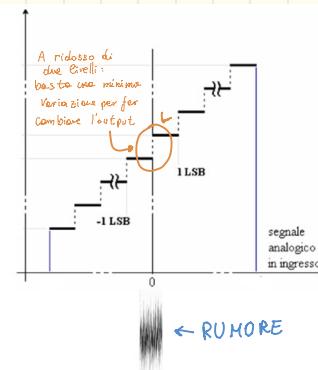
$$\Rightarrow Q = \frac{10V}{2^8} 39mV \Rightarrow \frac{0.0039V}{10V} \cdot 100 = 0.39\% \text{ del segnale in ingresso}$$

✓ BASSO ✓

QUANTIZZAZIONE NON SILENZIATO VS SILENZIATO

La differenza è semplice, per capirlo facciamo un esempio:
Se non stiamo trasmettendo nessun segnale, abbiamo comunque un **rumore di fondo**. A seconda della **caratteristica del quantizzatore** il segnale in out può comportarsi in due modi:

- 1) c'è un flickering tra due livelli di quantizzazione: **non silenziato**
- 2) il segnale rimane costante perché la caratteristica in quel punto è **silenziosa**, ovvero non siamo a ridosso tra due livelli di quantizzazione.



CODIFICA

CODIFICATORE UNIPOLARE

La codifica consiste nell'**associare un codice binario al valore quantizzato del livello corrispondente al valore analogico**.

Ovviamente un codice binario di 16 bit (65536 stati diversi) avrà meno stati di un codice a 32 bit.

Se abbiamo un codice a 4 bit Ed un Fondo scala di +10V, possiamo associare un livello a ciascun codice, che va da 0000 a 1111. Ovviamente avremo $2^4 = 16$ stati.

$$1LSB = Q = \frac{FS}{2^8} = \frac{10V}{16} = 0.625V = 625mV$$

$$2LSB = 2Q = 2 \cdot 625mV$$

$$NLSB = N \cdot Q < FS ! \quad \text{Perché codifichiamo anche lo zero !}$$

BASE 10 NUMBER	SCALE	+10V FS	BINARY	GRAY
+15	+FS - 1LSB = +15/16 FS	9.375	1111	1000
+14	+7/8 FS	8.750	1110	1001
+13	+13/16 FS	8.125	1101	1011
+12	+3/4 FS	7.500	1100	1010
+11	+11/16 FS	6.875	1011	1110
+10	+5/8 FS	6.250	1010	1111
+9	+9/16 FS	5.625	1001	1101
+8	+1/2 FS	5.000	1000	1100
+7	+7/16 FS	4.375	0111	0100
+6	+3/8 FS	3.750	0110	0101
+5	+5/16 FS	3.125	0101	0111
+4	+1/4 FS	2.500	0100	0110
+3	+3/16 FS	1.875	0011	0010
+2	+1/8 FS	1.250	0010	0011
+1	1LSB = +1/16 FS	0.625	0001	0001
0		0.000	0000	0000

Di conseguenza ci accorgiamo che o codifichiamo lo zero oppure codifichiamo l'ultimo valore (corrispondente a FS). Possiamo anche codificare il tutto tramite il **Gray code** che ci permette di tenere sotto controllo gli errori ventrali.

CODIFICATORE BIPOLARE

Avere un codificatore bipolare vuol dire lavorare **anche negli intervalli negativi**, ad esempio tra +5V e -5V.

I computer da noi utilizzati lavorano con un codice binario **complemento a due**. In questa codifica i numeri positivi sono rappresentati normalmente, mentre i numeri negativi sono ottenuti partendo dal corrispettivo negativo, invertendo tutti i suoi bit ed aggiungendo uno: 0001 (+1) -> inverto -> 1110 -> aggiungo uno -> 1111 (-1).

Convertire un convertitore unipolare a bipolare

Se abbiamo un convertitore unipolare ma un segnale che ha valori anche negativi, ci basta sommare un determinato valore per far sì che tutti i valori del segnale siano positivi. Se abbiamo una sinusoide che varia tra +5V e -5V, ci basta sommare un valore pari a 5V in modo che il nuovo segnale vari tra 0V e 10V.

Codifica							
BASE 10 NUMBER	SCALE	$\pm 5V FS$	OFFSET BINARY	TWOS COMP.	ONES COMP.	SIGN MAG.	
+7	+FS - 1LSB = +7/8 FS	+4.375	1111	0111	0111	0111	
+6	+3/4 FS	+3.750	1110	0110	0110	0110	
+5	+5/8 FS	+3.125	1101	0101	0101	0101	
+4	+1/2 FS	+2.500	1100	0100	0100	0100	
+3	+3/8 FS	+1.875	1011	0011	0011	0011	
+2	+1/4 FS	+1.250	1010	0010	0010	0010	
+1	+1/8 FS	+0.625	1001	0001	0001	0001	
0	0	0.000	1000	0000	*0000	*1000	
-1	-1/8 FS	-0.625	0111	1111	1110	1001	
-2	-1/4 FS	-1.250	0110	1110	1101	1010	
-3	-3/8 FS	-1.875	0101	1101	1100	1011	
-4	-1/2 FS	-2.500	0100	1100	1011	1100	
-5	-5/8 FS	-3.125	0011	1011	1010	1101	
-6	-3/4 FS	-3.750	0010	1010	1001	1110	
-7	-FS + 1 LSB = -7/8 FS	-4.375	0001	1001	1000	1111...	
-8	-FS	-5.000	0000	1000			

FINE SLIDES 1

PROBLEMI NEL PASSAGGIO DAC e ADC

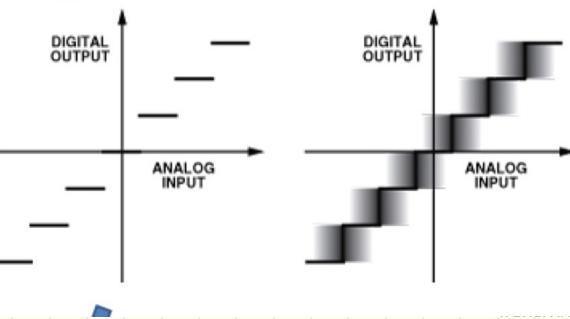
La caratteristica del quantizzatore non è "bella" come quella che abbiamo visto, ma nella realtà le **soglie** per le quali abbiamo in output un certo livello **si spostano**.

Di conseguenza possiamo solo dire che **esiste una certa probabilità che la soglia reale sia quella ideale**.

Abbiamo delle **soglie di transizioni** reali ed ideali; ci aspettiamo che se in ingresso poniamo sempre lo stesso valore, in uscita dovremmo ottenere lo stesso valore ogni volta. Osserviamo però che questo non è vero.

Esistono due codici (k e $k-1$) che hanno la probabilità del 50% di apparire: vuol dire che soglia di transizione reale è compresa tra questi due valori. Avendo n campioni, avremo che il 50%*n sarà k mentre l'altro 50%*n sarà $k-1$.

(a) IDEAL ADC (b) ACTUAL ADC



$T[k]$ Valore di TENSIONE IN INGRESSO che corrisponde allo transizione tra i codici k e $k-1$

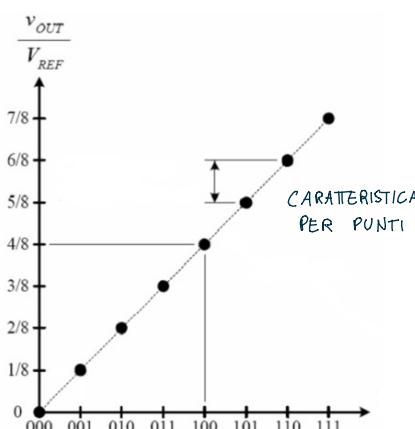
Larghezza di un codice: CODE BIN WIDTH $\Rightarrow W[k] = T[k+1] - T[k]$

BIN WIDTH IDEALE corrisponde al passo di quantizzazione = 1 $\Rightarrow LSB = \frac{V_{FS}}{2^b}$

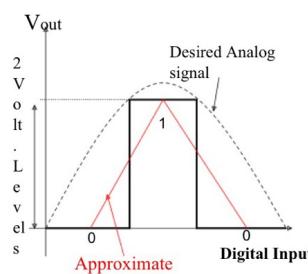
ERRORE $E[k] = T[k] - T_k$

Caratteristica convertitore DAC

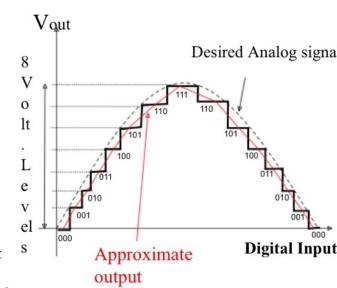
Nel convertitore da digitale ad analogico abbiamo una serie di valori quantizzati (bit) che corrispondono ad una **serie di punti** (valori analogici). È importante notare che tra un codice è l'altro non ci sono **valori intermedi**!



Poor Resolution(1 bit)



Better Resolution(3 bit)



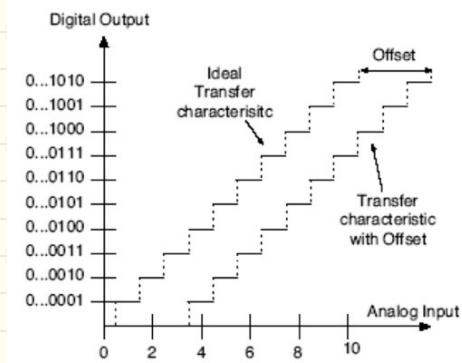
Maggiore è la risoluzione del convertitore DAC e meglio riusciremo a rappresentare il valore analogico!

$$\text{RISOLUZIONE} : V_{LSB} = \frac{V_{Ref}}{2^n}$$

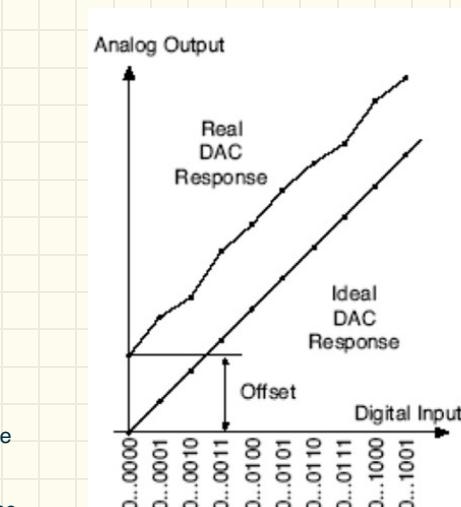
Volte di fondo Scalo
Inverso di prima

ERRORE CARATTERISTICO

ERRORE DI OFFSET



Potrebbe accadere che la caratteristica del convertitore ADC venga **shiftata** verso sinistra o verso destra. In questo caso (destra) otterremmo un valore digitale minore di quello che dovremmo ottenere: ad esempio se abbiamo in ingresso $V=6V$, in output otteniamo "11" ovvero 3, invece di 110!

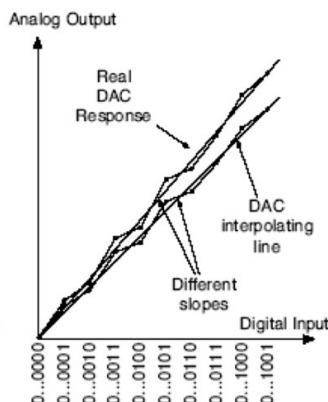
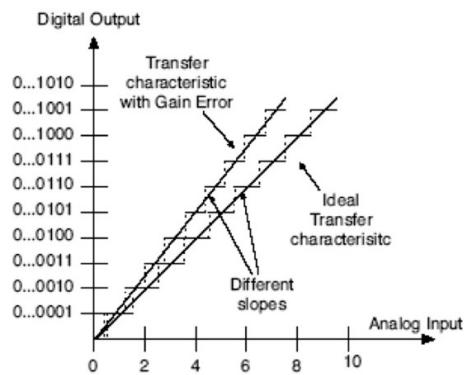


La stessa cosa può accadere nel passaggio da digitale ad analogico

Risolvere l'errore di offset

I convertitori odierni compensano automaticamente questi errori: se posizioniamo l'ingresso a massa (0V) allora l'uscita dovrà darci necessariamente il corrispettivo di 0V (analogico o digitale). Se otteniamo un codice diverso da quello che ci aspettiamo, ci basta sommarlo ad ogni output in modo da compensare l'offset.

ERRORE DI GUADAGNO



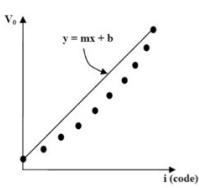
Se il fattore di guadagno cambia anche solo leggermente, la pendenza della retta di conversione cambia, e quindi abbiamo dei valori che non sono esatti.

Risolvere l'errore di guadagno

In questo caso ci basta calcolare l'output di due input che conosciamo, calcoliamo la pendenza e la confrontiamo con quella che dovrebbe essere idealmente. Ci basterà dividere per questo fattore per compensare.

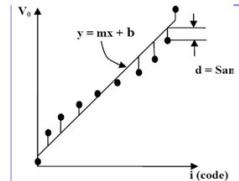
Linee di riferimento

ENDPOINT TERMINAL BASED



BESTFIT

INDEPENDENTLY BASED

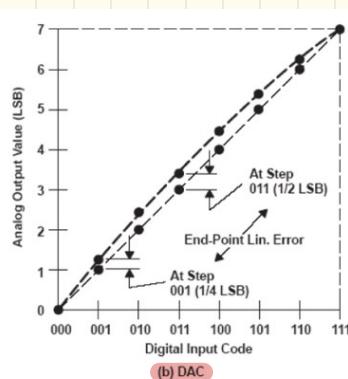
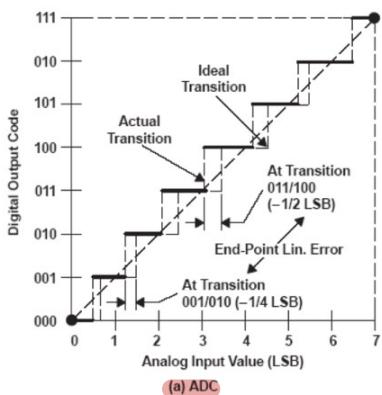


BOH?

Questa è la misura di linearità integrale più utile per applicazioni di misura e controllo per ADC e DAC, infatti la valutazione complessiva degli errori dipende dalla deviazione della caratteristica di trasferimento reale da quella ideale, e non da qualche arbitraria "best fit")

La *best straight line*, tuttavia, dà una migliore previsione di distorsione in applicazioni in c.a. e dà anche un valore più basso di "errore di linearità" su i fogli riportanti i dati del componente.

ERRORE DI DNL E INL

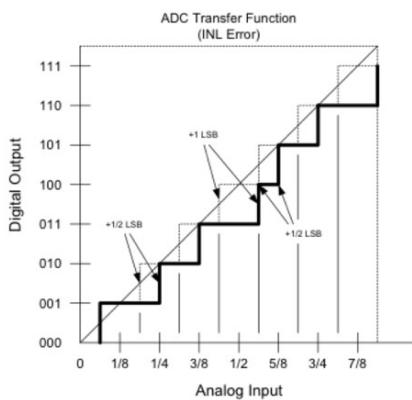


La **pedata** reale (ovvero la distanza tra due soglie successive) può variare rispetto a quella ideale. Possiamo valutare la differenza tra quella reale e quella ideale tramite la **DNL - Differential Non Linearity**

$$DNL[\kappa] = \frac{T[\kappa-1] - T[\kappa] - Q}{Q}$$

Se la "pedata" è ideale $DNL[\bar{\kappa}] = 1$

E' sostanzialmente un errore PERCENTUALE



La **INL - Integral Non Linearity** è sempre una misura dell'errore delle pedate, ma ci dice **quanto è spostata** (positivamente dx o negativamente sx) la pedata corrente (di cui calcoliamo l'INL) rispetto a quella che dovrebbe essere la posizione ideale.

In altre parole, se siamo al gradino κ e guardiamo i gradini sottostanti, vedremo che alcuni sono più ampi degli altri (pedata, non altezza!). Possiamo quindi calcolare di quanto ci siamo scostati rispetto alla posizione κ del convertitore ideale.

Calcoliamo L'INL andando a fare la somma di tutti gli errori precedenti al corrente:

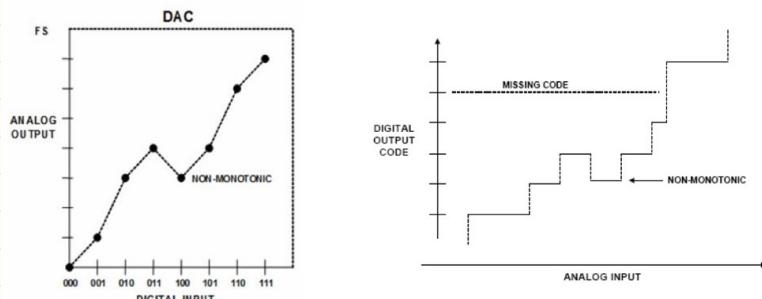
$$INL[\kappa] = \frac{T_\kappa - T[\kappa]}{V_{FS}} \cdot 100$$

$$INL[\kappa] = \sum_{i=1}^{\kappa} DNL[i]$$

$$INL = \max(|INL[\kappa]|)$$

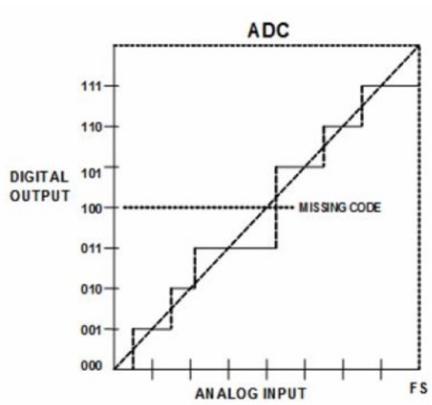
Il valore INL caratterizza il convertitore e ci fa capire il valore massimo con cui il convertitore si discosta dal valore ideale.

NON MONOTONICITÀ DEL CONVERTITORE



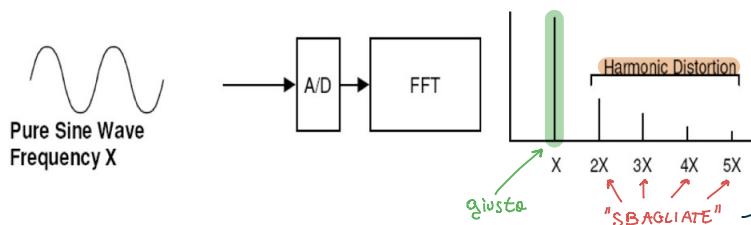
Quando all'aumentare della tensione di ingresso **l'uscita invece di aumentare diminuisce** per poi aumentare di nuovo abbiamo un errore.

Nei convertitori, più ci si avvicina all'utilizzo massimo dichiarato dal costruttore (*ipotizzo ad esempio la frequenza n.d.s.*) e **più si ottengono errori**.



Anche quando si **salta** un codice abbiamo un errore.

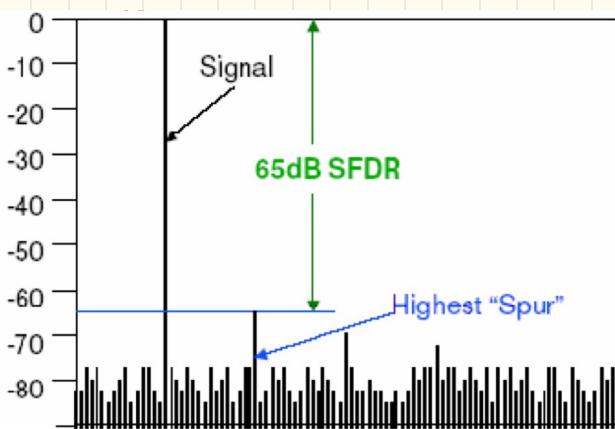
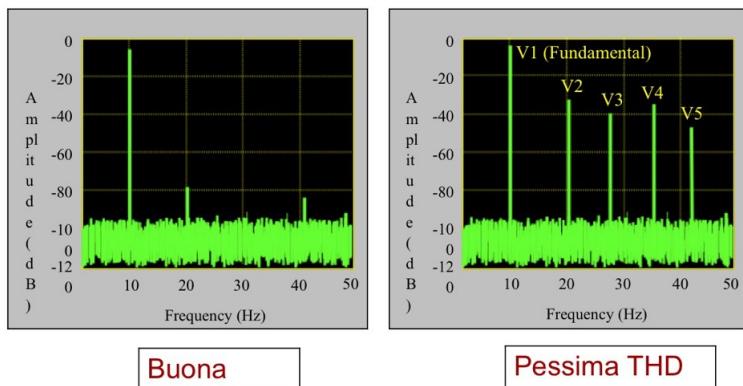
PARAMETRI USATI PER VALUTARE LA MACCHINA (ADC - DAC)



Quando abbiamo un **sistema lineare** che manipola un segnale (ad esempio sinusoidale), possiamo fare la **trasformata di Fourier** per ottenere l'armonica (singola) corrispondente.

Il problema è che se il sistema non è lineare, non troviamo una singola armonica!

Provocate da
una distorsione



$$THD (-dB) = 20 \log \left(\sqrt{V_2^2 + V_3^2 + \dots + V_n^2} / V_1 \right)$$

LOW IS BETTER

Armonica fondamentale

Tutte le altre

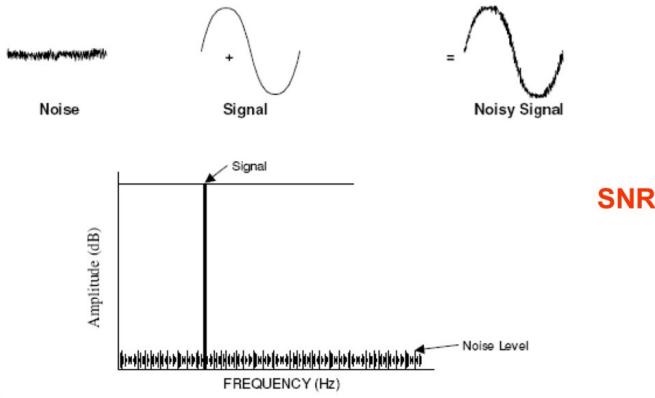
SFDR

SFDR (Spurious Free Dynamic Range) è la differenza tra il valore del segnale di uscita desiderato (Signal) e il valore dell'ampiezza più alta tra le armoniche dello spettro del segnale di uscita (Highest Spur).

Se abbiamo che le armoniche di distorsione sono di un valore si re a quello dell'armonica principale (segnale) il convertitore ha una banda minore su cui lavorare e quindi è meno performante. Su quest'ultima affermazione non è che ci abbia capito molto n.d.s.

#n.d.s.

SNR RAPPORTO SEGNALE RUMORE



Se abbiamo un segnale e gli sommiamo del rumore, otteniamo un segnale rumoroso. Se facciamo la trasformata di Fourier di questo segnale rumoroso, otteniamo qualcosa simile a quello graficato in basso. Possiamo calcolare l'**SNR** come il rapporto tra i **valori efficaci** (ovvero il valore costante corrispondente ad un segnale non costante) del segnale originario e quello del rumore:

$$SNR = 20 \log_{10} \left(\frac{V_x \text{ rms}}{N \text{ rms}} \right)$$

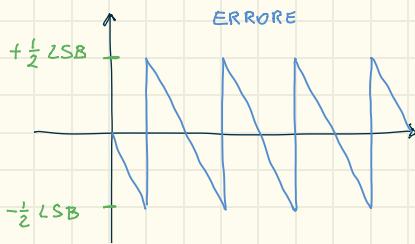
in dB

Segnale

Rumore Senza ARMONICHE

"NOISE!"

VALORE EFFICACIE DELL'ERRORE DI QUANTIZZAZIONE



$$RMS = \sqrt{\frac{1}{T} \int_0^T f(x)^2 dx}$$

$$\Rightarrow E_j^2 = \frac{1}{q_1} \int_{-\frac{q}{2}}^{+\frac{q}{2}} E_j^2 dE = \frac{q^2}{12}$$

POTENZA DEL RUMORE \bar{N}^2

ovvero $\pm \frac{1}{2} LSB$

segnale a destra di sega

Questo risultato è importantissimo perché ci dice che il valore efficace dell'errore non dipende dalla frequenza di campionamento, ma solo dal passo di campionamento (LSB). Di conseguenza l'errore dipende dalla costruzione del convertitore.

Elevando al quadrato il valore efficace (come in questo caso) otteniamo la potenza del rumore, ed è costante!

Pongo in ingresso una Sinusoide

$$\bar{F}^2(t) = \frac{1}{2\pi} \int_0^{2\pi} A^2 \sin^2(\omega t) dt = \frac{A^2}{2} \bar{F}^2$$

$$F^2 = \frac{A^2}{2}$$

Potenza del valore efficace di una sinusoide
dal calcolo del VE integ.

$$q = \frac{FS}{2^N} \quad e \quad F(t) = \frac{A}{\sqrt{2}} \quad e \quad 2A = FS \text{ convertitore} \quad \Rightarrow q = \frac{2A}{2^N} = \frac{2A}{2 \cdot 2^{N-1}} = \frac{A}{2^{N-1}}$$

V.E. Sinusoide

$$q^2 = \frac{A^2}{2^{2(N-1)}}$$

$$N^2 = \frac{q^2}{12} = \frac{\frac{A^2}{2^{2(N-1)}}}{12} = \frac{A^2}{12 \cdot 2^{2(N-1)}} = \frac{A^2}{12 \cdot \frac{2^{2N}}{3 \cdot 2}} = \frac{A^2}{3 \cdot 2^{2N}}$$

dal calcolo del VE integ.

$$2^{(N-1)} = \frac{2^{(N-1)}}{2} = \frac{N}{2^2}$$

$$\begin{aligned} \Rightarrow SNR &= 20 \log_{10} \frac{\bar{F}}{\bar{N}} = 20 \log_{10} \left(\frac{\bar{F}^2}{\bar{N}^2} \right)^{\frac{1}{2}} = 10 \log_{10} \frac{\bar{F}^2}{\bar{N}^2} \quad (1) \\ &= 10 \log \frac{3 \cdot 2^N}{2} = 10 \log \left(\frac{3}{2} \right) + 10 \log (2^N) = 10 \log \left(\frac{3}{2} \right) + N \cdot 10 \log (2) = \\ &= 1.76 dB + 6.02 N dB \quad QED \end{aligned}$$

ES: Convertitore a 40 bit $\Rightarrow n = 40$

$$\Rightarrow SNR = 6.02 \cdot 40 + 1.76 dB = 60.2 + 1.76 = 61.86 dB$$

Vuol dire che quando costruiamo un convertitore, possiamo conoscere immediatamente il suo rapporto segnale-rumore.

Processo inverso:
in convertitore ha un $SNR = 80 dB$, bit = ?

$$SNR = 6.02 \cdot n + 1.76 = 80 \Rightarrow$$

$$n = \frac{SNR - 1.76}{6.02}$$

$$\Rightarrow n_{80} = 12.99 \approx 13 \text{ bit}$$

bit conv.

Se però lo spettro in out presenta degli spike (oltre all'armonica principale, ovvero il segnale) non lavoriamo ad 80dB, ma ad un valore minore (corrispondente al valore dello spike più alto):

SPIKE \leftrightarrow DISTORSIONE

Spike = -65 dB $\Rightarrow n_{65} = 10.5 \text{ bit} < 13 \text{ bit}$ dichiarati dal produttore

SINAD

Signal To Noise And Distortion

Questo indice è più accurato, perché al denominatore poniamo anche le armoniche di distorsione

$$\text{SINAD} = 20 \log_{10} \left(\frac{\sqrt{V_{X\text{RMS}}}}{N_{\text{RMS}} + D_{\text{RMS}}} \right)$$

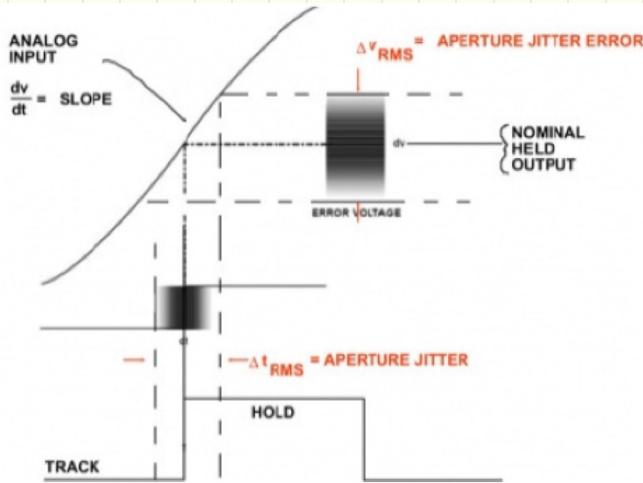
Velocità effettiva della DISTORSIONE

ENOB Effective Number Of Bit

$$ENOB = \frac{SINAD_{dB} - 1.76}{6.02}$$

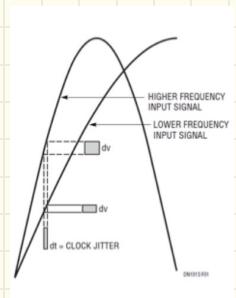
Questo valore ci dice il numero al quale la macchina **lavora effettivamente**.

ERRORE IN AMPIEZZA ~ JITTER



Abbiamo visto che **idealtamente** quando effettuiamo l'operazione di sample & hold, appena diamo il segnale di hold andiamo a campionare il valore in input. **Questo nella realtà non vale**: abbiamo un'incertezza quando diamo il segnale di hold: il segnale reale può avvenire un po' prima o un po' dopo, e di conseguenza **il valore del segnale in input può essere un po' sotto o un po' sopra** rispetto a quello che ci aspettiamo.

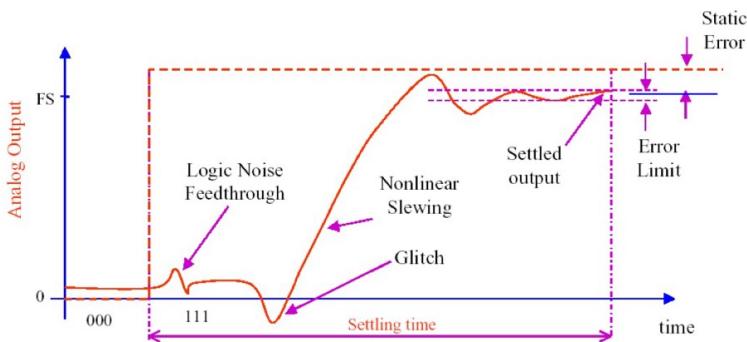
Inoltre, questo errore **dipende dalla derivata del segnale**: è ovvio che se il segnale ha una pendenza maggiore (**una frequenza maggiore!**) avremo un errore maggiore.



SETTLING TIME

Convertitori

Digitale → Analogico



Gli impulsi non sono altro che un segnale ad altissima frequenza. Abbiamo visto quali sono gli effetti di una frequenza molto alta: tramite le capacità parassite si vengono a creare dei veri e propri **cortocircuiti** che attraversano la scheda producendo risultati non desiderati.

Quando cambiamo immediatamente i codici di input, ad esempio da 00111111 passiamo a 01000000 andiamo a cambiare lo stato di un'enorme numero di componenti (transistors) e di conseguenza gli eventuali micro glitch di ogni componente vengono sommati e creano un glitch molto più grande, che potrebbe creare dei problemi.

Se un convertitore DAC riceve in input un valore del tipo 111 (il più grande possibile) dovrebbe rispondere con un'uscita analogica "alta" corrispondente al fondo scala FS. Questo avviene, ma **non istantaneamente**. Dal momento di input a quello di output **abbiamo un transitorio** in cui compaiono dei **glitch** (impulsi) seguiti da **una salita non lineare**.

Per testare un convertitore si agisce proprio in questo modo, gli si dà in input improvvisamente il codice "più grande" immediatamente dopo aver dato il codice "più grande" è si osserva l'output

