

# **ELABORAZIONE DEL SUONO**

INTRODUZIONE ALLA SINTESI AUDIO ED ELEMENTI DI PROGRAMMAZIONE  
CSOUND

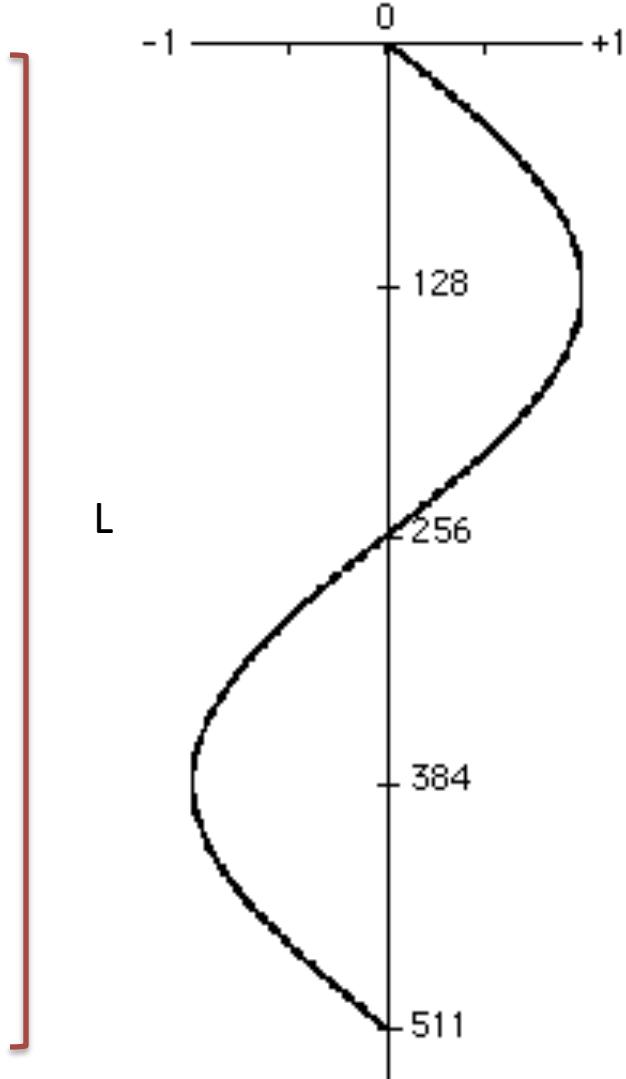
Parte Seconda

E. Giordani

# L'oscillatore digitale

Indirizzo      Dato o valore

0	0.0000
1	0.0123
2	0.0245
⋮	⋮
127	0.9999
128	1.0000
129	0.9999
⋮	⋮
256	0.0000
257	-0.0123
⋮	⋮
383	-0.9999
384	-1.0000
385	-0.9999
⋮	⋮
511	-0.0123



Rappresentazione tabellare di un periodo completo di una forma d'onda sinusoidale composta da 512 valori.

$$F_n = sr / L$$

$F_n$  = frequenza naturale  
 $sr$  = sampling rate o frequenza di campionamento  
 $L$  = lunghezza tabella

# Frequenza dell' oscillatore digitale

Ad esempio per  $sr = 44100$  e  $L = 512$  si ha:

$$Fn = 44100/512 = 86,1328125 \text{ Hz}$$

La frequenza è generata attraverso un algoritmo di incremento progressivo e recursivo della fase:

$$\text{Fase} = \text{Fase} + \text{Inc}$$

dove Inc è il “passo di lettura della tabella”. La frequenza generica può allora essere espressa nel modo seguente:

$$F = \text{Inc} (sr / L)$$

La fase corrente rappresenta quindi l'indirizzo a cui accedere in tabella per estrarre il campione.

$\text{Inc} = 1$        $F$  coincide con la frequenza naturale  $Fn$

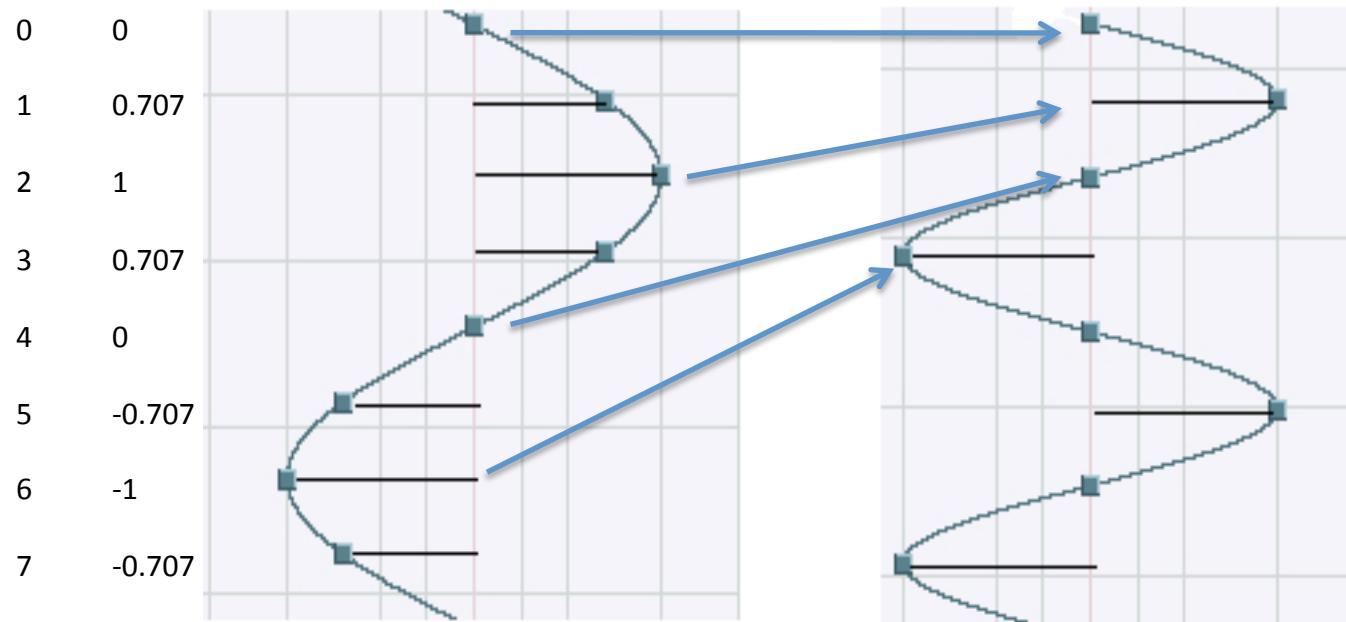
$\text{Inc} > 1$        $F > Fn$

$\text{Inc} < 1$        $F < Fn$

# Incremento non unitario

Per una tabella a 8 valori (non è un caso realistico ma utile per l'esempio) se  $\text{Inc} = 2$  si avrà una sequenza di indirizzi pari a:

0, 2, 4, 6, 0, 2, 4, 6, 0, 2, ....etc.



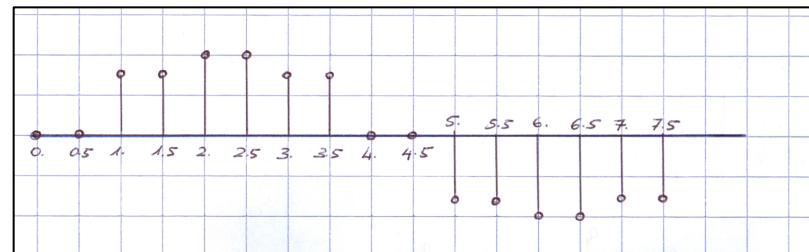
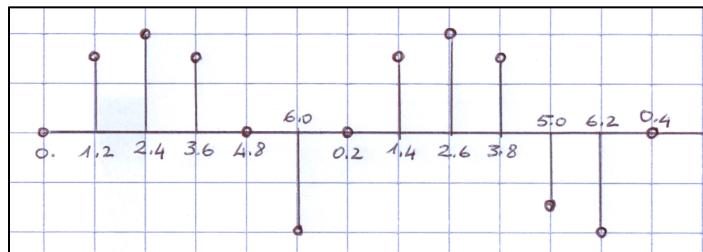
Si ottiene così una frequenza doppia e una forma d'onda virtualmente indistorta.

# Incremento frazionario

Per incrementi frazionari ( $> 1$  o  $< 1$ ) si può sintetizzare qualsiasi frequenza, ma poiché l'indirizzo dei valori è un numero intero occorre fare una scelta quando la fase corrente è un numero frazionario. Ad esempio se  $\text{Inc} = 1.2$  si ha la sequenza di indirizzi e valori:

0	1.2	2.4	3.6	4.8	6.0	0.2	1.4	2.6	3.8	5.0	6.2	0.4
0.	0.707	1.	0.707	0.	-1.	0.	0.707	1.	0.707	-0.707	-1.	0.

La scelta più brutale consiste nel troncare il risultato della fase e scegliere solo la parte intera. In questo modo si vede che l'aumento di frequenza consiste di fatto nel “saltare” di tanto in tanto un campione (nel primo ciclo il campione 5, nel secondo il campione 4 e così via). In tale modo l'approssimazione è troppo grossolana e ciò peggiora il S/N.



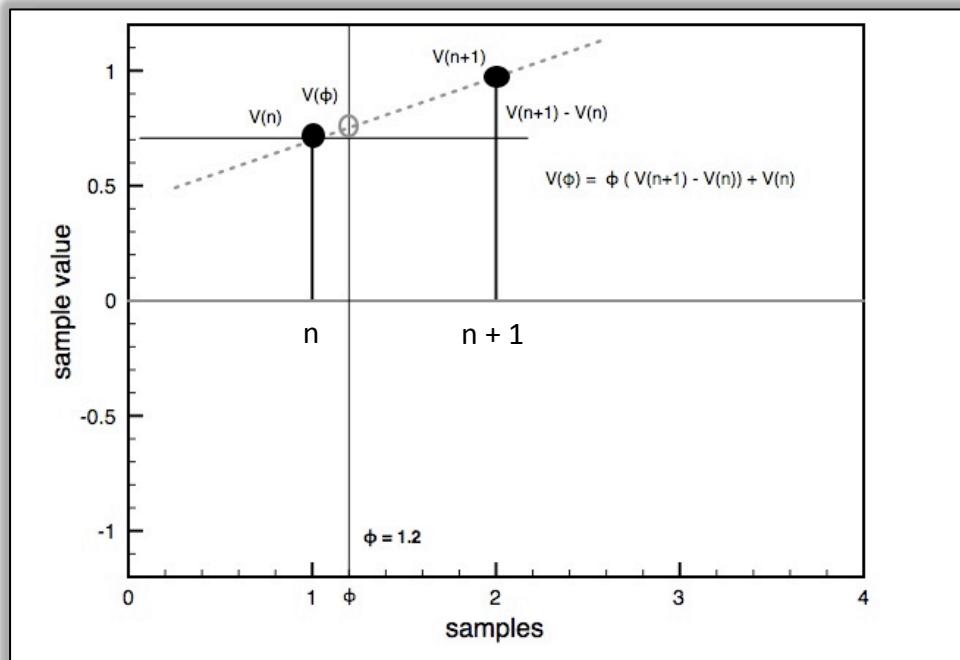
Con  $\text{Inc} < 1$  (es. 0.5) si ottiene una sequenza dove ogni campione viene letto due volte consecutive con evidente distorsione della forma d'onda risultante.

0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0
0.	0.	0.707	0.707	1.	1.	0.707	0.707	0.	0.	-0.707	-0.707	-1.

# Interpolazione lineare

Per migliorare sensibilmente la qualità della sintesi i campioni vengono riprodotti attraverso un processo d'interpolazione che al minimo grado è di tipo lineare. Il metodo consiste nel leggere due campioni  $V(n)$  e  $V(n+1)$  consecutivi e ipotizzare che tali campioni appartengano ad una retta che li congiunge. Detta  $\phi_f = \phi - \phi_n$  la parte frazionaria della fase corrente, il campione interpolato  $v(\phi_f)$  si ottiene dalla relazione:

$$V(\phi) = \phi_f [ V(n+1) - V(n) ] + V(n)$$



# Interpolazione lineare (esempio)

**Esempio.** Nella figura precedente la parte frazionaria della fase corrente è  $\phi_f = 0.2$  (essendo  $\phi = 1.2$ ). In questo caso il valore cade tra due campioni successivi, precisamente tra i campioni situati all'indirizzo 1 e 2 e i cui valori sono nell'ordine 0.707 e 1.

Applicando  $V(\phi_f) = \phi_f [ V(n+1) - V(n) ] + V(n)$  si ottiene:

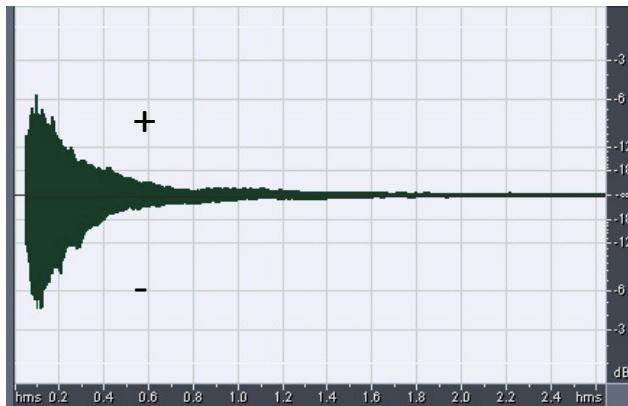
$$V(0.2) = 0.2 (1 - 0.707) + 0.707 = 0.7656$$

Come si vede il valore ottenuto è diverso (lievemente maggiore) da 0.707 e da 1 come è verosimile che sia.

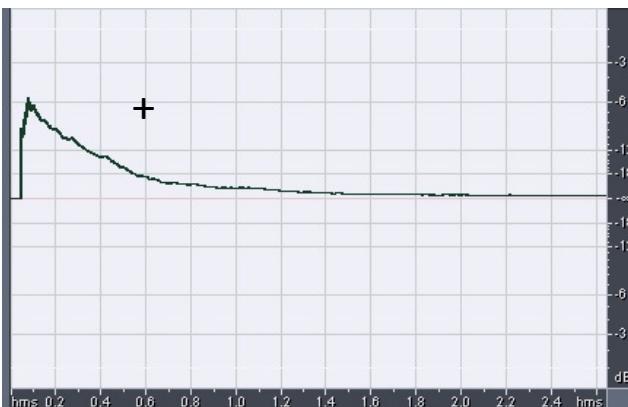
*NOTA: Risultati ancora migliori si ottengono con interpolazioni di ordine superiore (quadratico e cubico). In questo caso il tempo di calcolo del campione aumenta per effetto del maggior numero di accessi in memoria campioni e per la complessità dell'algoritmo.*

# Profilo dinamico e inviluppo

I suoni naturali, ed in particolare i suoni prodotti dagli strumenti musicali, presentano sempre un profilo dinamico (energetico e quindi di ampiezza) variabile nel tempo. I due grafici sotto rappresentano rispettivamente la forma d'onda di una nota di pianoforte (La4 440 Hz) e il suo profilo dinamico (o inviluppo) estratto con procedura matematica



Nota di pianoforte  
La4 (440 Hz)

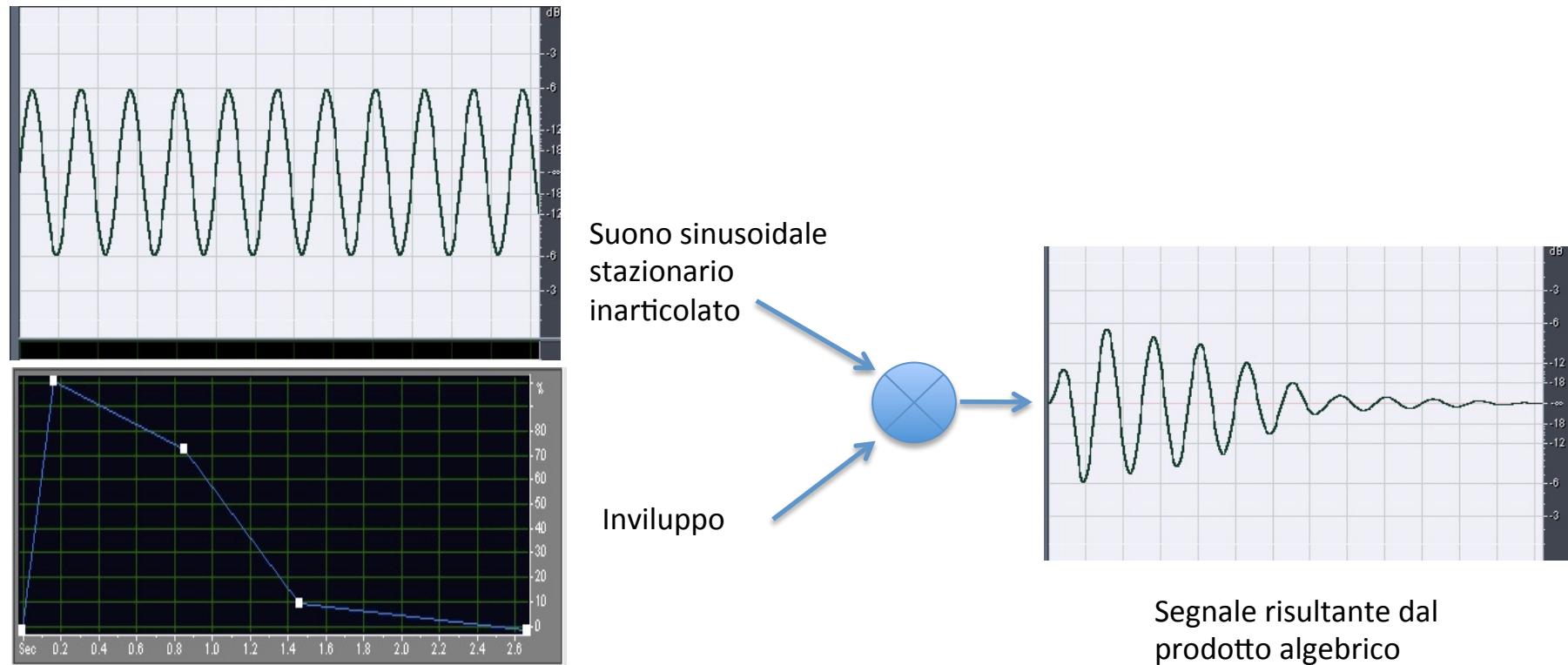


Inviluppo

Come si vede, il segnale che rappresenta la nota presenta variazioni di ampiezza sia positive che negative (bipolari) mentre il suo **inviluppo** è costituito da valori solo positivi (unipolari)

# Sintesi e Inviluppo

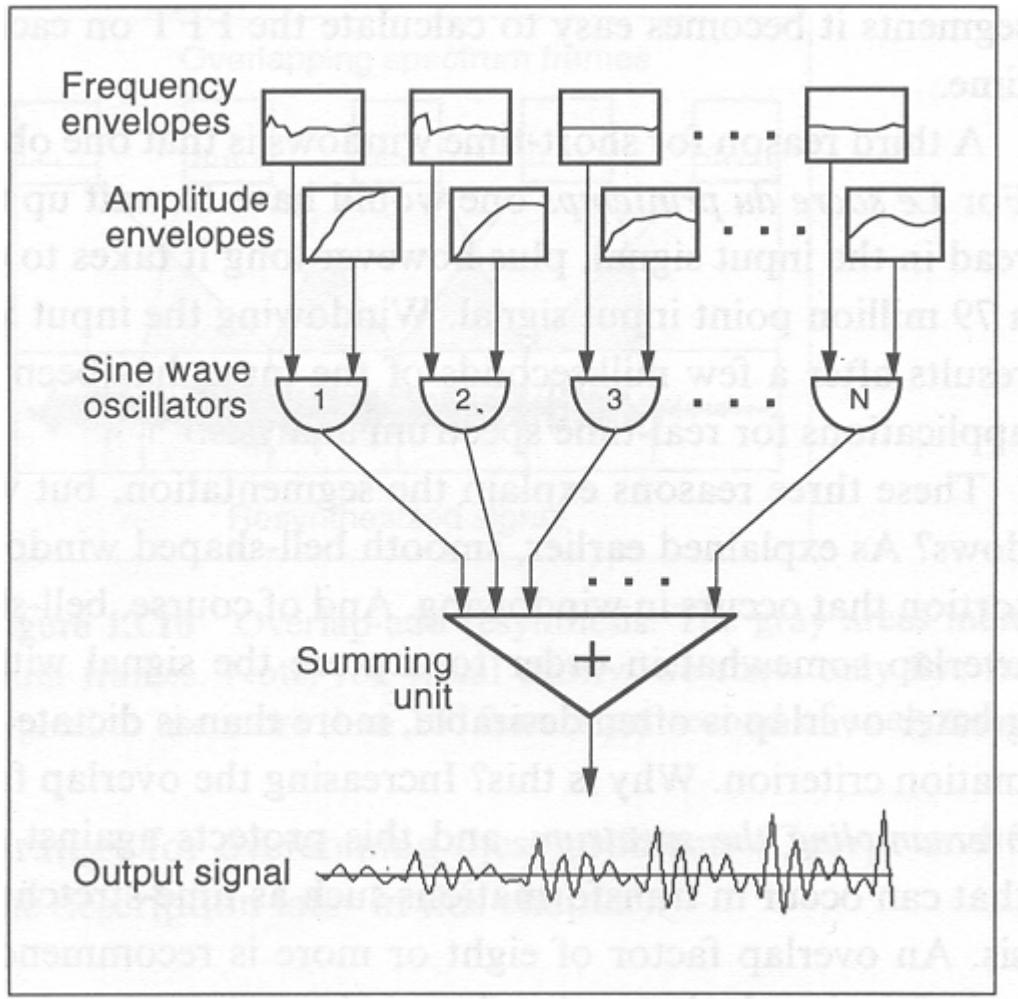
Nella sintesi del suono, il profilo dinamico viene imposto successivamente alla generazione inarticolata del suono. Da un punto di vista operativo questa operazione consiste nella moltiplicazione algebrica tra un segnale audio e un segnale di controllo.



# Tecniche di Sintesi Audio

- Le tecniche di sintesi audio si riferiscono al processo di generazione digitale del suono attraverso algoritmi la cui gestione dipenda da un insieme di variabili di controllo che possono essere gestite individualmente, strutturalmente, manualmente o automaticamente.
- Ogni tipologia di sintesi si basa su un modello matematico che ne definisce il funzionamento e la gestione.
- Le tecniche di sintesi possono suddividersi principalmente in **Sintesi Lineari e Non Lineari**. Nella prima le componenti elementari sono predefinite a priori mentre nella seconda il numero e la loro relazione dipende fortemente dai parametri di controllo.
- Ad esempio la sintesi additiva e sottrattiva sono sintesi lineari poiché si conosce in anticipo tutte le componenti mentre la FM (Modulazione di frequenza) e Waveshaping (Distorsione non Lineare) appartengono alla seconda categoria.
- Le varie sintesi possono avvenire in **tempo differito** o in **tempo reale**

# Sintesi Additiva

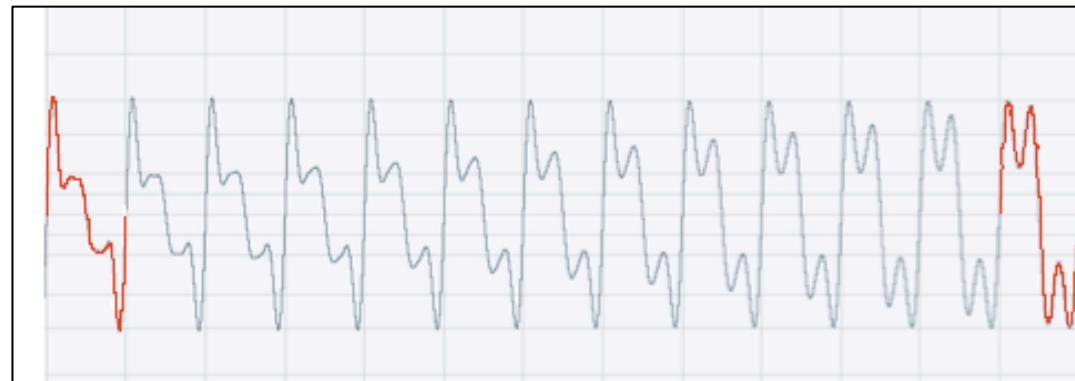


Consiste nel produrre suoni attraverso la somma di un numero rilevante di componenti sinusoidali controllate in **ampiezza** e **frequenza**.

Se i rapporti di frequenza sono di tipo armonico la sintesi produrrà forme d'onda stazionarie e con pitch identificabile.

# Sintesi a forma d'onda fissa

- La sintesi a forma d'onda fissa è un tipo di sintesi molto semplificato che utilizza forme d'onda periodizzate e memorizzate in tabelle di memoria.
- Nella forma più elementare il suono prodotto è assolutamente periodico e le forme d'onda possono o essere pre-calcolate matematicamente o eventualmente estrapolate da un segnale campionato in cui si utilizza un singolo periodo.
- Un migliore risultato si può ottenere attraverso una trasformazione graduale tra diverse forme d'onda (cross-fade) a due a due (vedi figura).



Forma d'onda iniziale

X-fade

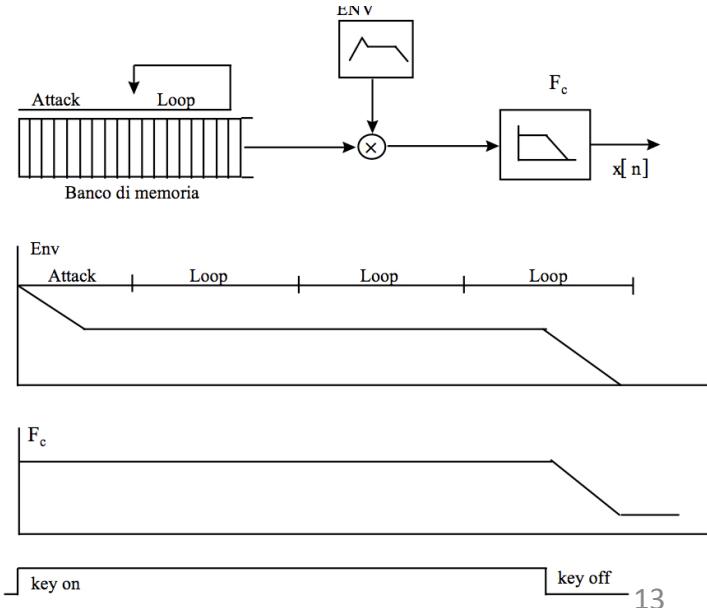
Forma d'onda finale

# Sintesi per campionamento (PCM)

Le tecniche PCM fanno ricorso a tabelle di ricerca (look-up) nelle quali sono stati precedentemente memorizzati i campioni dei suoni che si vogliono riprodurre. Tali tecniche non costituiscono dei veri metodi di sintesi, in quanto non fanno ricorso ad un sistema di calcolo per la generazione del segnale. Tuttavia, i risultati che si ottengono in termini di fedeltà nella riproduzione dei suoni di strumenti tradizionali d'orchestra sono superiori a quelli ottenuti con le tecniche propriamente di sintesi.

Un problema legato alla riproduzione dei suoni PCM è dovuto al fatto che in molti casi si vuole scindere la durata di un suono presente in memoria da quella imposta dall'esecutore in fase di riproduzione. Il modo per ottenere una durata indefinita per il suono consiste nel leggere ripetutamente una parte della forma d'onda memorizzata. Il banco di memoria che contiene i relativi campioni viene diviso in tre segmenti: -

1. segmento che viene riprodotto alla fase di attack ( o PCM) –
2. segmento che viene riprodotto alla fase di sustain (Loop) –
3. segmento che viene riprodotto alla fase di release. (Release PCM)



# Modulazione AM

In generale il processo di modulazione è esprimibile matematicamente nel seguente modo: supponendo  $v_m(t)$  e  $v_c(t)$  due segnali cosinusoidali del tipo

$$a_m(t) = A_m \cos \omega_m t \quad \text{e} \quad a_c(t) = A_c \cos \omega_c t \quad \text{con } \omega_c \gg \omega_m \text{ si ha:}$$

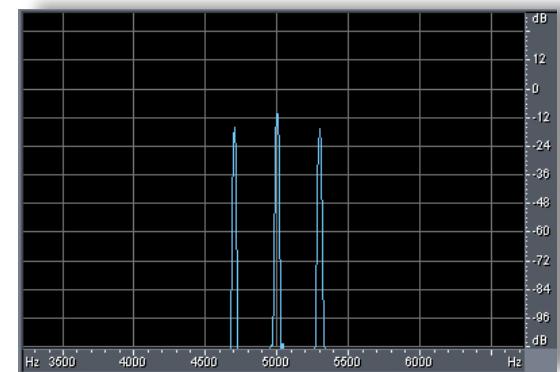
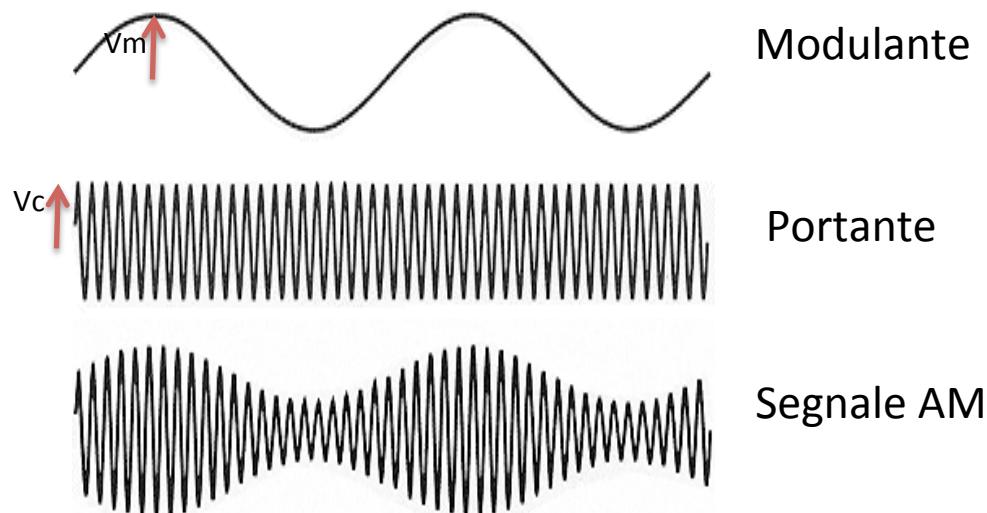
$$\begin{aligned} a(t) &= [(A_c + K_a A_m \cos \omega_m t) \cos \omega_c t] \\ a(t) &= A_c [(1 + K_a A_m / A_c \cos \omega_m t) \cos \omega_c t] \end{aligned}$$

dove  $v(t) = \text{segnale AM}$

$V_c$  = ampiezza di picco del segnale portante

$V_m$  = " " " modulante

$K_a A_m / A_c = m$  = indice di modulazione d'ampiezza



Spettro AM –

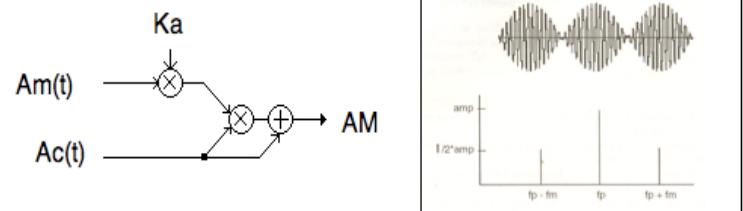
Si noti la presenza della portante oltre alle righe laterali.

L'ampiezza delle righe (bande) laterali vale  $m A_c / 2$

# Modulazioni AM-RM

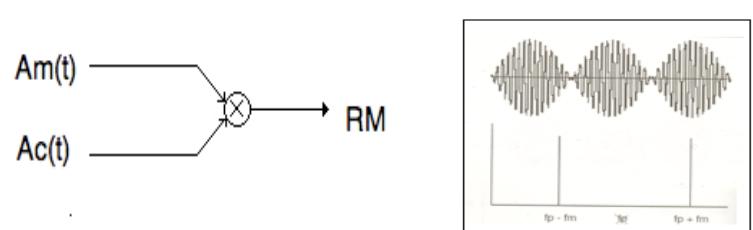
La AM (*Modulazione d'Aampiezza*) e la RM (*Ring-Modulazione o Modulazione ad Anello*, un caso particolare di AM) appartengono alla categoria delle sintesi non-lineari assieme alla FM (*Modulazione di Frequenza*) e alla WAVESHAPING (*Distorsione Non Lineare*).

Nella AM il suono si ottiene attraverso una combinazione tra somma e prodotto tra due segnali. Per semplicità supponiamo che i due segnali siano entrambi sinusoidali di frequenza rispettivamente fm e fc (freq. Modulante e freq. Portante).



AM

Nella RM (più utilizzata nella sintesi) il suono si ottiene attraverso un semplice prodotto tra due segnali. Per semplicità supponiamo che i due segnali siano entrambi sinusoidali di frequenza rispettivamente fm e fc (freq. Modulante e freq. Portante).



RM

# Modulazione RM

Nella modulazione RM, supponendo sempre  $v_m(t)$  e  $v_c(t)$  due segnali cosinusoidali del tipo

$$a_m(t) = A_m \cos \omega_m t \quad \text{e} \quad a_c(t) = A_c \cos \omega_c t \quad \text{con } \omega_c \gg \omega_m \text{ si ha:}$$

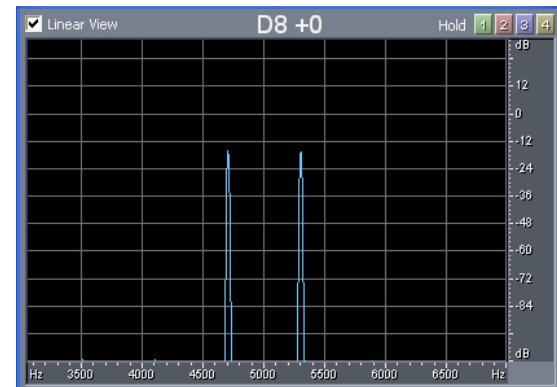
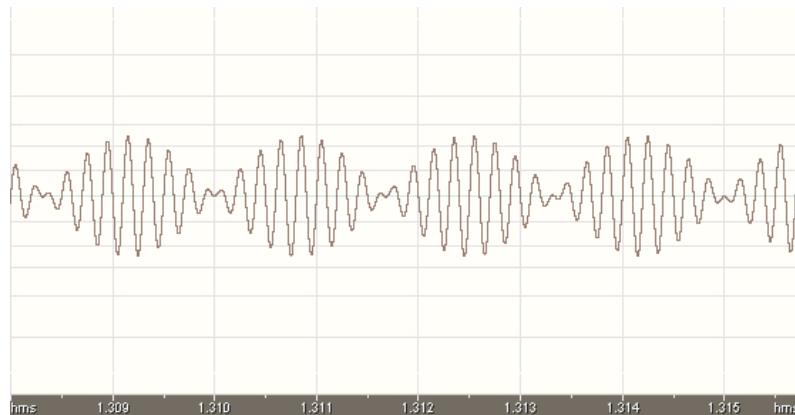
Segnale RM  $\rightarrow a(t) = (A_m \cos \omega_m t) (A_c \cos \omega_c t)$

dove  $v(t) = \text{segnale AM}$

$V_c$  = ampiezza di picco del segnale portante

$V_m$  = " " " modulante

L'indice di modulazione è semplicemente un fattore di scala.



Spettro RM –  
Si noti l'assenza della portante.

L'ampiezza delle righe (bande) laterali  
vale sempre  $m A_c / 2$

# Modulazione RM di segnali non sinusoidali

Nel caso in cui il segnale modulante sia una forma d'onda non sinusoidale (ad esempio una somma di 4 sinusoidi come mostrato nelle figure) la relazione somma/differenza si instaura tra ogni componente della modulante con la portante.



Spettro di un segnale RM con portante sinusoidale a 5 KHz sinusoidale e modulante costituito da un segnale complesso (4 componenti armoniche) con freq. fondamentale a 300 Hz. Nel grafico di sinistra sono mostrati insieme lo spettro del modulante e della portante. Nel grafico di destra, il segnale ring-modulato. Si noti la traslazione dello spettro e la duplicazione simmetrica

# Proprietà dei segnali RM

Nel caso in esame le frequenze del modulante sono:

$F_m = 300$ ,  $2F_m = 600$ ,  $3F_m = 900$ ,  $4F_m = 1200$  Hz mentre la portante  $F_c$  vale 5 KHz.

Dopo la modulazione lo spettro risulta essere composto da:

Banda laterale inferiore:  $F_c - F_1 = 4700$

$$F_c - 2F_1 = 4400$$

$$F_c - 3F_1 = 4100$$

$$F_c - 4F_1 = 3800$$

Banda laterale superiore:  $F_c + F_1 = 5300$

$$F_c + 2F_1 = 5600$$

$$F_c + 3F_1 = 5900$$

$$F_c + 4F_1 = 6200$$

Data la particolare sequenza di frequenze risulterà uno spettro a partire da un ordine elevato di armonica. In questo caso si tratta di un segnale la cui fondamentale vale 100 Hz e in cui la prima armonica che compare è la numero 38 ( $3800/100$ ). Il risultato acustico sarà sicuramente molto simile ad un suono totalmente inarmonico.

# Frequenze riflesse

Generalizzando il caso appena visto si può dire che nella sintesi RM si generano sempre somma e differenza tra la portante e il modulante. Nel caso di entrambe le onde di tipo sinusoidale si ha una sola coppia  $F_{i,s}$ : (inferiore, superiore)

$$F_{i,s} = F_c \pm F_m$$

Se  $F_c > F_m$  la frequenza  $f_i$  rimane  $> 0$ .

Se  $F_c < F_m$  la frequenza  $f_i$  diventa negativa e in questo caso viene riflessa simmetricamente attorno al valore 0.

Es: Se  $F_c = 500$  e  $F_m = 600$  si avrà:

$$f_i = 500 - 600 = -100 \rightarrow 100 \text{ Hz}$$

$$f_s = 500 + 600 = 1100$$

*Quindi tutte le frequenze che diventano negative verranno riflesse intorno all'origine. Nel caso di spettro complesso del modulante le righe riflesse possono cadere sulle posizioni di quelle non riflesse andando a modificare la simmetria dello spettro risultante.*

# Osservazioni su AM/RM

- La modulazione AM/RM, al di là delle lievi differenze evidenziate nell'un caso o nell'altro, può essere vista come un tecnica applicabile sia a segnali sintetici o a segnali concreti.
- In generale, questo tipo di modulazione produce due diversi effetti a seconda che la velocità della modulazione sia elevata o no.
  - a) Velocità di modulazione bassa : produce effetti nell'ambito temporale (tremolo)
  - b) Velocità di modulazione più elevata ( $> 16$  Hz) : produce effetti nell'ambito frequenziale-timbrico
- Per definizione, l'ampiezza del segnale modulato varia nel tempo

# Modulazione di frequenza (FM)

Nella modulazione di ampiezza, l'ampiezza (istantanea) del segnale modulato è proporzionale all'ampiezza della modulante.

Nelle modulazioni di frequenza e di fase, l'ampiezza del segnale modulato è costante; variano o la frequenza (istantanea) o la fase (istantanea) del segnale modulato. In entrambi i casi, la modulante cambia l'argomento (cioè l'angolo) della sinusoide portante.

Modulazione di frequenza:

$$y(t) = \cos(2\pi f(t)t), \text{ con } f(t) = F(m(t))$$

Modulazione di fase:

$$y(t) = \cos(2\pi f_c t + \varphi(t)), \text{ con } \varphi(t) = \Phi(m(t))$$

# Modulazione di fase (PM) (1/2)

---

Il caso più semplice è la modulazione di fase (**PM = phase modulation**), in cui la fase è proporzionale all'ampiezza istantanea della modulante:

$$\Phi(m(t)) = k_\varphi m(t)$$

Il segnale modulato  $\cos(2\pi f_c t + k_\varphi m(t))$  è anticipato o ritardato rispetto alla portante.

$\Phi(m(t))$  è la deviazione istantanea di fase. La deviazione massima di fase è il massimo valore assoluto che  $\Phi(m(t))$  assume al variare di  $t$ .

La differenza non è solo formale poiché nell'implementazione reale dei sistemi di sintesi la FM è sempre una PM poiché permette una migliore gestione dei parametri ed in particolare consente l'automodulazione. Da un punto di vista concettuale è più facile pensare in termini di FM.

# Modulazione di fase e di frequenza

---

$$f(t) = f_c + \frac{k_\Phi}{2\pi} \frac{d}{dt} m(t)$$

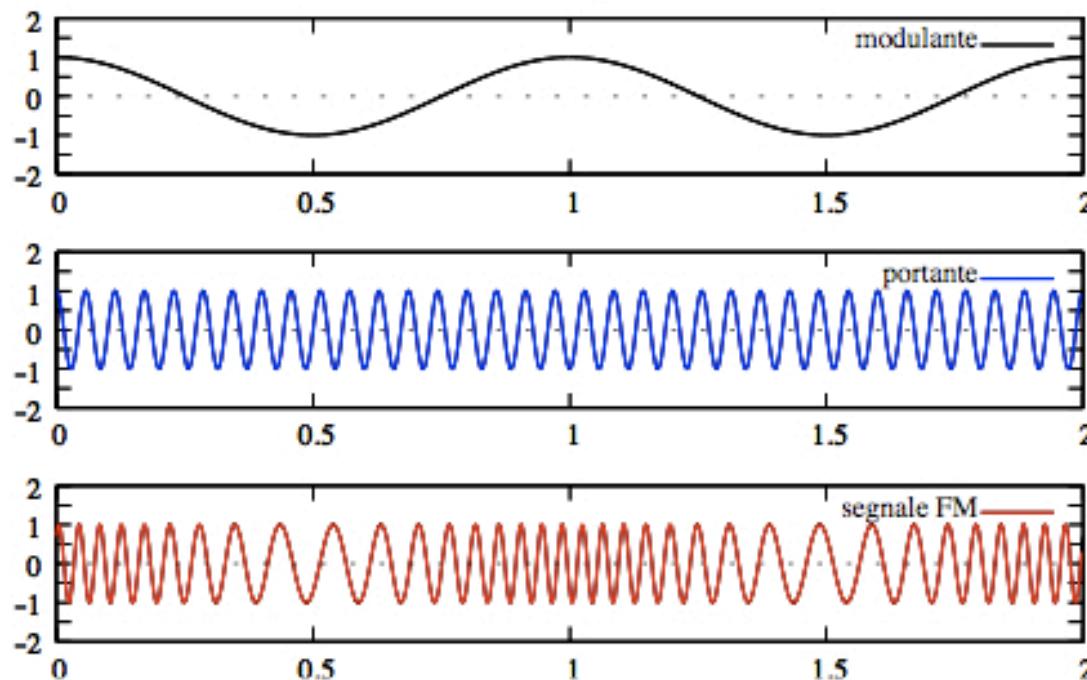
Una variazione istantanea della fase comporta anche una variazione della frequenza, che è la derivata della fase.

- È impossibile modulare la fase senza modulare contemporaneamente anche la frequenza, e viceversa.
- La modulazione di fase con  $m(t)$  è equivalente alla modulazione di frequenza con  $\frac{d}{dt}m(t)$ .
- La modulazione di frequenza con  $m(t)$  è equivalente alla modulazione di fase con  $\int m(t) dt$ .

La differenza non è solo formale poiché nell'implementazione reale dei sistemi di sintesi la FM è sempre una PM poiché permette una migliore gestione dei parametri ed in particolare consente l'automodulazione. Da un punto di vista concettuale è più facile pensare in termini di FM.

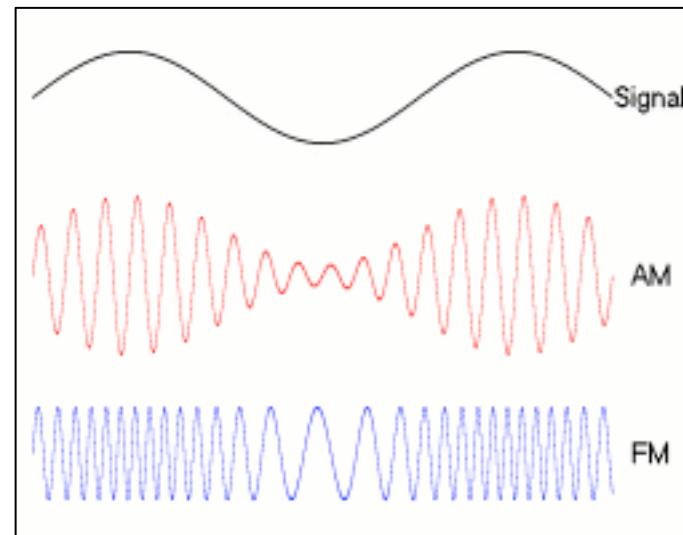
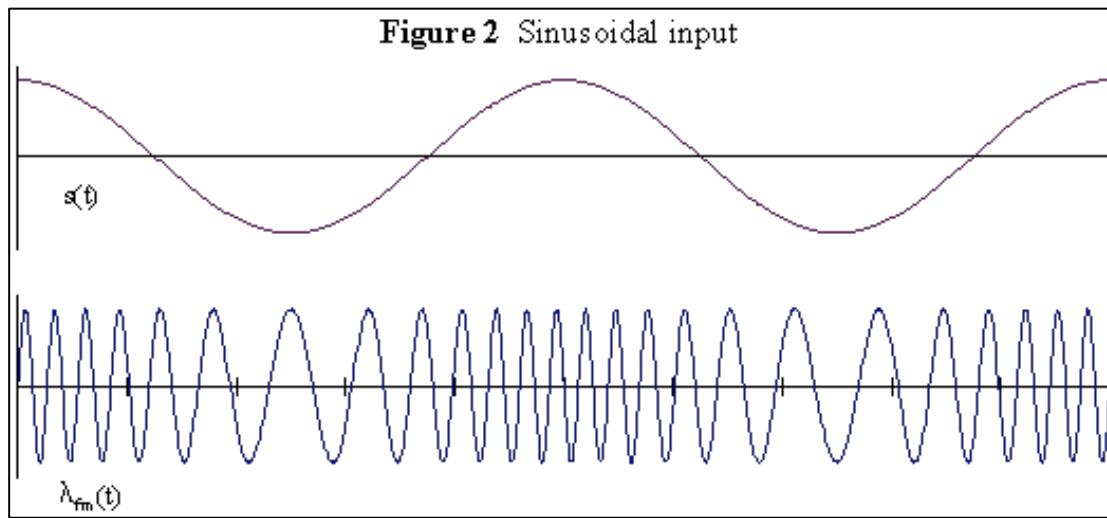
# Modulazione di frequenza (FM)

modulante:  $m(t) = \cos 2\pi f_1 t$ ; portante:  $p(t) = \cos 2\pi f_c t$   
segnale FM:  $y(t) = \cos(2\pi f_c t + k_\varphi \sin 2\pi f_1 t)$



Quando  $m(t) > 0$  gli attraversamenti dello zero di  $y(t)$  sono più frequenti; quando  $m(t) < 0$  gli attraversamenti dello zero di  $y(t)$  sono meno frequenti.

# Il segnale FM vs AM



AM vs FM

# Indice di modulazione

Per una modulante sinusoidale  $m(t) = \cos 2\pi f_1 t$ , il segnale modulato  $y(t) = \cos(2\pi f_c t + k_\varphi \sin 2\pi f_1 t)$  ha una frequenza istantanea

$$f(t) = f_c + \frac{1}{2\pi} \frac{d}{dt} (k_\varphi \sin 2\pi f_1 t) = f_c + k_\varphi f_1 \cos 2\pi f_1 t$$

La deviazione istantanea di frequenza  $\Delta f$  del segnale FM è:

$$\Delta f(t) = f(t) - f_c = k_\varphi f_1 \cos 2\pi f_1 t$$

La deviazione massima di frequenza  $\Delta f_{\max}$  è:

$$\Delta f_{\max} = k_\varphi f_1$$

e  $k_\varphi$  è detto indice di modulazione.

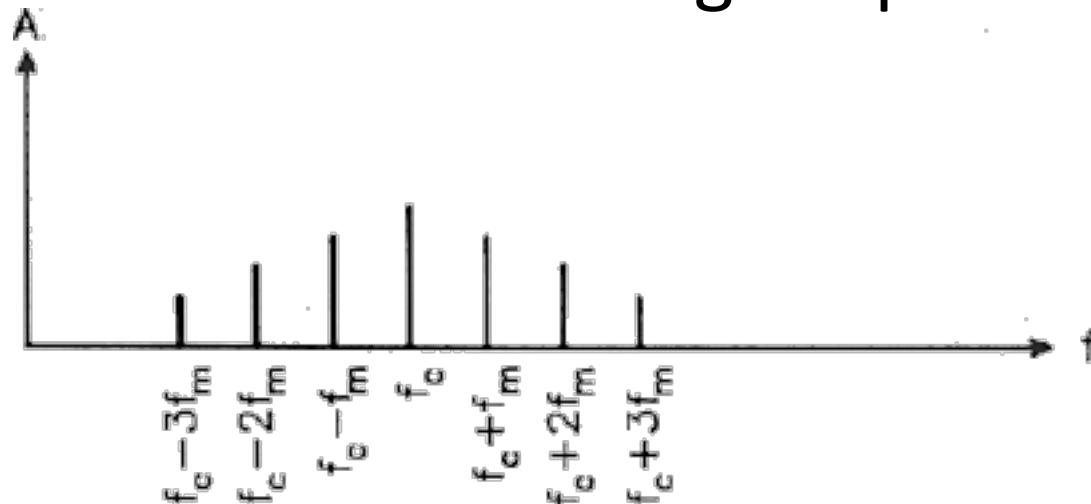
# Spettro del segnale FM

Lo spettro del segnale FM non può essere evidentemente calcolato in maniera elementare come nella AM/RM poiché in questo caso l'argomento di una funzione trigonometrica è a sua volta una funzione trigonometrica. Si deve ricorrere all'impiego di un artificio matematico che consiste nell'utilizzare uno sviluppo in serie utilizzando le funzioni di Bessel del primo tipo (J).

Omettendo per brevità i passaggi intermedi, si perviene alla seguente formula finale tenendo conto che  $f_1$  = frequenza modulante:

$$\begin{aligned}y(t) = & J_0(k_\varphi) \cos 2\pi f_c t \\& - J_1(k_\varphi)(\cos 2\pi(f_c - f_1)t - \cos 2\pi(f_c + f_1)t) \\& + J_2(k_\varphi)((\cos 2\pi(f_c - 2f_1)t + \cos 2\pi(f_c + 2f_1)t) \\& - J_3(k_\varphi)((\cos 2\pi(f_c - 3f_1)t - \cos 2\pi(f_c + 3f_1)t) \\& + \dots\end{aligned}$$

# Posizione delle righe spettrali

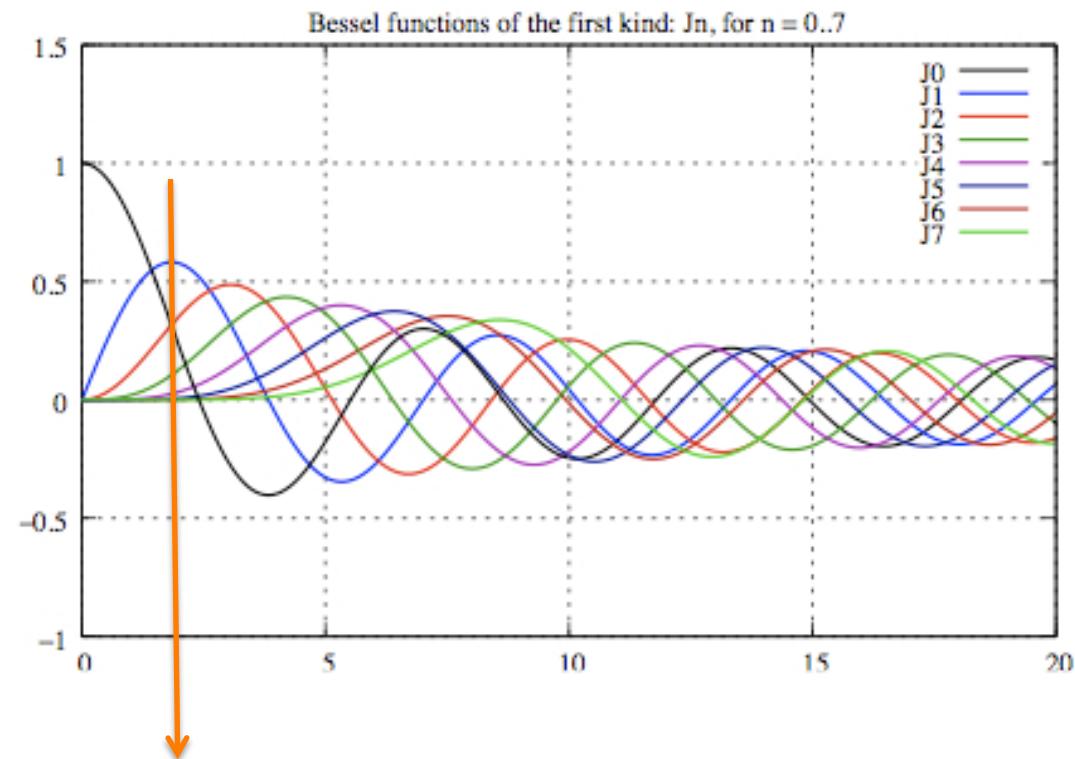


n	Inferiore		Superiore	
	Freq	Amp	Freq	Amp
1	$f_c - f_m$	$-J_1(I)$	$f_c + f_m$	$J_1(I)$
2	$f_c - 2f_m$	$J_2(I)$	$f_c + 2f_m$	$J_2(I)$
3	$f_c - 3f_m$	$-J_3(I)$	$f_c + 3f_m$	$J_3(I)$
4	$f_c - 4f_m$	$J_4(I)$	$f_c + 4f_m$	$J_4(I)$
5	$f_c - 5f_m$	$-J_5(I)$	$f_c + 5f_m$	$J_5(I)$

# Funzioni di Bessel

Le  $J_n(x)$  sono le Funzioni di Bessel (del primo tipo), e costituiscono le soluzioni dell'equazione di Bessel:

$$x^2 \frac{d^2y}{dx^2} + x \frac{dy}{dx} + (x^2 - n^2)y = 0$$



Le ampiezze delle componenti del segnale FM sono date dai valori delle funzioni di Bessel che intercettano un valore particolare dell'indice di modulazione. ( Nell' esempio indicato con la freccia nel circa a 2.0)

# Schema del calcolo dei segni delle bande laterali FM

n	Inferiore		Superiore	
	Freq	Amp	Freq	Amp
1	$f_c - f_m$	$- J_1(l)$	$f_c + f_m$	$J_1(l)$
2	$f_c - 2f_m$	$J_2(l)$	$f_c + 2f_m$	$J_2(l)$
3	$f_c - 3f_m$	$- J_3(l)$	$f_c + 3f_m$	$J_3(l)$
4	$f_c - 4f_m$	$J_4(l)$	$f_c + 4f_m$	$J_4(l)$
5	$f_c - 5f_m$	$- J_5(l)$	$f_c + 5f_m$	$J_5(l)$
:	:	:	:	:

# Calcolo dello spettro FM (1/3)

Si deve ricavare lo spettro di un segnale FM ottenuto con i seguenti dati di progetto:

$$F_c = 1000 \text{ Hz}$$

$$F_m = 300 \text{ Hz}$$

$$m_i = 2$$

limitando lo studio fino alla 6° coppia utilizzando la seguente tabulazione delle  $J_n(x)$

Funzioni di Bessel tabulate per alcuni valori di  $x$  (o  $m_i$ , indice di modulazione)

$x$ ( $m_i$ )	(CARRIER)	# OR ORDER															
		$J_0$	$J_1$	$J_2$	$J_3$	$J_4$	$J_5$	$J_6$	$J_7$	$J_8$	$J_9$	$J_{10}$	$J_{11}$	$J_{12}$	$J_{13}$	$J_{14}$	$J_{15}$
0.00	1.00	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0.25	0.98	0.12	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0.5	0.94	0.24	0.03	—	—	—	—	—	—	—	—	—	—	—	—	—	—
1.0	0.77	0.44	0.11	0.02	—	—	—	—	—	—	—	—	—	—	—	—	—
1.5	0.51	0.56	0.23	0.06	0.01	—	—	—	—	—	—	—	—	—	—	—	—
2.0	0.22	0.58	0.35	0.13	0.03	—	—	—	—	—	—	—	—	—	—	—	—
2.5	-0.05	0.50	0.45	0.22	0.07	0.02	—	—	—	—	—	—	—	—	—	—	—
3.0	-0.26	0.34	0.49	0.31	0.13	0.04	0.01	—	—	—	—	—	—	—	—	—	—
4.0	-0.40	-0.07	0.36	0.43	0.28	0.13	0.05	0.02	—	—	—	—	—	—	—	—	—
5.0	-0.18	-0.33	0.05	0.36	0.39	0.26	0.13	0.05	0.02	—	—	—	—	—	—	—	—
6.0	0.15	-0.28	-0.24	0.11	0.36	0.36	0.25	0.13	0.06	0.02	—	—	—	—	—	—	—
7.0	0.30	0.00	-0.30	-0.17	0.16	0.35	0.34	0.23	0.13	0.06	0.02	—	—	—	—	—	—
8.0	0.17	0.23	-0.11	-0.29	-0.10	0.19	0.34	0.32	0.22	0.13	0.06	0.03	—	—	—	—	—
9.0	-0.09	0.24	0.14	-0.18	-0.27	-0.06	0.20	0.33	0.30	0.21	0.12	0.06	0.03	0.01	—	—	—
10.0	-0.25	0.04	0.25	0.06	-0.22	-0.23	-0.01	0.22	0.31	0.29	0.20	0.12	0.06	0.03	0.01	—	—
12.0	0.05	-0.22	-0.08	0.20	0.18	-0.07	-0.24	-0.17	0.05	0.23	0.30	0.27	0.20	0.12	0.07	0.03	0.01
15.0	-0.01	0.21	0.04	-0.19	-0.12	0.13	0.21	0.03	-0.17	-0.22	-0.09	0.10	0.24	0.28	0.25	0.18	0.12

## Calcolo dello spettro FM (2/3)

Le due frequenze (portante e modulante) formano un rapporto C:M = 10:3.

Le righe laterali del processo si possono trovare allora nel modo seguente:

C	1000 Hz	→ J0
C ± M	700 1300	→ J1
C ± 2 M	400 1600	→ J2
C ± 3 M	100 1900	→ J3
C ± 4 M	-200 2200	→ J4
Etc...		

(m)	$J_0$	$J_1$	$J_2$	$J_3$	$J_4$	$J_5$	$J_6$
0.00	1.00	—	—	—	—	—	—
0.25	0.98	0.12	—	—	—	—	—
0.5	0.94	0.24	0.03	—	—	—	—
1.0	0.77	0.44	0.11	0.02	—	—	—
1.5	0.51	0.56	0.23	0.06	0.01	—	—
2.0	0.22	0.58	0.35	0.13	0.03	—	—
2.5	-0.05	0.50	0.45	0.22	0.07	0.02	—

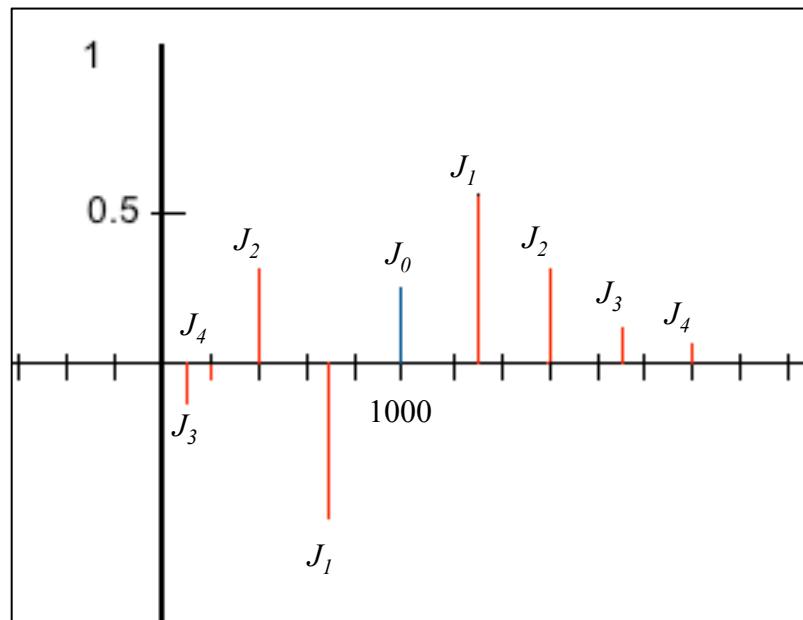
Per le ampiezze si procede nel modo seguente:

- 1) Per n = 0 abbiamo la sola portante (J0) e possiamo andare a leggere il valore direttamente sulla tabella di Bessel e vale 0.22
- 2) Per n=1 abbiamo la prima coppia di bande laterali e J1 vale 0.58. A questo punto guardiamo la tabella dei segni : la riga inferiore è negativa e quella superiore positiva quindi l'ampiezza sarà -0.58 per la frequenza 700 Hz e +0.58 per la frequenza a 1300 Hz.
- 3) Per n = 2 abbiamo la seconda coppia e J2 vale 0.35. Dalla tabella dei segni entrambe le righe sono positive quindi le ampiezze saranno 0.35 per entrambe le frequenze 400 Hz e 1600 Hz
- 4) Si procede fino ad esaurimento delle coppie (in questo caso il limite è J4 perché per valori superiori le ampiezze sono trascurabili).

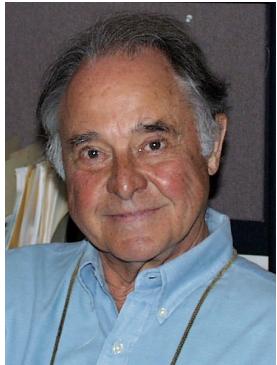
Tenendo conto della riflessione delle frequenze negative si ottiene il seguente risultato:

## Calcolo dello spettro FM (3/3)

C            1000 Hz     $\rightarrow J_0 \rightarrow 0.22$   
C ± M      700 1300     $\rightarrow J_1 \rightarrow -0.58 / +0.58$   
C ± 2 M     400 1600     $\rightarrow J_2 \rightarrow +0.35 / +$   
0.35  
C ± 3 M     100 1900     $\rightarrow J_3 \rightarrow -0.13 / +0.13$   
C ± 4 M     -200 2200    $\rightarrow J_4 \rightarrow -0.03 / +0.03$



# FM applicata alla sintesi audio



J. Chowning

Mentre la RM/AM è generalmente utilizzata come tecnica per modificare suoni preesistenti, la FM viene impiegata come processo di sintesi diretta del suono. L'applicazione della FM alla generazione audio si deve a John Chowning, ricercatore, musicista statunitense che al CCRMA di Stanford ha formulato l'algoritmo, utilizzato in seguito dalla Yamaha Corporation per produrre una serie di strumenti – la serie DX – che hanno segnato la storia della liuteria digitale negli anni '80.



Il DX7 Yamaha è stato il primo sintetizzatore interamente digitale in commercio, basato sulla Sintesi FM (l'implementazione è in effetti una modulazione di fase e non di frequenza) commercializzato tra il 1982 ed il 1986, con una versione successiva nota come DX7II, dotata in una versione di floppy, tra il 1987 ed il 1989. La versione da rack dello strumento è denominata TX802.

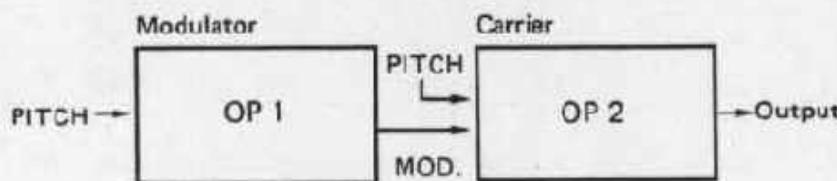
# FM e DX-7

## • Basic Operator Functions

### 1) Relationship of Carrier to Modulator

An operator can be used as either a carrier or modulator. These two basic operator functions are the basis for the FM tone generation system. Two operators can be combined in two different ways.

#### 1. Modulator and carrier combinations



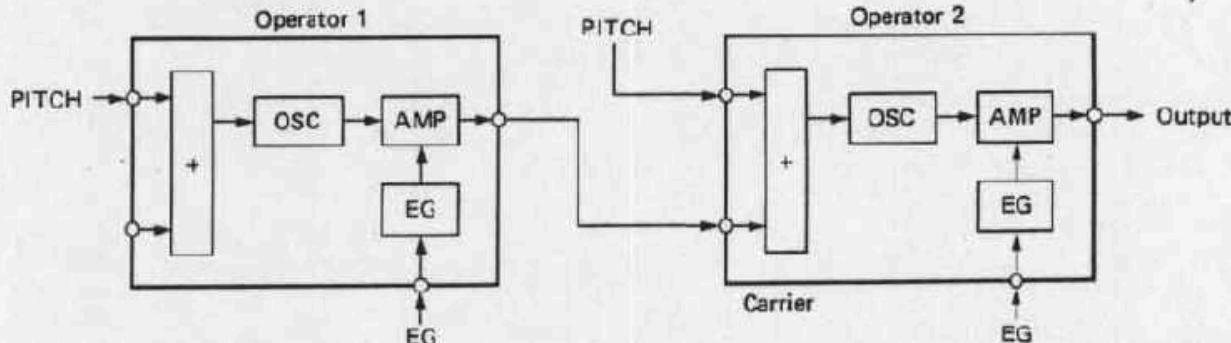
### 3) Modulator and Carrier

In the modulator/carrier configuration using two operators, shown in the figure, the operator on the left is the modulator and the operator on the right is the carrier. In the FM system, the last operator in a chain of two or more oper-

Nel DX7 Yamaha la generazione è ottenuta dall'interazione e connessione tra moduli (operatori, in pratica oscillatori digitali con controllo dell'inviluppo). A sinistra si vede la connessione più semplice: un operatore modulante che controlla un operatore portante.

Sotto è visibile la struttura interna di tali operatori

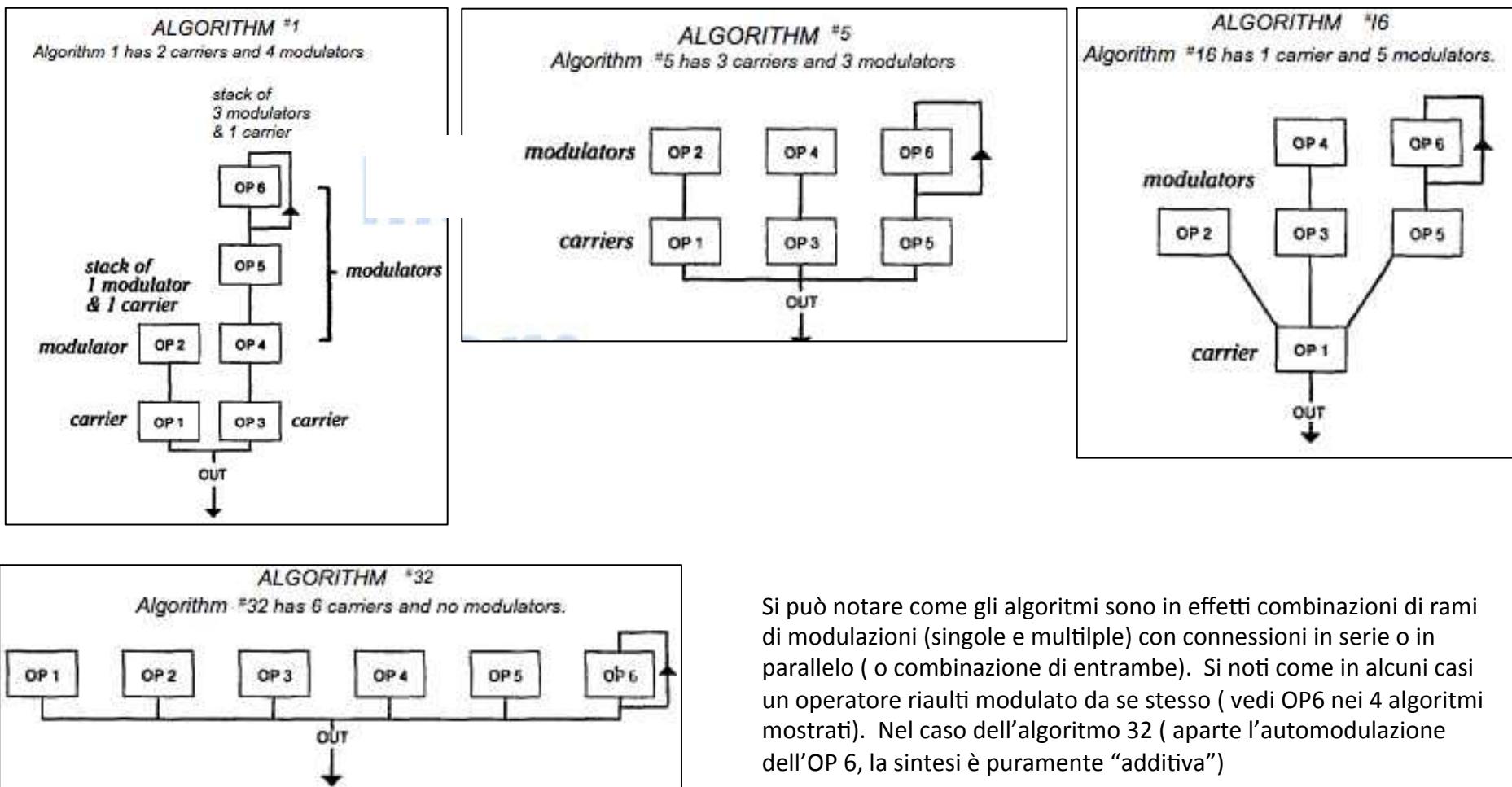
#### • Modulator and carrier combinations



ators is the carrier. By varying the ratio of the modulator and carrier frequencies, and by varying the envelope of the modulator, an extremely broad range of highly complex waveforms (complex harmonic structure) can be created.

# Operatori e algoritmi DX-7

Nel DX7 sono disponibili per la sintesi 6 operatori che interconnessi secondo diverse modalità (32 in totale) costituiscono la struttura base della sintesi. Ciascuna delle 32 combinazioni è detta “algoritmo”. Di seguito sono riportati alcuni degli algoritmi disponibili



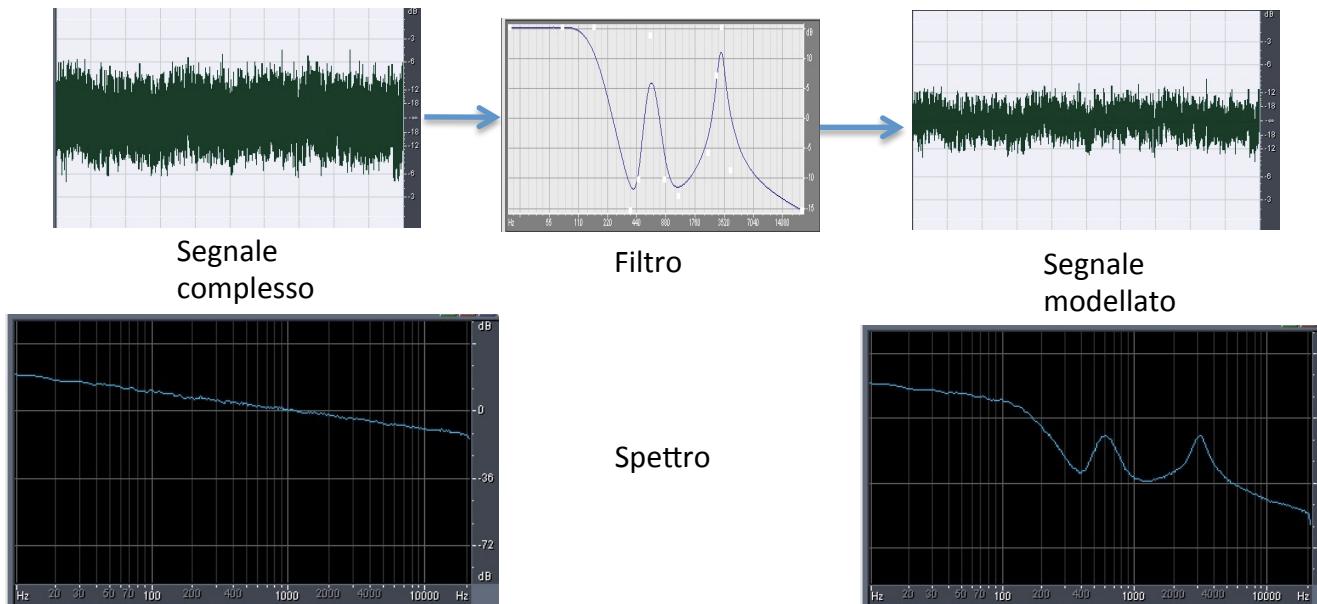
Si può notare come gli algoritmi sono in effetti combinazioni di rami di modulazioni (singole e multiple) con connessioni in serie o in parallelo (o combinazione di entrambe). Si noti come in alcuni casi un operatore sia modulato da se stesso (vedi OP6 nei 4 algoritmi mostrati). Nel caso dell'algoritmo 32 (a parte l'automodulazione dell'OP 6, la sintesi è puramente “additiva”)

# Sintesi Sottrattiva

La Sintesi Sottrattiva si basa sul principio che ogni suono viene generato modellando lo spettro di un segnale complesso. In altre parole, da un suono molto ricco spettralmente, per sottrazione, si arriva a un suono finale che abbia le caratteristiche timbriche desiderate. Il suono di origine può essere sia un suono reale pre-registrato o eventualmente acquisito in tempo reale, o un suono totalmente sintetico.

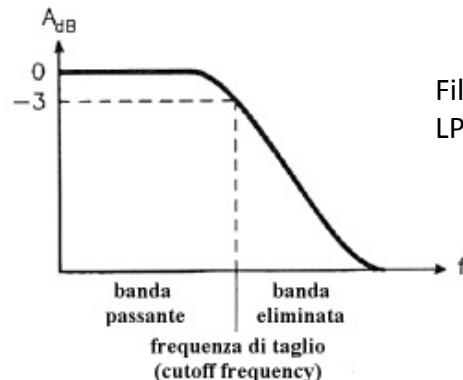
La sintesi sottrattiva si applica indistintamente ai suoni periodici e aperiodici, continui e impulsivi.

Il processo di modellazione per sottrazione, o più genericamente per attenuazione o esaltazione, si ottiene attraverso l'uso di filtri.

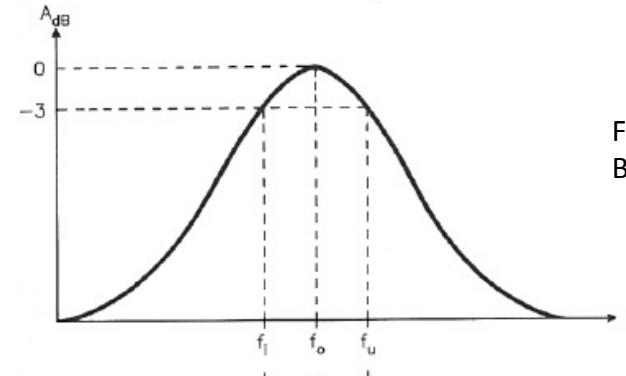


# Tipologie base dei filtri

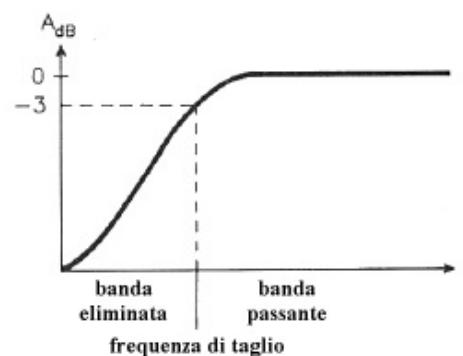
La sintesi sottrattiva digitale trae origine dall'omologa nel campo analogico dal quale ha ereditato alcune tipologie base di filtri con la possibilità di ampliare a dismisura le possibili forme di modellazione dello spettro. In ogni caso le tipologie basilari rappresentano un punto imprescindibile anche nei sistemi digitali. Tali tipologie base sono:



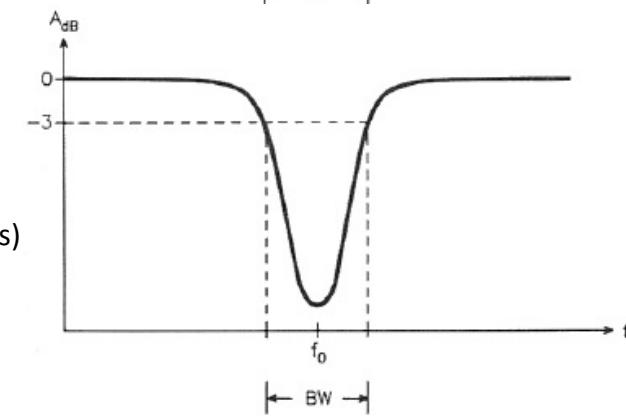
Filtro Passa-Basso  
LP Filter (Low Pass)



Filtro Passa-Banda  
BP Filter (Band Pass)



Filtro Passa-Alto  
HP Filter (High Pass)

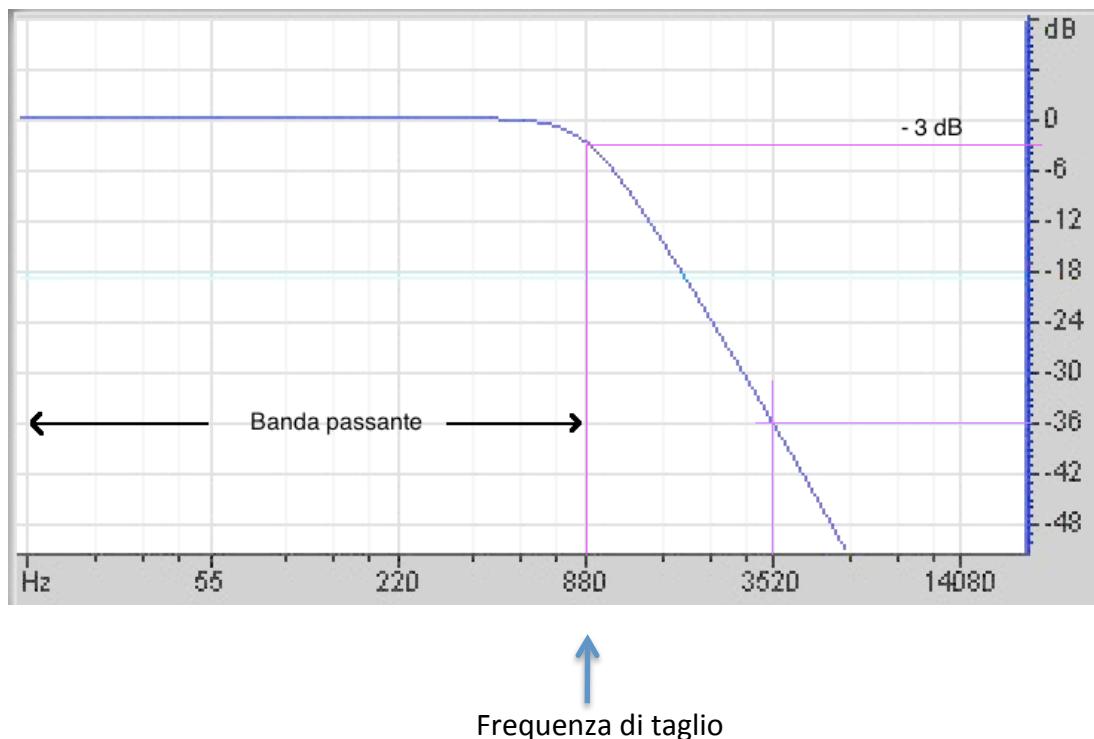


Filtro Elimina-Banda  
BR Filter (Band Reject)

I diagrammi rappresentano la Risposta in Frequenza di tali filtri. In ordinata l'ampiezza in dB e in ascissa la frequenza in Hz.

# Caratterizzazione dei filtri

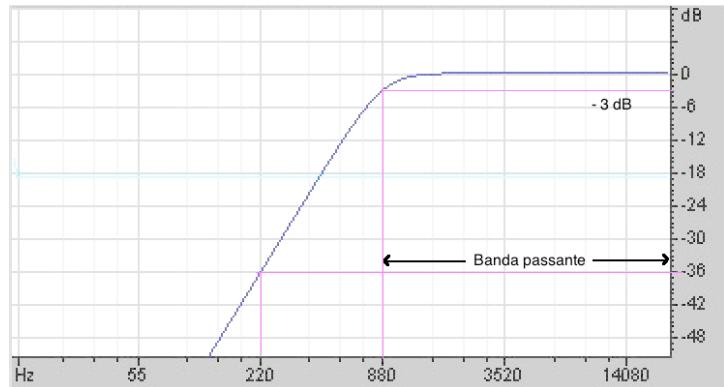
I filtri sono caratterizzati, oltre che dalla tipologia della loro risposta, anche da altri fattori. Uno di questi è il loro grado di selettività cioè la capacità di attenuare il segnale al di fuori della banda passante



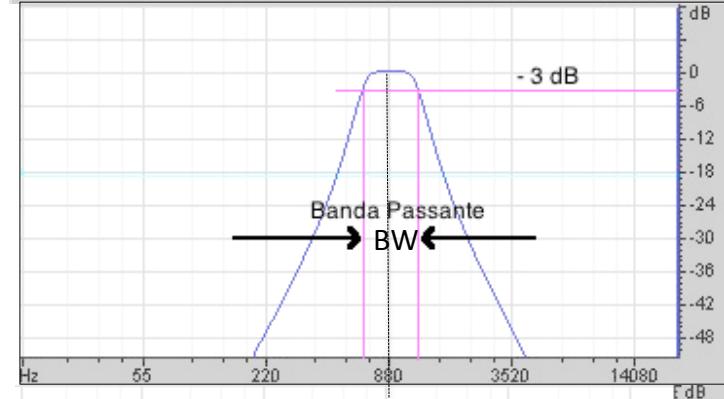
In questo esempio è mostrata la risposta in frequenza di un filtro passa-basso del terzo ordine con frequenza di taglio a 880 Hz. In corrispondenza a tale frequenza la risposta si è ridotta di -3 dB. La selettività misura la pendenza della risposta espressa come dB/ottava. Come si vede dalla figura, a 4 volte la frequenza di taglio ( $880 \times 4 = 3520$ ) cioè 2 ottave sopra, la risposta si è ridotta a -36 dB, e quindi -18 dB per un'ottava. La selettività quindi dipende dall'ordine del filtro:

- 1° ordine : 6 dB/oct
- 2° ordine : 12 dB/oct
- 3° ordine : 18 dB/oct
- ...etc....

# Passa Alto- Passa Banda – Elimina Banda

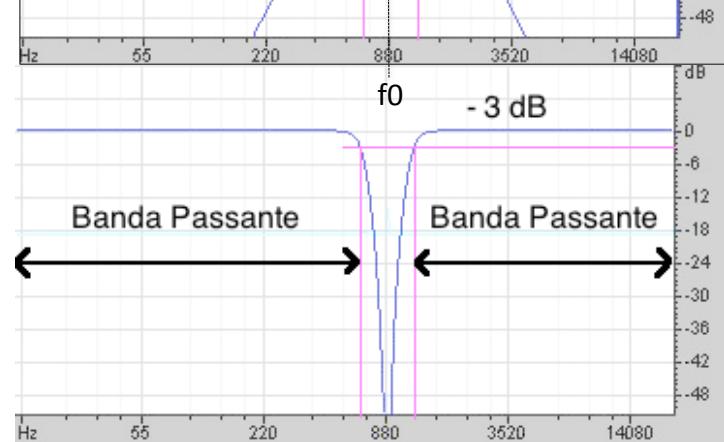


**Filtro Passa Alto (HP)** del 3° ordine con frequenza di taglio a 880 Hz ( 18 dB/ottava)



**Filtro Passa Banda (BP)** del 3° ordine con frequenza centrale a 880 Hz ( 18 dB/ottava). In questo caso la banda passante è delimitata dai due punti a – 3 dB superiormente e inferiormente.

Un altro parametro caratterizzante è il fattore di merito Q dato da  $f_0 / BW$

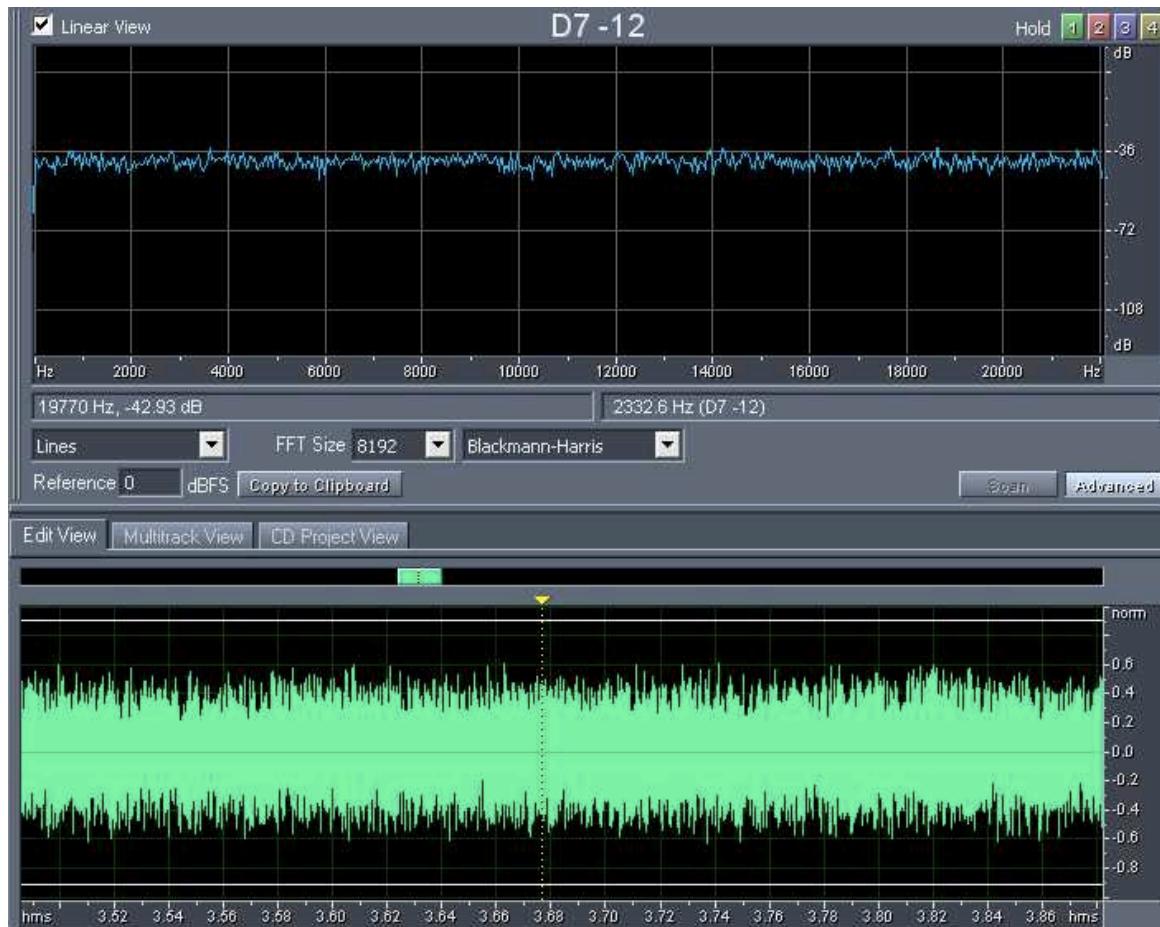


**Filtro Elimina Banda (BR)** del 3° ordine con frequenza centrale a 880 Hz ( 18 dB/ottava). In questo caso la banda passante è delimitata tra 0 e il primo punto a – 3dB e tra il secondo punto a – 3dB e la frequenza massima.

# Segnali aleatori (rumore bianco)

L'estremo opposto dei segnali periodici è rappresentato dai segnali **aleatori** che presentano andamenti generalmente imprevedibili e non ripetitivi e sono per questo catalogati come segnali di rumore

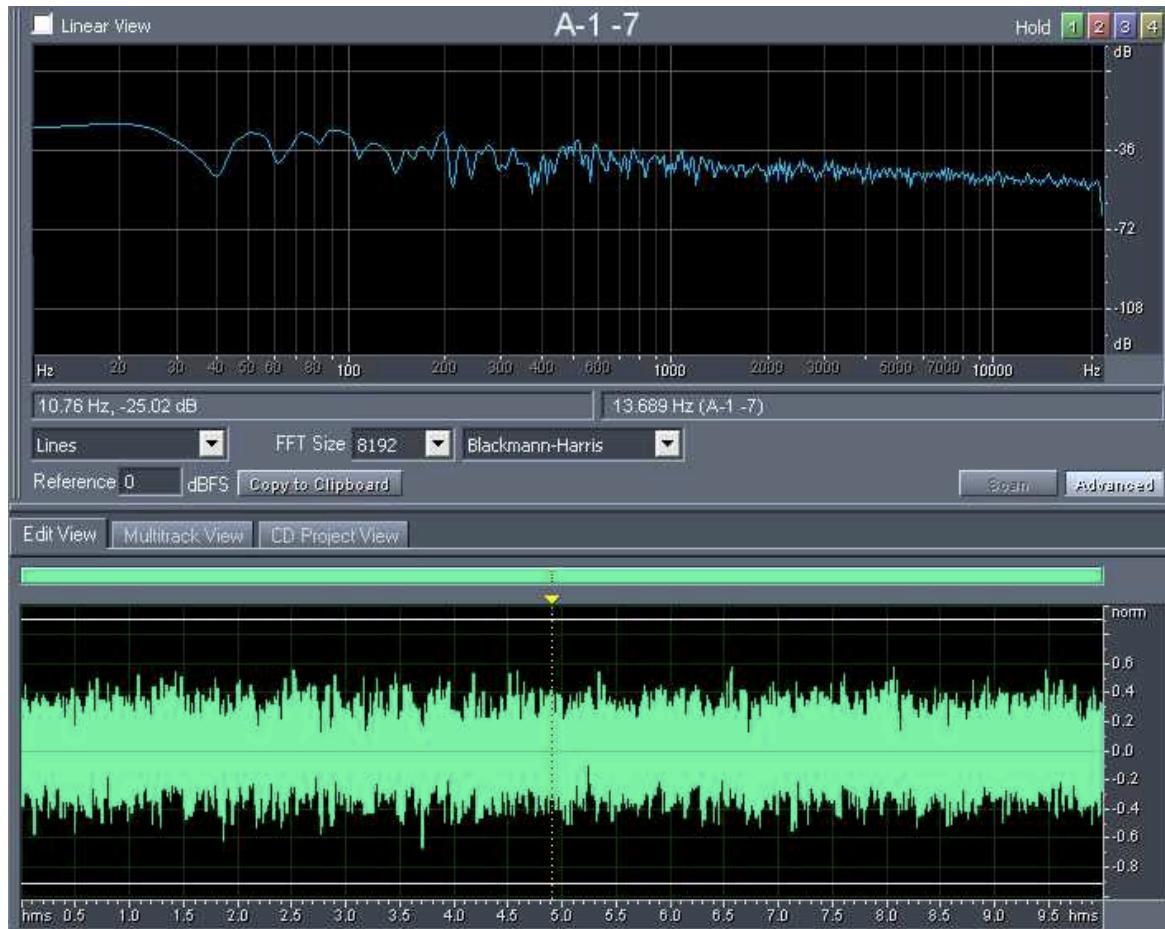
Tra questi, è di importante rilevanza il **rumore bianco** (*white noise*). Il rumore bianco si caratterizza quindi per totale mancanza di periodicità nel tempo e ampiezza costante lungo l'asse frequenziale.



- Lo spettro di ampiezza del rumore bianco è essenzialmente piatto per tutte le frequenze.  
Nella pratica dei segnali digitali si possono produrre al più sequenze pseudo-random con periodi di ripetizione molto estesi.
- Il rumore bianco, contenendo in pratica tutte le frequenze dello spettro è utilizzato come segnale di test per filtri e sistemi da caratterizzare.
- Nelle applicazioni audio il rumore è usato nella sintesi sottrattiva. Attraverso i filtri si è in grado di produrre "fasce di rumore" colorate.

# Segnali aleatori (rumore rosa)

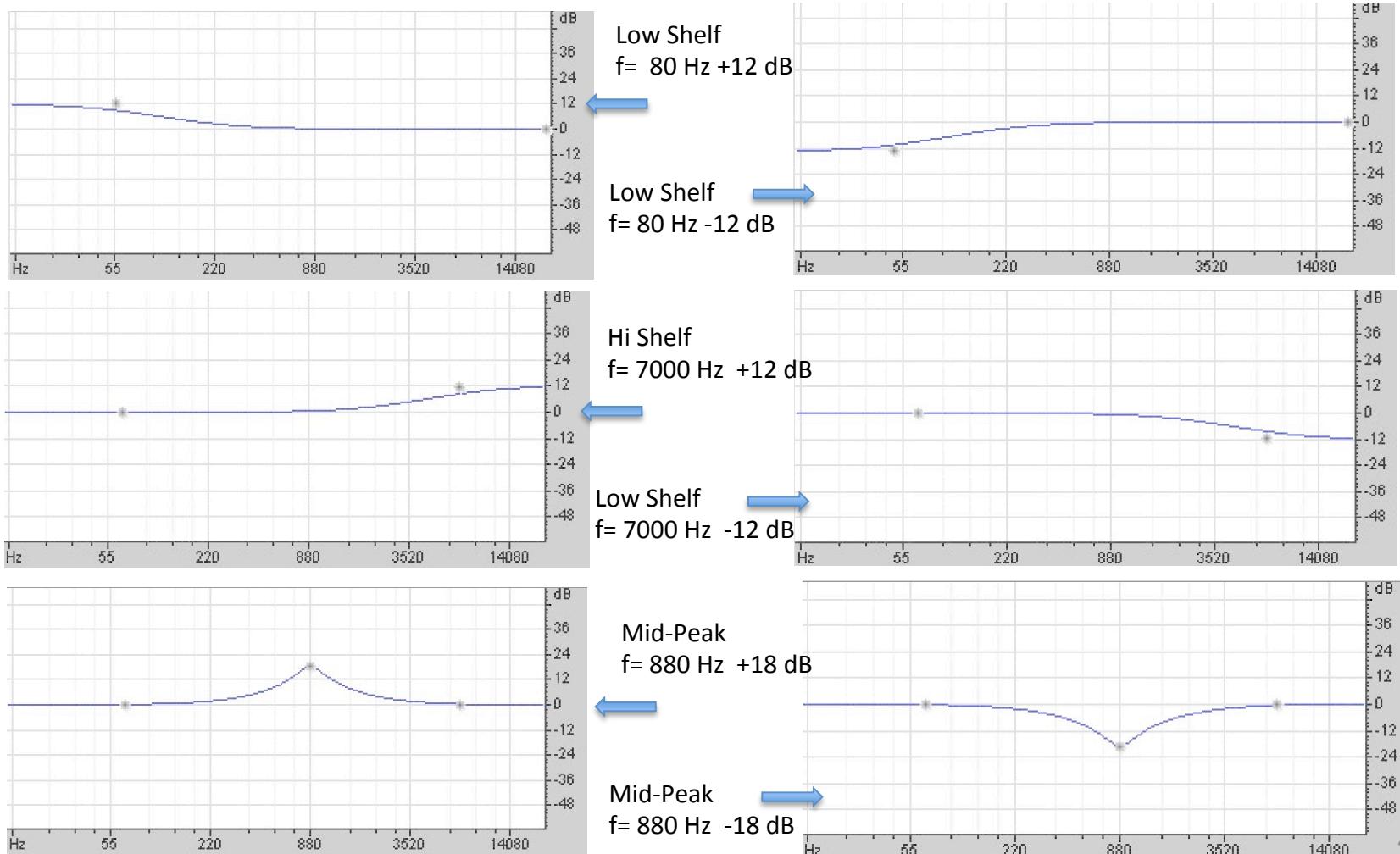
Il **rumore rosa** (*pink noise*) è una variante del rumore bianco ed è caratterizzato da una dipendenza con la frequenza di tipo **1/f** e per tale motivo le frequenze gravi presentano una maggiore ampiezza di quelle acute.



- Lo spettro di ampiezza del rumore rosa decresce di circa 3 dB ogni decade di frequenza (o 1.98 dB / ottava) ed è utilizzato in acustica per il test delle sale essendo più vicino alla risposta uditiva umana.
- Il decremento in ampiezza rispetto alla frequenza fa sì che il rumore rosa presenti lo stesso livello di energia per ogni ottava mentre nel rumore bianco questa energia tende ad aumentare.

# Filtr di Equalizzazione (Shelves)

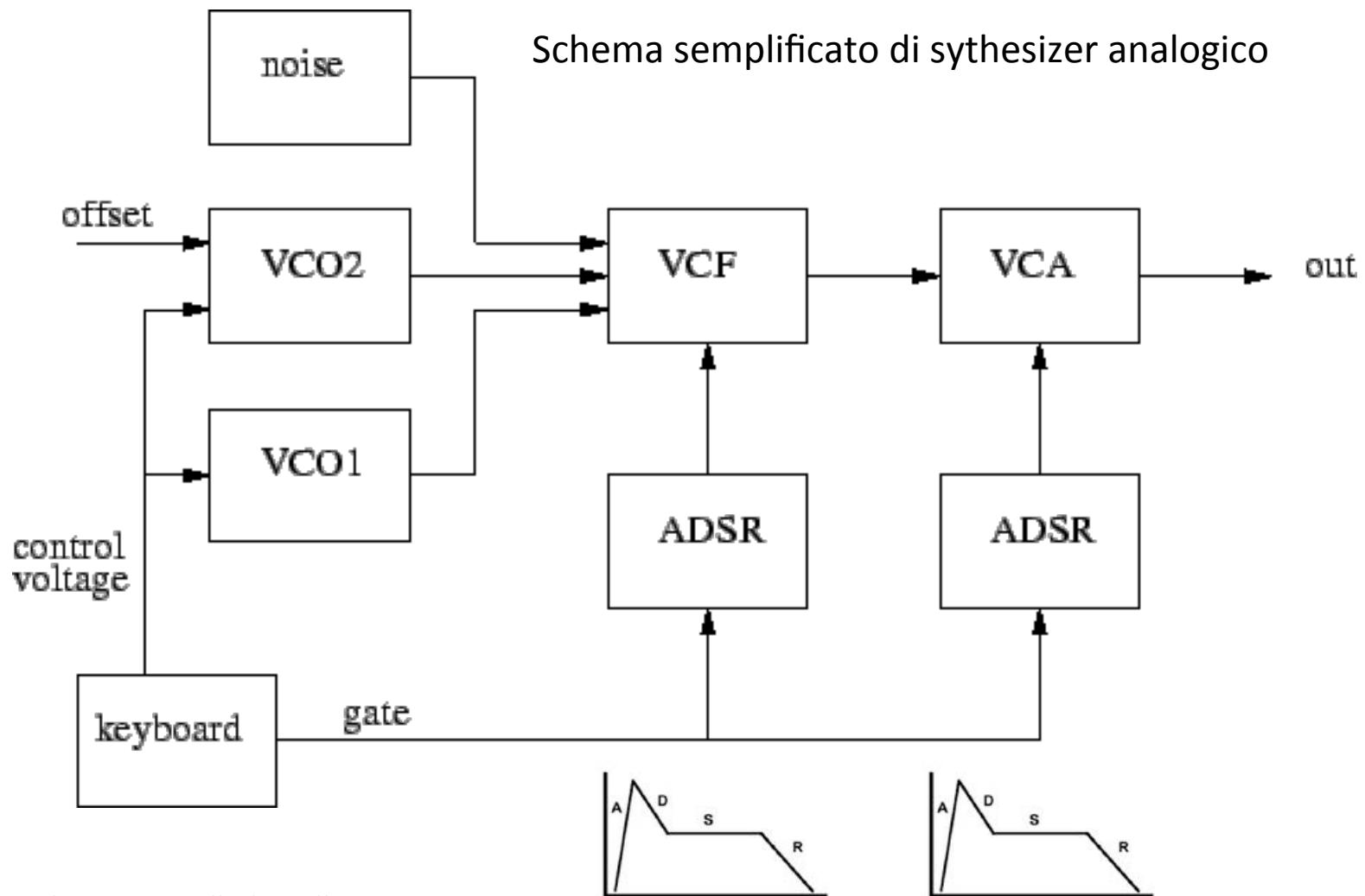
Sono filtri che servono a modificare la risposta operando in zone specifiche della frequenza con azioni di *esaltazione* e di *attenuazione* ma lasciando invariato il resto della banda. Di seguito sono mostrate le tipologie base: *Low-Shelf / Hi-Shelf / Mid-Peak*. Trovano posto nei mixer analogici e digitali.



# Sintetizzatori

- Sono macchine musicali elettroniche per la sintesi/elaborazione del suono prodotti industrialmente tra la metà degli anni 60 e la fine degli anni 70 in versione “analogica”. Sono preceduti da strumenti “elettrofoni” di sperimentazione o di scarsa produzione industriale che possono essere definiti i loro precursori. Tra questi ricordiamo:
  - Thelarmonium
  - Trautonium
  - Theremin
  - Onde Martenot
  - Clavioline
- La maggior parte dei sintetizzatori nella versione analogica utilizza la tecnica di sintesi **sottrattiva** ed in particolare l’uso di un principio di connessione dei moduli principali di generazione/elaborazione conosciuto con il nome di **controllo di tensione** (control voltage)

# Sintesi sottrattiva applicata ai synthesizer



VCO : voltage controlled oscillator

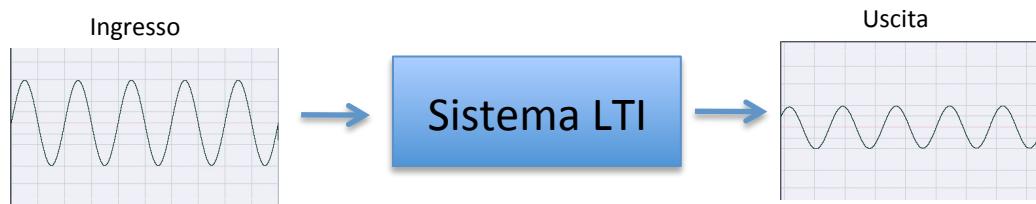
VCF : voltage controlled filter

VCA : voltage controlled amplifier

*NOTA: Lo schema, anche nella forma semplificata può comprendere un ulteriore oscillatore LFO (Low Frequency Oscillator) che viene impiegato per modulare in ampiezza e frequenza i vari moduli VC*

# Definizione di Risposta

- Il comportamento di un filtro e in generale di un sistema che sia in grado di accettare un ingresso e produrre un'uscita è descritto interamente con la risposta nel tempo (**risposta impulsiva**) e nel dominio della frequenza (**risposta in frequenza**).
- Il termine “risposta” denota quindi la reazione d’uscita di un sistema ad un certo stimolo presentato al suo ingresso. Esiste una categoria di sistemi definiti LTI (Lineare a Tempo Invariante) per i quali tali risposte li descrivono completamente. I filtri precedentemente introdotti appartengono a tale categoria.
- Se un sistema allora è LTI, applicando al suo ingresso un segnale sinusoidale stazionario l’uscita sarà ancora un segnale sinusoidale di stessa frequenza ma di *ampiezza e fase* modificate.

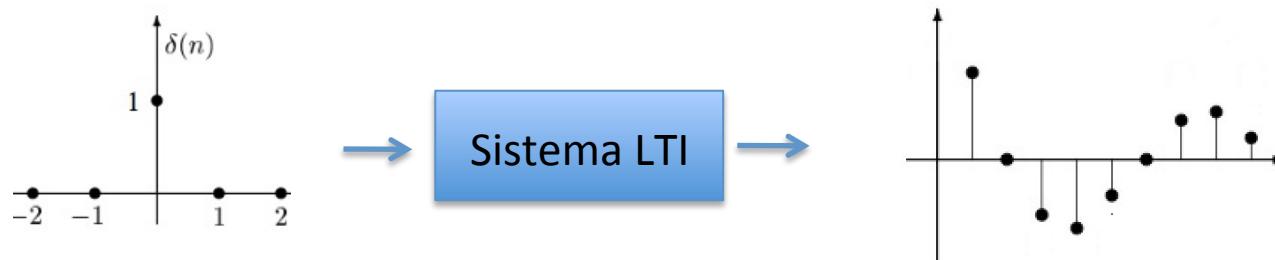
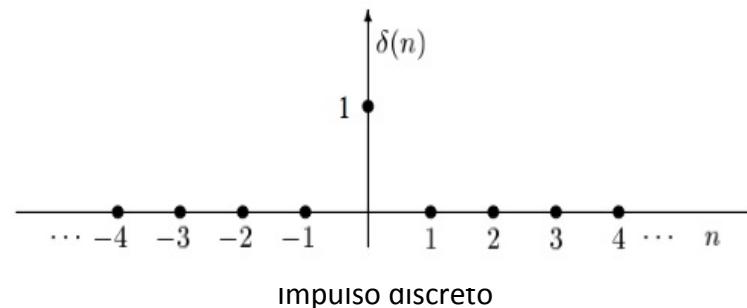
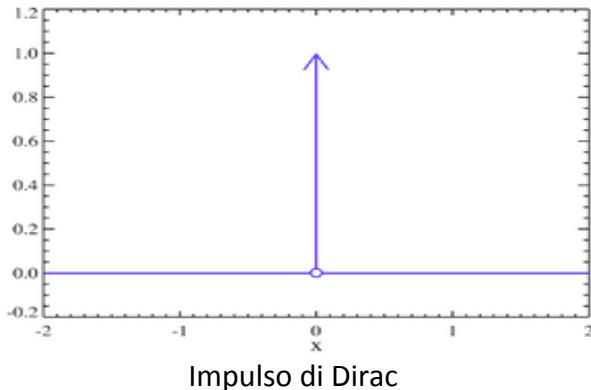


Se in tali condizioni, la frequenza del segnale d’ingresso viene fatta variare nel range di interesse (per esempio da 20 a 20 KHz nel caso dei segnali audio) per ogni valore di frequenza saremo in grado di conoscere i rispettivi valori di uscita di frequenza e fase e cioè appunto ciò che di fatto costituisce la risposta in frequenza del sistema LTI

I diagrammi rappresentano la Risposta in Frequenza di tali filtri. In ordinata l’ampiezza in dB e in ascissa la frequenza in Hz.

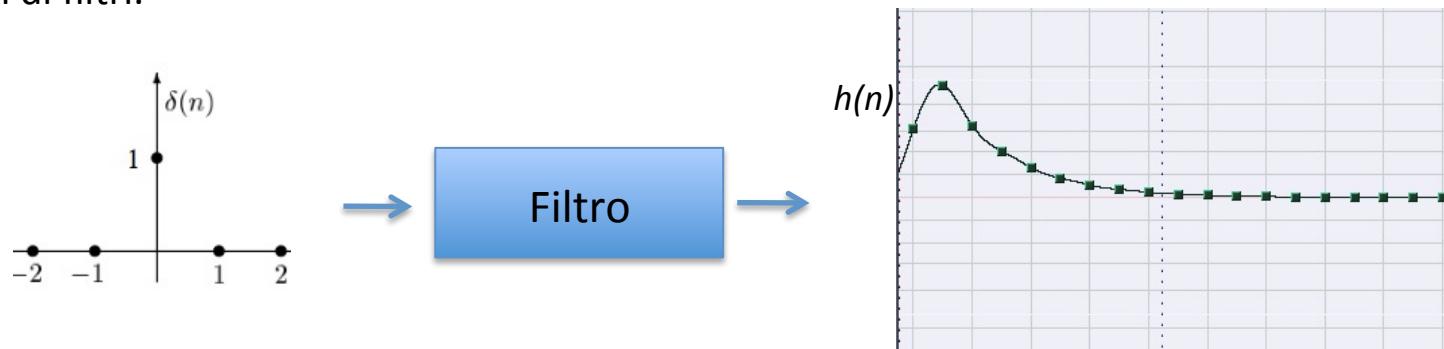
# Risposta impulsiva 1/2

Un modo per ottenere invece la risposta temporale di un sistema è quella di stimolarlo con un segnale non periodico di tipo impulsivo. Nel caso dei segnali analogici il segnale di eccitazione è una particolare funzione detta impulso di Dirac o Delta di Dirac. La formulazione matematica di tale particolare funzione (non è realizzabile integralmente nella pratica e di fatto si accetta una certa approssimazione) prevederebbe che l'impulso sia nullo fuorchè per  $t = 0$  e la sua area sottesa sia unitaria. Nel caso dei segnali discreti tale stimolo risulta di più semplice attuazione poiché rappresentato da una sequenza di campioni nulli tranne uno unitario nell'origine



# Risposta Impulsiva 2/2

Sfruttando alcune proprietà matematiche dei sistemi LTI è possibile definire una fondamentale relazione che intercorre tra la risposta impulsiva e quella in frequenza. Tale relazione ha notevoli implicazioni pratiche nell'utilizzo nella modellazione del suono con le tecniche che fanno uso di filtri.



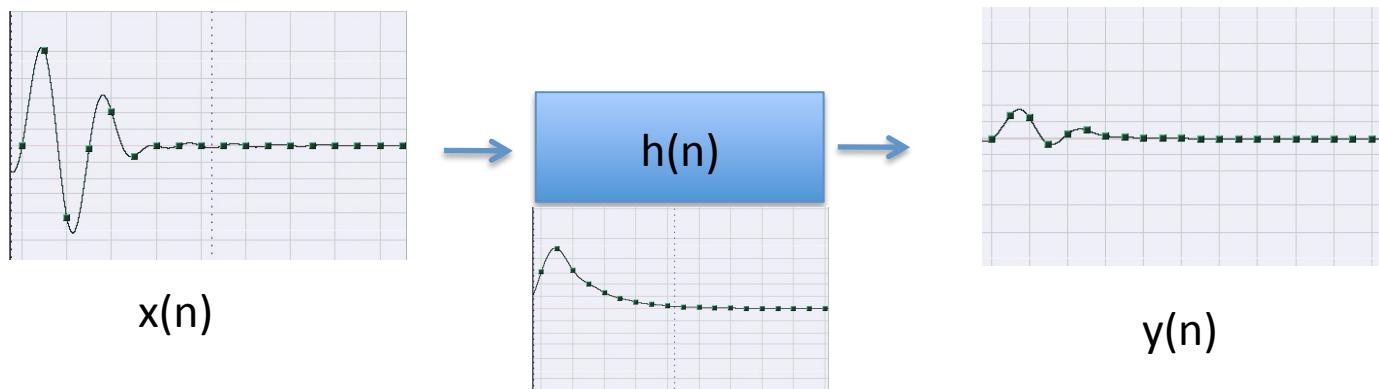
Quando un filtro viene stimolato con un impulso unitario, come già detto, la sua risposta temporale è detta **risposta impulsiva**. Tale risposta è una sequenza di impulsi successivi nel tempo di vario segno e ampiezza. Denominiamo tale sequenza come  $h(n)$ .

Se al posto dello stimolo unitario il filtro viene stimolato da un segnale, ogni impulso di tale segnale produrrà una risposta ritardata nel tempo e proporzionale alla singola ampiezza e segno. Ciascuna di queste singole risposte si sommano tra di loro formando così la risposta complessiva del filtro.

# Convoluzione 1/3

Una volta che sia nota la risposta impulsiva  $h(n)$  possiamo conoscere il comportamento del sistema per qualunque stimolo  $x(n)$ . Il processo di somma ritardata delle singole risposte descritto nella pagina precedente è formalizzabile con una operazione matematica detta *somma di convoluzione* o semplicemente **convoluzione**. Secondo questa formalizzazione quindi il segnale d'uscita si ottiene eseguendo la convoluzione tra il segnale d'ingresso e la sua risposta impulsiva:

$$y(n) = x(n) \star h(n)$$



Nota: in questo esempio la risposta  $h(n)$  è relativa ad un filtro passa-basso del 1° ordine (6dB/oct) con frequenza di taglio  $F_c = 3\text{Khz}$

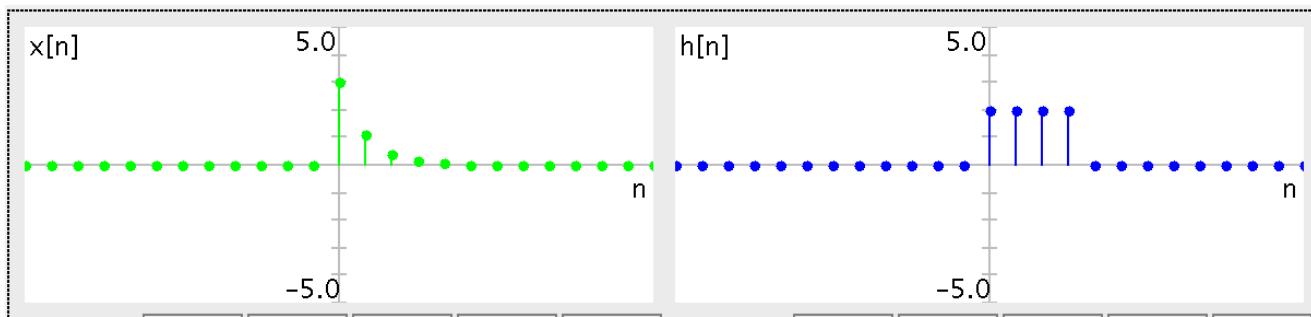
# Convoluzione 2/3

In un sistema discreto, lineare e tempo invariante, la convoluzione è espressa nel modo seguente:

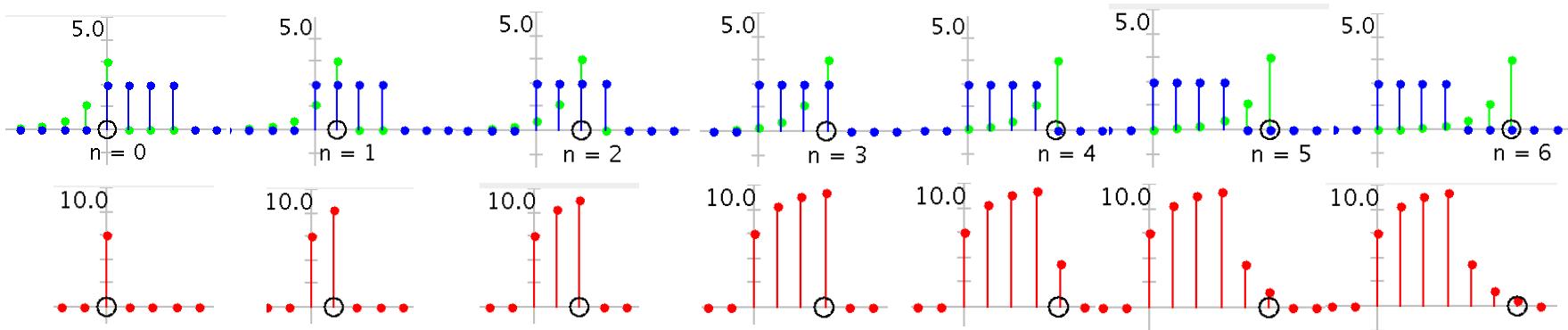
$$y[n] = \sum_{k=-\infty}^{\infty} h[k] x[n-k]$$

dove :  $y(n)$  o  $y(nT_c)$  è la versione discreta di  $y(t)$  e  $T_c = 1/F_c$

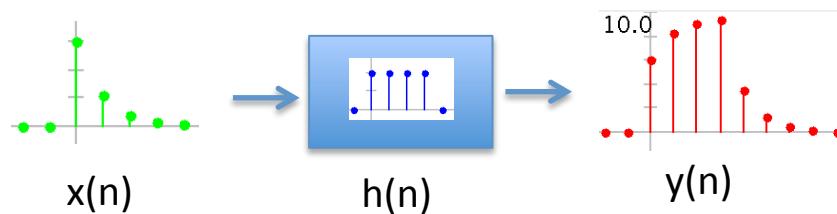
:  $x(n)$  o  $x(nT_c)$  " " "  $x(t)$   
:  $h(n)$  o  $h(nT_c)$  " " "  $h(t)$



# Convoluzione 3/3



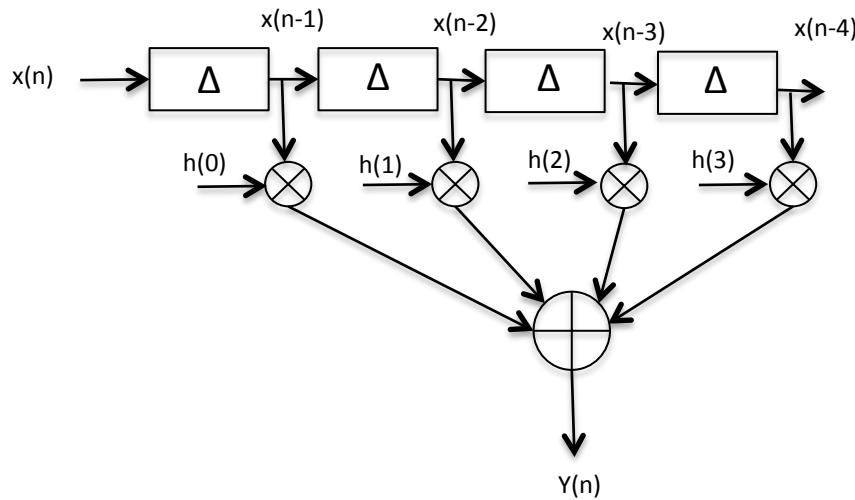
- $y(n)$  : Segnale d'uscita
- $x(n)$  : Segnale d'uscita
- $h(n)$  : Risposta impulsiva



La somma di convoluzione ( o convoluzione) ha una semplice interpretazione grafica. Per prima, rappresentare  $h[k]$  e la forma "invertita e spostata"  $x[n - k]$  sull'asse  $k$ , dove  $n$  è fissato. Secondo, moltiplicare i due segnali per ottenere un grafico della sequenza in somma indicizzata da  $k$ . Sommare i valori di questa sequenza rispetto a  $k$  su  $y[n]$ . Queste operazioni devono essere ripetute per ogni valore di  $n$ .

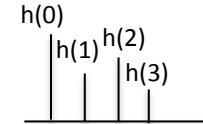
# Convoluzione 2/3

L'operazione di convoluzione in ambito digitale può essere realizzato nel dominio del tempo discreto secondo il seguente schema:



Il segnale d'ingresso  $x(n)$  viene fatto transitare attraverso una delay line (linea di ritardo) costituita da ritardi unitari ( $\Delta = 1$  sample). Le uscite dai singoli ritardi vengono poi moltiplicate per i valori di ampiezza della risposta impulsiva  $h(n)$

Es:



Per ogni campione del segnale d'ingresso  $x(n)$  devono essere quindi eseguite tante moltiplicazioni quanti sono i valori che costituiscono la risposta impulsiva più una somma.

# Relazione tempo-frequenza

Il processo di convoluzione , nonostante sia concettualmente semplice da realizzare, nella pratica può diventare oneroso da un punto di vista computazionale poiché ciascun campione del segnale d'ingresso deve essere moltiplicato per tutti i campioni (ritardati) che costituiscono la risposta impulsiva e sommati tra loro. Fortunatamente, la conoscenza delle proprietà matematiche della trasformata di Fourier consentono di realizzare lo stesso processo nel dominio della frequenza. Vale allora la seguente e importante corrispondenza:

$$\begin{aligned}y(n) &= x(n) * h(n) && \text{(dominio del tempo)} \\Y(f) &= X(f) \cdot H(f) && \text{(dominio della frequenza)}\end{aligned}$$

dove  $Y(f)$  e  $X(f)$  sono le trasformate di Fourier rispettivamente dei segnali d'uscita, ingresso e  $H(f)$  è la trasformata di Fourier della risposta impulsiva (corrispondente alla Risposta in frequenza). Ne consegue che l'operazione di convoluzione può essere sostituita dalla operazione più elementare di prodotto, a patto che si considerino le trasformate dei segnali in gioco.

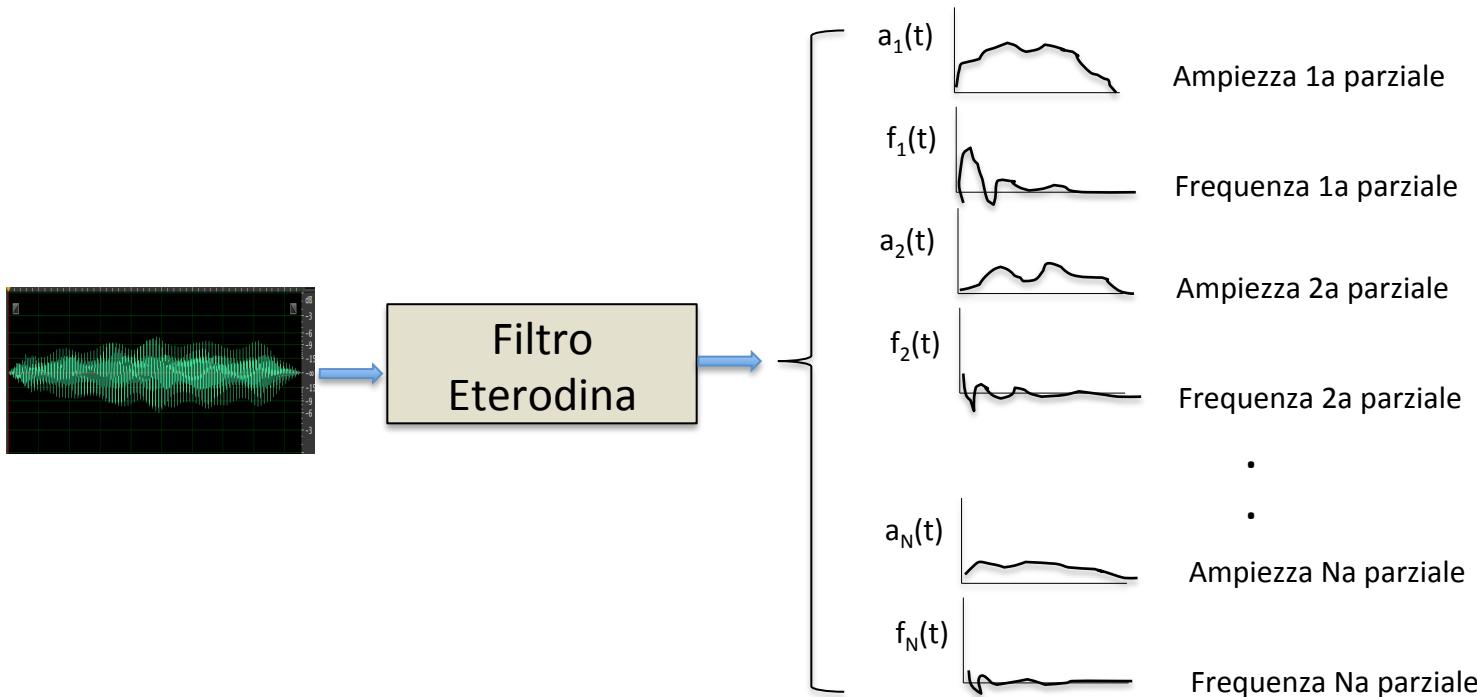
Per riottenere il segnale nel dominio del tempo , una volta eseguito il prodotto delle trasformate occorre eseguire il processo di trasformata inversa di Fourier.

# Analisi e risintesi

- La tecnica di sintesi additiva, oltre ad essere una modalità di produzione creativa del suono, può essere adottata per sintetizzare suoni reali precedentemente analizzati.
- Esistono molte varianti di processi di analisi-sintesi. Tra i tanti possiamo ricordare quelli basati sulla tecnica “**eterodina**” e sul “**phase-vocoder**”
- Entrambe le tecniche prevedono quindi che la fase di sintesi sia preceduta da un processo di analisi.
- In generale, il processo d’analisi è in grado di rappresentare il suono attraverso una descrizione strutturale che può essere assunta come punto di partenza per la risintesi. Il processo di risintesi può essere limitato alla pura ricostruzione del suono originale o all’elaborazione dei dati d’analisi consentendo così di creare varianti rispetto all’originale.

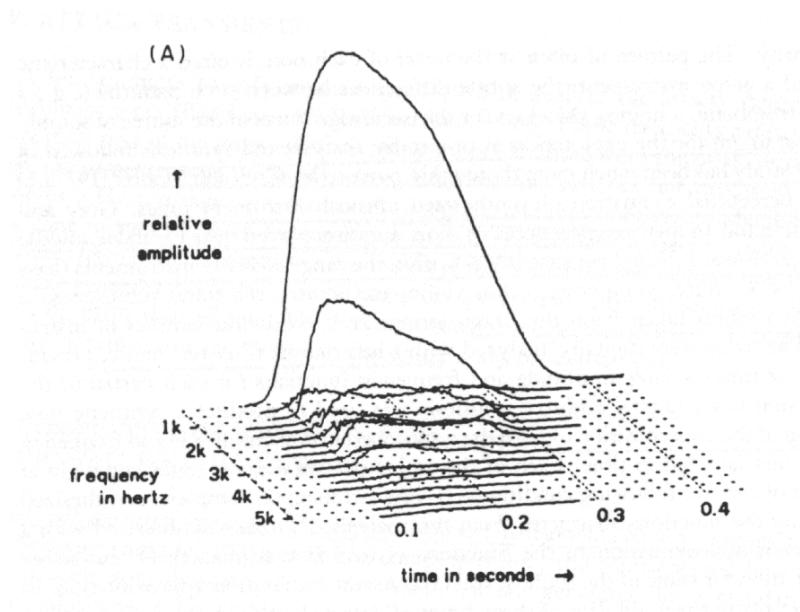
# Metodo Eterodina

- Il metodo “eterodina” consiste nell’utilizzazione di un procedimento di analisi che è in grado di decomporre il suono in componenti sinusoidali correlate armonicamente. A partire da un frequenza definita fondamentale, vengono estratte coppie di funzioni che descrivono nel tempo l’andamento di ampiezza e frequenza di ogni parziale. Il metodo è indicato per analizzare suoni con spiccata struttura armonica.



# Rappresentazione dei dati d'analisi

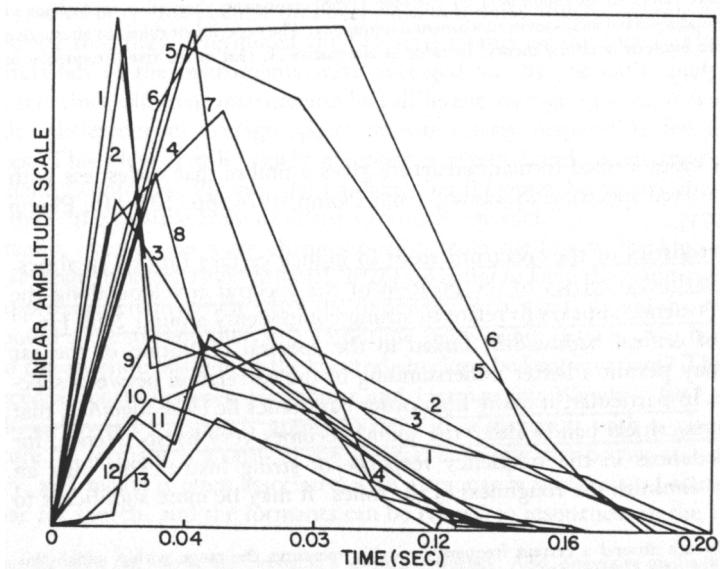
- L'andamento delle funzioni d'ampiezza può essere rappresentato complessivamente in un diagramma tridimensionale con  $x = \text{tempo}$ ,  $y = \text{ampiezza}$  e  $z = \text{frequenza}$ . In questa rappresentazione le singole tracce di frequenza appaiono costanti e multiple intere della frequenza fondamentale. Questa semplificazione permette di valutare in modo chiaro gli andamenti d'ampiezza e le relative proporzioni. Nel grafico sottostante è rappresentata l'analisi di un suono strumentale di clarinetto.



L'analisi mostra una fondamentale di ampiezza molto maggiore delle parziali superiori. Inoltre, dalla rappresentazione 3D si evince che le parziali di ordine pari sono sistematicamente molto inferiori rispetto a quelle dispari.

# Linearizzazione dei dati d'analisi

La tecnica di analisi ha permesso di eseguire studi molto interessanti sulla natura del timbro degli strumenti musicali (Risset, Grey e Moorer). In particolare uno studio del 1975, "An Exploration of Musical Timbre" è stato sviluppato attraverso le tecniche d'analisi e risintesi con il metodo eterodina. (<https://ccrma.stanford.edu/files/papers/stanm2.pdf>). Le funzioni che rappresentano l'ampiezza delle singole parziali può essere linearizzato cioè definito attraverso spezzate lineari

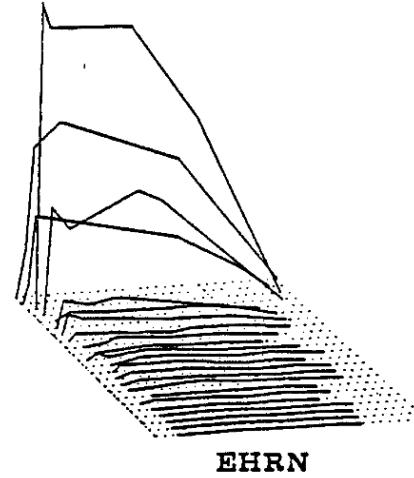
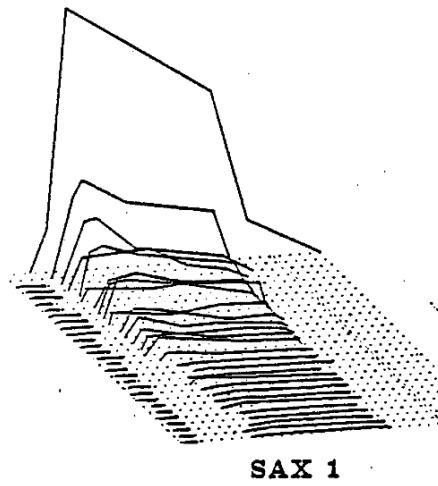
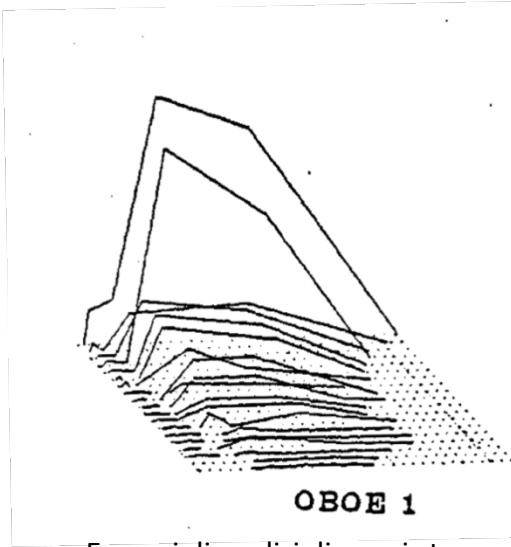


Rappresentazione delle prime 13 parziali di un suono di tromba (D4, 0.2 secondi di durata) attraverso una riduzione a segmenti lineari delle funzioni d'ampiezza. (J.C. Risset). Lo studio permette di osservare le asincronie temporali delle singole parziali e alcune oscillazioni nella fase di attacco di alcune di esse (11, 12, 13).

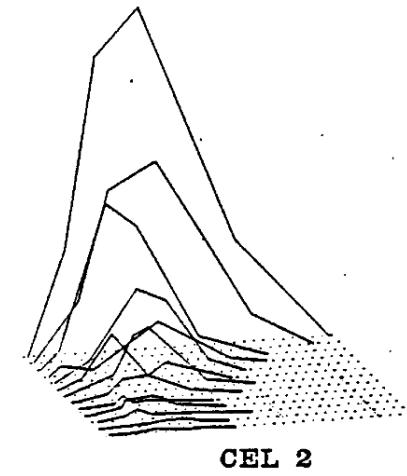
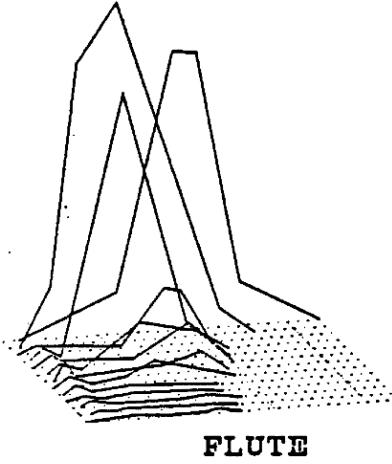
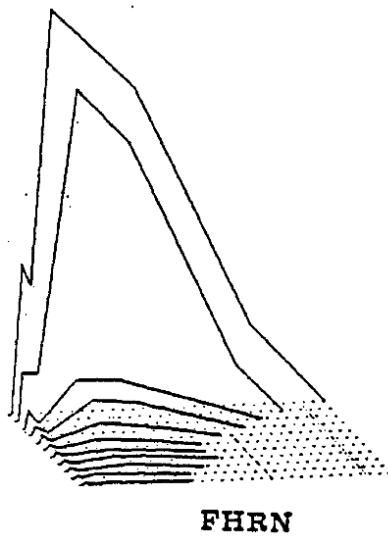
Le funzioni di ampiezza sono

# Riduzione dei dati d'analisi

Gli studi di Grey hanno dimostrato che se le funzioni d'ampiezza vengono approssimate con un processo di linearizzazione, la risintesi produce risultati molto prossimi all'originale e che in generale il dato più critico da ridurre è l'informazione frequenziale e di asincronia temporale.

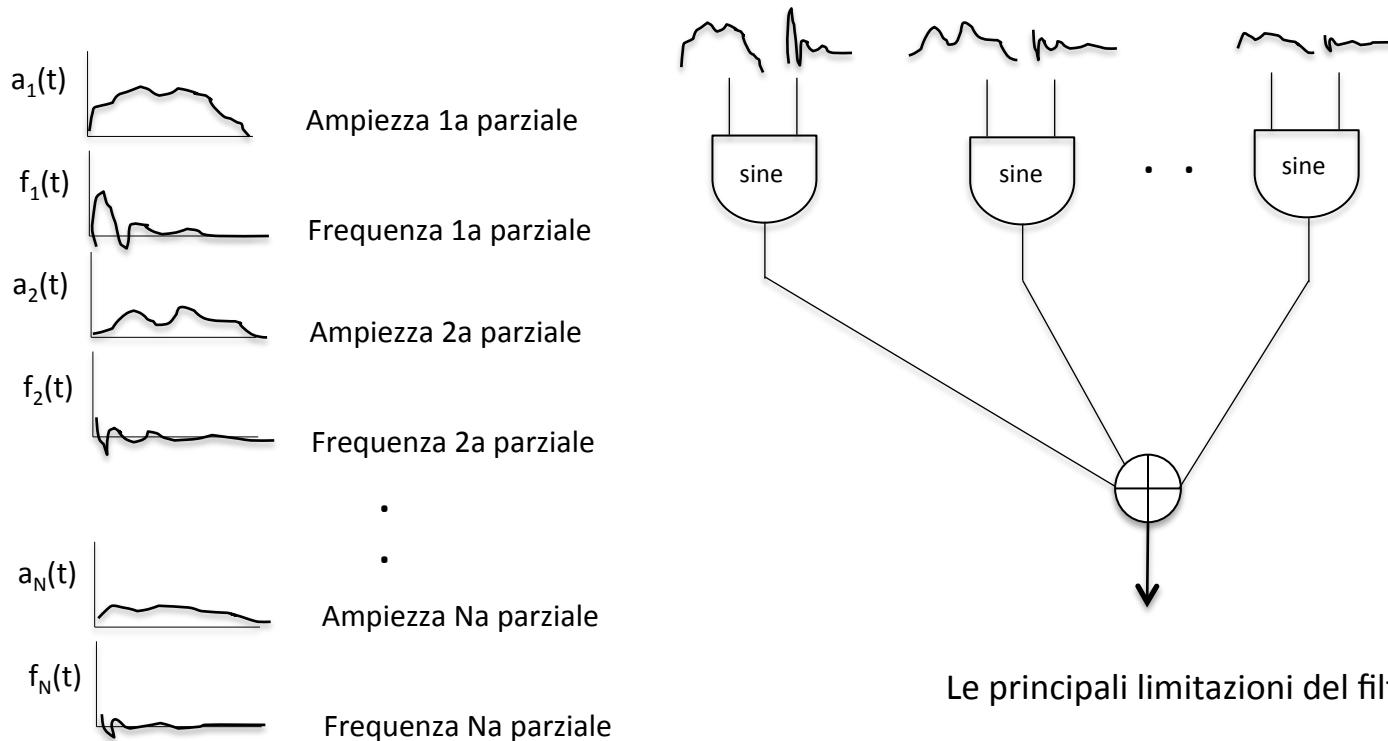


Esempi di analisi di suoni strumentali derivanti dalle ricerche di Gray (CCRMA – Stanford- Report N. STAN-M-2)



# Dall'analisi alla sintesi

La tecnica di analisi basata sul filtro eterodina permette di ricostruire il suono d'origine attraverso la somma di un banco di oscillatori sinusoidali controllati ciascuno da una coppia di inviluppi (ampiezza e frequenza) estratti dal filtro.



Le principali limitazioni del filtro eterodina sono:

- a) Difficoltà di seguire variazioni veloci di ampiezza
- b) La sintesi avviene con le componenti a fase zero

# Vocoder (comunicazioni)

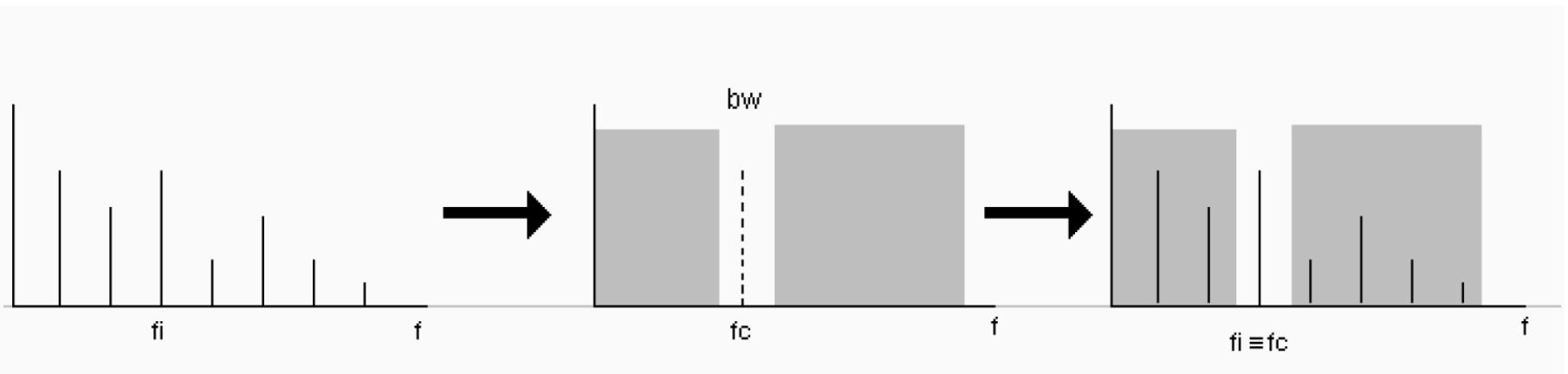
**Premessa storica:** il vocoder, (VOice CODER) è un sistema di analisi / sintesi, utilizzato per riprodurre il linguaggio umano. Nel codificatore, l'ingresso viene fatto passare attraverso un filtro multibanda, ciascuna banda viene fatto passare attraverso un estrattore d'inviluppo. Ciascun inviluppo costituisce un segnale (non audio) da trasmettere via radio e inviati al decodificatore di destinazione. Il decodificatore utilizza questi segnali come inviluppi di controllo per i filtri corrispondenti nel sintetizzatore. Poiché i segnali di comando cambiano solo lentamente rispetto alla forma d'onda di parlato originale, la larghezza di banda necessaria per trasmettere è molto inferiore rispetto a quella necessaria per l'intero segnale audio. Questo permette più canali vocali di condividere un circuito di radio o cavo sottomarino. Con la crittografia dei segnali di controllo, la trasmissione di voce può essere protetta contro le intercettazioni.



Vocoder analogico usato per le comunicazioni telefoniche criptate

Il Phase Vocoder (PVOC) è un processo matematico per convertire un segnale campionato in una rappresentazione spettrale tempo-variante. Esso è computazionalmente efficiente quando viene implementato con la trasformata rapida di Fourier (FFT). Se il suono viene ri-sintetizzato dai dati di analisi, l'uscita sarà virtualmente identica al suono originale. Il PVOC offre alcuni vantaggi in più rispetto al filtro eterodina nell'analisi tempo-variante dei suoni.

# Phase Vocoder



Il concetto base del PV è dato dal processo di selezione delle frequenze con un banco di filtri passabanda di larghezza di banda opportuna. Il PV moderno impiega in realtà la Trasformata di Fourier al posto dei filtri dal momento che la trasformata stessa opera come un banco di filtri

La determinazione della larghezza di banda dei filtri richiede qualche attenzione. Se essa è troppo grande in confronto con la spaziatura tra le parziali, passeranno attraverso di essa più componenti.

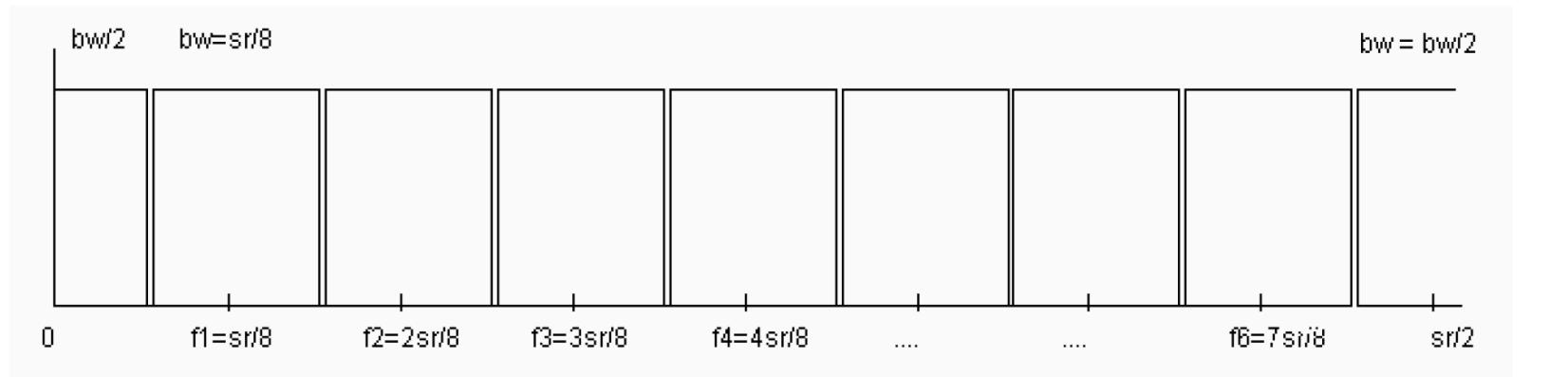
D'altra parte, larghezze di banda più strette richiedono un numero maggiore di filtri al fine di coprire l'intera estensione dello spettro, aumentando così il numero di calcoli da eseguire.

Per una certa frequenza  $f$ , sono necessari "n" campioni (dove  $n = sr / f$ ) per definire il periodo intero. Questo significa che bisogna processare n campioni per analizzare la frequenza  $f$ .

Ad esempio per una frequenza  $f = 40$  Hz, se  $sr = 40000$  occorre processare un blocco di almeno 1000 campioni

# Phase Vocoder

Un banco di filtri può essere implementato con una *Fast Fourier Transform* (FFT) Questo procedimento converte un blocco di  $2^N$  campioni in una rappresentazione spettrale che è concettualmente equivalente all' uscita di un banco di filtri passa-banda linearmente distribuiti come mostrato nella seguente figura (nell'esempio  $N = 16$  ) e  $bw$  = larghezza di banda di ogni filtro



Concettualmente, tutti i filtri sono dei passa-banda e hanno la stessa larghezza di banda  $bw$ , ad eccezione del primo, che è un passa-basso con frequenza di taglio pari a  $bw/2$ , e dell'ultima, che è un passa-alto con frequenza di taglio pari a  $sr/2 - bw/2$ . L'FFT è ottimizzata per processare blocchi di campioni in numero pari potenze di 2

# Phase Vocoder

Al fine di rendere efficace l' impiego del banco di filtri, la frequenza della parziale a frequenza più bassa deve coincidere con la frequenza centrale del primo filtro passa-banda. Se il numero di campioni è  $2N$  allora la frequenza centrale del primo filtro è :

$$f_1 = sr / 2^N = sr / (2 * 2^{N-1}) = \text{Nyquist} / 2^{N-1}$$

A questo punto possiamo calcolare la larghezza di banda dei filtri se si assume che la frequenza centrale è esattamente nel centro del filtro passa-banda. Si ha quindi che:

$$n_{bp} = (\text{Nyquist} - \text{lorgh. Banda Low Pass} - \text{lorgh. Banda High Pass}) / \text{lorgh. Banda}$$

dove  $n_{bp}$  = numero di filtri passa-banda

In conclusione, se consideriamo la somma dei due filtri passa-basso e passa-alto come un ulteriore passa-banda possiamo affermare che:

*Il numero di campioni usati al fine di analizzare un segnale determina il numero e la larghezza dei filtri passa-banda. Usando una FFT a  $2^N$  campioni, si divide lo spettro in  $2^{N-1}$  canali di larghezza pari a  $\text{Nyquist}/2^{N-1}$*

# Phase Vocoder

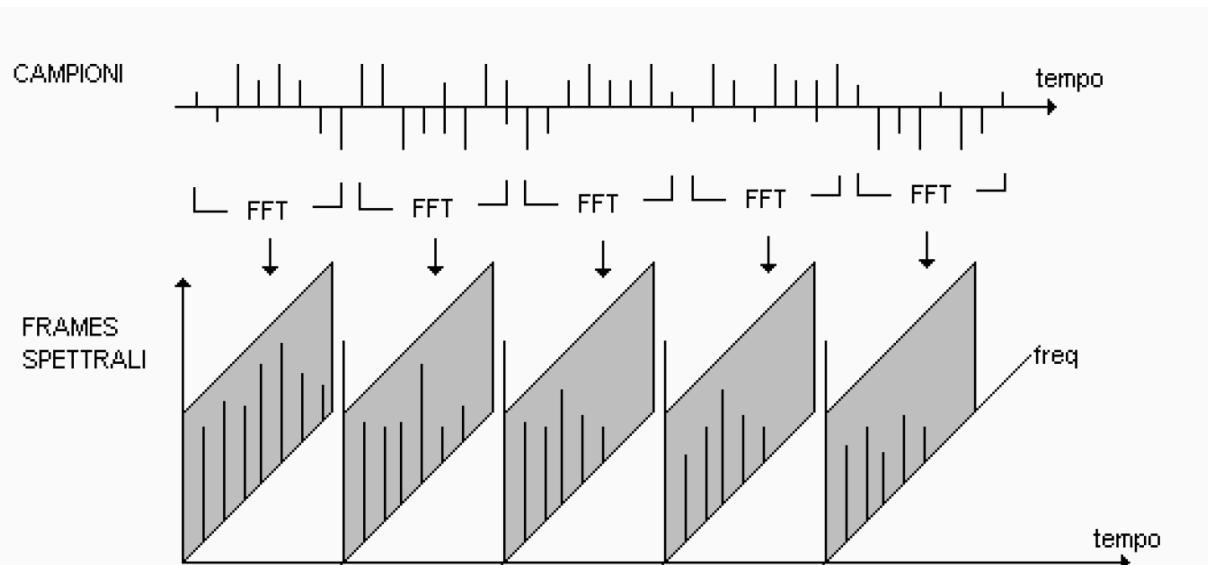
Come detto in precedenza, al fine di inseguire i cambiamenti dello spettro di un segnale che muta nel tempo, è necessario campionare tale spettro ad intervalli appropriati.

Il metodo più semplice consiste nell' applicazione della FFT a gruppi consecutivi di campioni. Se queste istantanee spettrali sono grandi, e regolarmente intervallate, una grande quantità di dettagli possono andare perduti. Viceversa se le istantanee sono piccole e ravvicinate si corre il rischio di perdere risoluzione frequenziale.

Le istantanee spettrali sono definite **frames** mentre il numero di frames /sec è detta **frequenza d'analisi**.

Un problema ulteriore sorge con l' isolamento di gruppi di campioni dal segnale originale.

All' interno del segnale, il primo e l' ultimo campione del gruppo sono preceduti e seguiti da altri campioni. Tuttavia, quando i gruppi di campioni vengono analizzati in isolamento, questi campioni sono preceduti e seguiti da nulla; in pratica i gruppi appaiono iniziare e finire in modo brusco, producendo transitori apparenti nel segnale (si possono udire come dei clicks). Quindi lo spettro derivato dall' FFT sarà gravemente distorto dalla presenza dei bordi di ciascun gruppo.

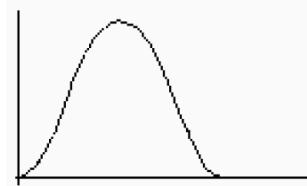


Trasformata di Fourier applicata a gruppi di 8 campioni successivi e contigui

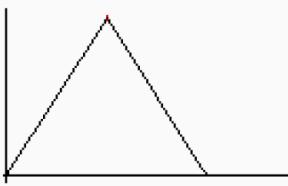
# Phase Vocoder

Il problema delle discontinuità può essere notevolmente ridotto operando continue assolvenze/dissolvenze tra i gruppi di campioni processati e attuando una sovrapposizione parziale (“*overlap*”) tra i frames. Ogni gruppo di campioni viene quindi sottoposto a “finestratura” (*windowing*) prima di eseguire l’FFT e di fatto ciò corrisponde all’applicazione di un inviluppo che produce un ammorbidente (*“smoothing”*). L’analisi frequenziale così articolata è conosciuta con il nome di STFT (Short Time Fourier Transform)

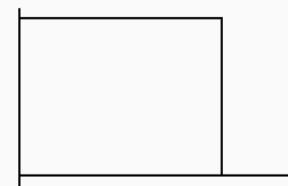
I tipi più comuni di finestra impiegate nella STFT corrispondono ai seguenti nomi: Bartlett, Blackmann, Hanning, Hamming e Kaiser. La prima di queste è un triangolo mentre Hanning è un coseno rialzato. Hamming e Blackmann sono varianti della forma a campana del coseno e Kaiser è una finestra molto più complessa che consente di eseguire controlli accurati sui filtri passa-banda della FFT. Se non viene applicato alcun inviluppo, la finestra è detta implicitamente rettangolare.



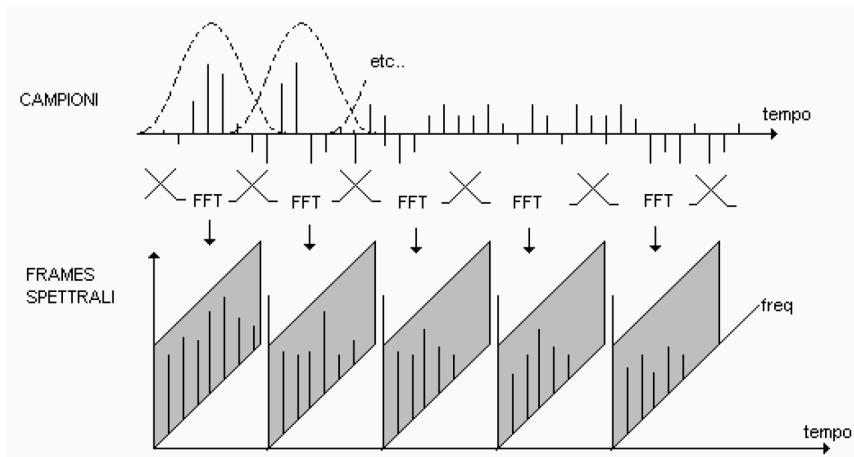
Hanning



Bartlett (Triangolare)



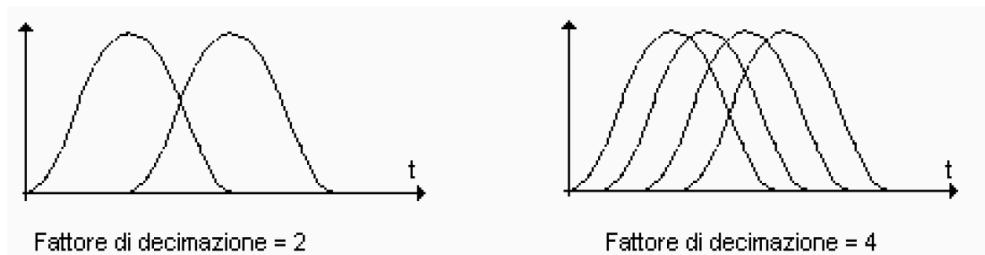
Rettangolare



FFT applicata a 8 gruppi di campioni successivi e contigui con overlap e windowing

# Phase Vocoder

La quantità di overlap è misurata usando un *fattore di decimazione*, che indica il numero di sovrapposizioni attraverso la durata di una singola finestra. In figura 7 sono mostrati fattori di decimazione pari a 2 e a 4.



È possibile trovare una relazione tra la lunghezza della finestra e la frequenza di campionamento originale del segnale. Se la finestra non viene sovrapposta (fattore di decimazione  $D = 1$ ), la frequenza di campionamento di analisi (asr) è semplicemente:

$$\text{asr} = \text{sr} / (\text{lunghezza finestra}) = \text{sr} / 2^N$$

Se  $D > 1$  (ovvero c'è sovrapposizione) allora l'asr viene di fatto aumentato. Per esempio, se  $D = 2$ , gli istanti di partenza delle finestre sono separati dalla metà della lunghezza della finestra stessa. Quindi la frequenza di analisi raddoppia in confronto con quella del caso senza sovrapposizione. Si può infatti scrivere che:

$$\text{asr} = D (\text{sr} / 2^N)$$

*La lunghezza della finestra ed il fattore di decimazione determinano la risoluzione della STFT usata per misurare l'evoluzione spettrale di un segnale.*

I filtri considerati all'inizio della trattazione, sono una idealizzazione della risposta in frequenza desiderata. Nei filtri reali, la transizione tra la banda passante e quella attenuata non è immediata: presenta una certa pendenza. Inoltre, la banda attenuata possiede una ondulazione (ripple) che permette ad una piccola parte del segnale di passare indisturbato. Questo fenomeno è chiamato perdita spettrale (*spectral leakage*) la cui quantità dipende dal tipo di finestra impiegata.

# Origini della computer music



Max Mathews con una delle sue Invenzioni: il Radio Baton, un sistema per controllare gestualmente la performance di un brano generato da computer.

[Max Mathews](#) ( 1926, Columbus, Nebraska, USA - 2011, San Francisco, CA, USA)

E' considerato il padre delle computer music. Ha dato vita al primo linguaggio di programmazione per la generazione ed elaborazione digitale del sonoro. Ha prevalentemente svolto la sua attività presso i Bell Labs.

Inventore del sistema [GROOVE](#) (Real-time Generated Operations On Voltage-controlled Equipment) un sistema ibrido (parte di controllo digitale e parte sonora analogica) per l'impossibilità di produrre il suono digitale in real-time



Sistema Groove

Il primo linguaggio per la generazione del suono con il computer è stato Music I seguito da forme evolutive di tale linguaggio Music II ... fino a **Music V** utilizzato da molti compositori di computer music negli anni 70. Questi linguaggi sono catalogati come linguaggi Music-n e ancora oggi si utilizza un discendente di tale linea denominato **CSOUND**

# Linguaggi Music-n

Lo sviluppo di tali linguaggi ha risentito, almeno nelle fasi iniziali dello sviluppo, del sistema informatico su cui venivano redatti. A titolo di esempio:

- **MUSIC I -II** IBM 704 a valvole– 32 Kbyte core mem. multiply in 250 usec Assembler - 1957 -Bell Labs
- **MUSIC III** IBM 7094 (transistori)
- .etc..
- **MUSIC V** IBM 360 - FORTRAN - 1968

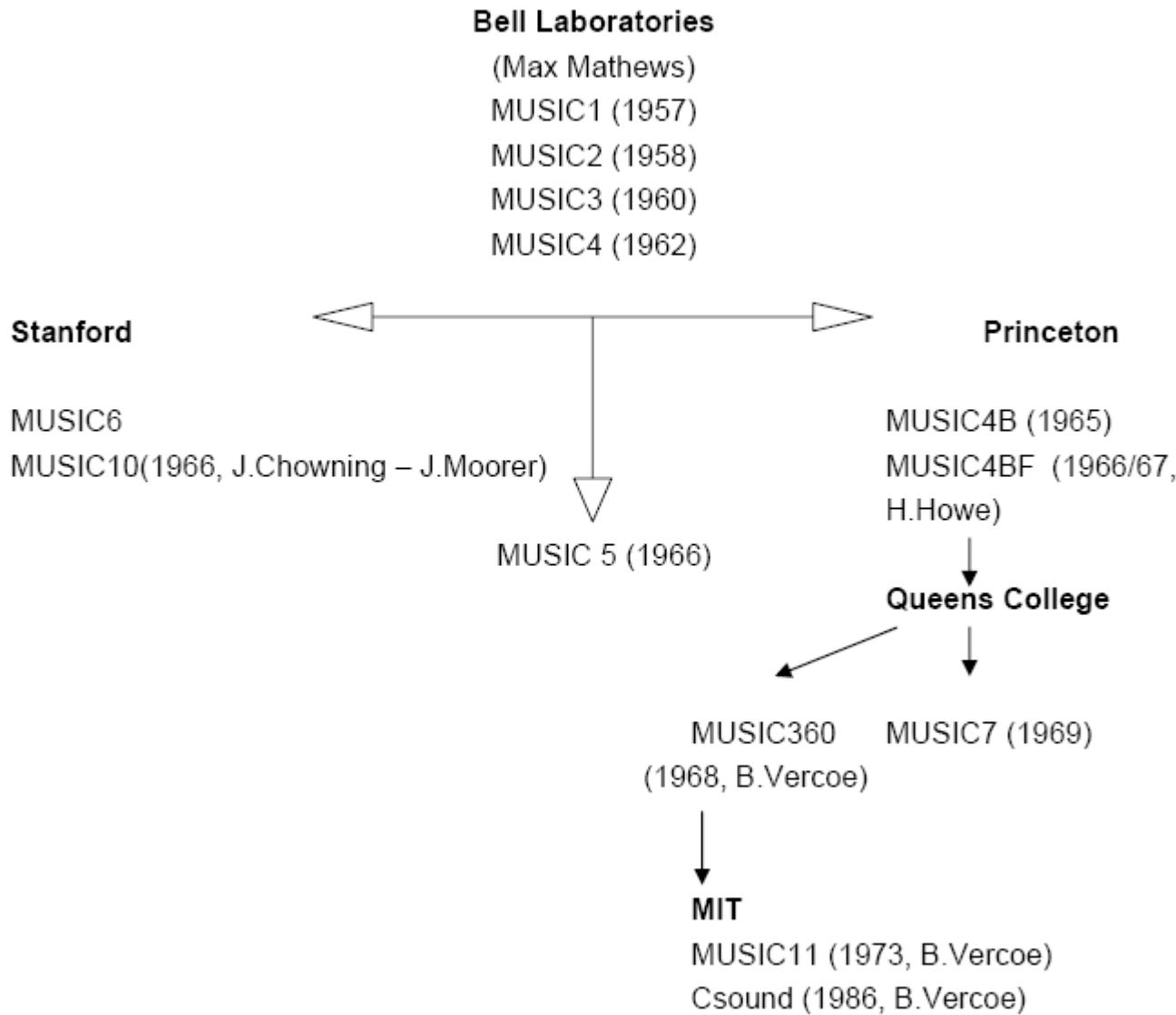


IBM 360



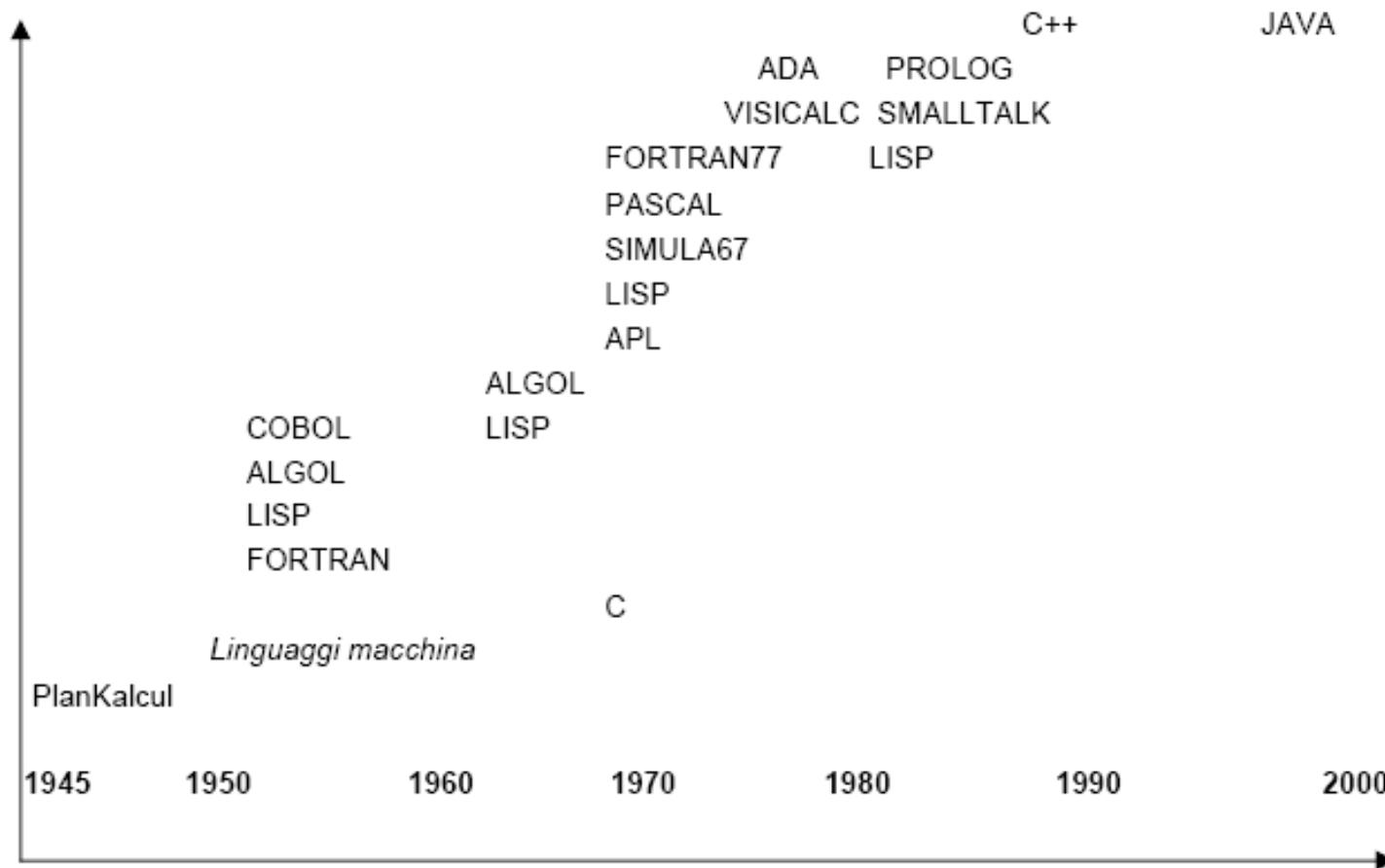
IBM 704

# Albero di sviluppo dei linguaggi di sintesi audio



I linguaggi di tipo Music-n sono veri e propri linguaggi di programmazione ma a differenza di questi ultimi che sono per uso generale (*general purpose*) e che necessitano di alti tempi di apprendimento, sono molto più semplici da utilizzare e molto potenti per le applicazioni specifiche dell'audio.

# Timeline dei linguaggi di programmazione general purpose



# Esigenze computazionali per la generazione sonora

Secondo uno standard medio il suono digitale viene prodotto con frequenza di campionamento pari a 44100. Ciò comporta per il sistema digitale:

1 secondo di suono → 44100 numeri binari a 16 bit per canale audio (standard CD audio)

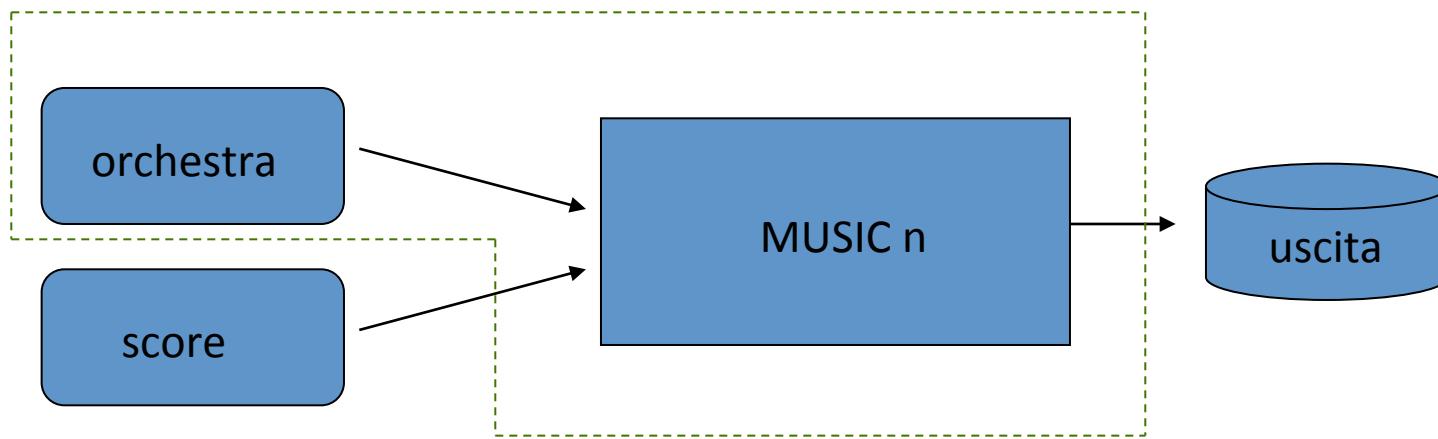
44100 è la quantità che esprime il SAMPLING RATE (frequenza di campionamento)

**Generazione differita:** viene eseguito il calcolo per la generazione/elaborazione ( 1 secondo di suono generato in un tempo  $T_{gen} > 1 \text{ sec}$  ) e successivamente convertito da numerico/analogico e riprodotto ove  $T_{gen}$  è il tempo di calcolo effettivo.

**Generazione in tempo reale:** viene eseguito il calcolo e contemporaneamente convertito ( 1 secondo generato in un tempo  $T_{gen} \leq 1 \text{ sec}$  )

# Struttura dei linguaggi Music-n

I linguaggi Music-n sono tutti basati su un paradigma di fondo che è dato dal dualismo delle struttura dati elaborata dal programma. In pratica si basa sulla descrizione di uno o più algoritmi di generazione/elaborazione del suono organizzati all'interno di un file descrittivo a caratteri ASCII denominato **ORCHESTRA** e su un secondo file (sempre ASCII) che definisce la temporizzazione degli strumenti e il controllo dei parametri d'esecuzione detto **SCORE** secondo lo schema indicato.



L'utente utilizza quindi un programma che è un **interprete di linguaggio** e che è in grado di decodificare le varie direttive e istruzioni specificate nei files relativi. La programmazione avviene quindi per **script** (scrivendo le varie istruzioni all'interno di un file editabile di tipo ASCII).

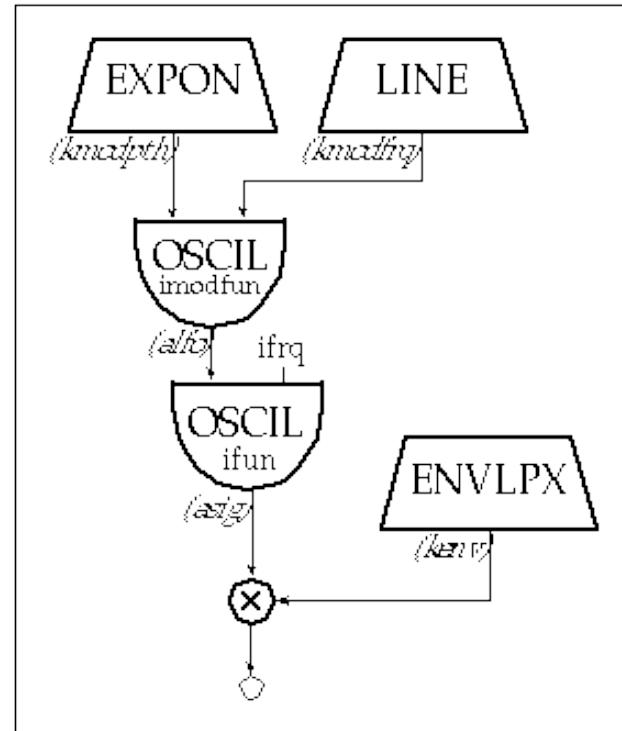
# Unità generative e connettività

L'ORCHESTRA contiene quindi, come detto, la descrizione dell'algoritmo (degli algoritmi) di sintesi ed elaborazione del suono che consiste principalmente nella combinazione di varie unità primitive di elaborazione o unità generative (oscillatori, filtri, inviluppi, operatori matematici, ecc..) attraverso un processo di connessione tra tali unità. Da un punto di vista sinottico, questa operazione può essere rappresentata in forma di schemi di flusso in cui le *ugens* (unità generative) sono raffigurate come box di varia forma da cui fuoriescono e/o entrano terminali di collegamento.

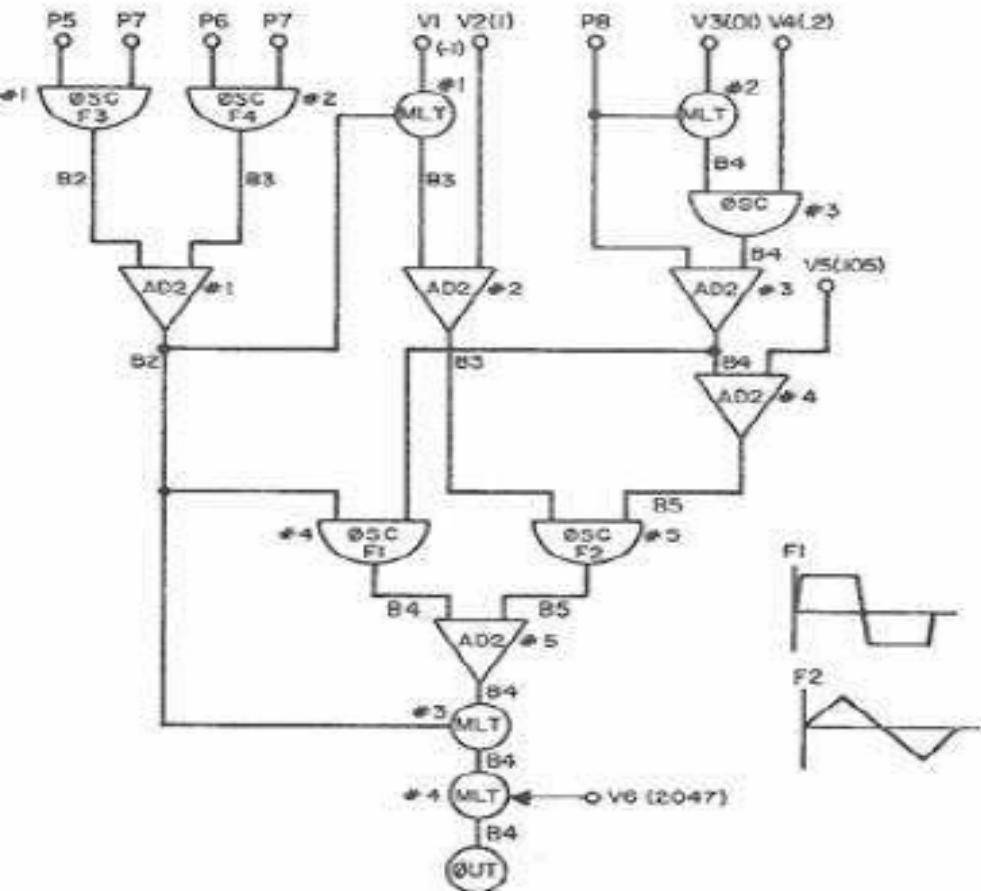
Nella figura sono visibili 3 generatori di inviluppo (*expon*, *line* e *envlpx*), 2 oscillatori (*oscil*) e un operatore prodotto (x)

L'uscita di *expon* controlla l'ampiezza del primo oscillatore mentre *line* controlla la sua frequenza. A sua volta il primo oscillatore controlla l'ampiezza del secondo mentre la sua frequenza rimane costante (*ifrq*). Infine l'uscita del secondo oscillatore viene moltiplicata per il terzo inviluppo *envlpx*.

Si noti come i collegamenti tra i moduli vengano indicati con nomi di variabili (*kmodpth*, *kmodfrq*, *alfa*, *ifrq*, *asig*, *kenv*). Tali nomi sono scelti arbitrariamente dal programmatore ma devono essere preceduti da lettere chiave ( a, k o i ) che ne identificano la tipologia di appartenenza:  
**a** : variabile che identifica flusso audio  
**k** : variabile che identifica flusso di controllo  
**i** : variabile che identifica inizializzazione



# Schema Music V e codifica



```

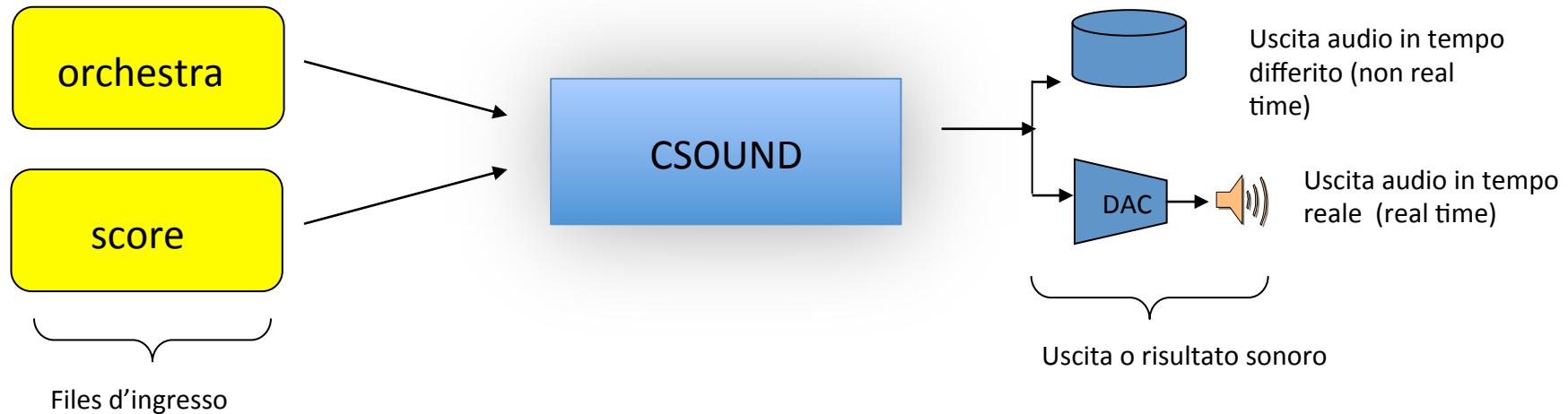
INS 0 4 ;
OSC P5 P7 B2 F3 P30 ;
OSC P6 P7 B3 F4 P29 ;
AD2 B2 B3 B2 ;
MLT B2 V1 B3 ;
AD2 B3 V2 B3 ;
MLT P8 V3 B4 ;
OSC B4 V4 B4 F5 P28 ;
AD2 P8 B4 B4 ;
AD2 B4 V5 B5 ;
OSC B3 B5 B5 F2 V7 ;
OSC B2 B4 B4 F1 V8 ;
MLT B2 B4 B4 ;
MLT B4 V6 B4 ;
OUT B4 B1 ;
END :

```

Questo esempio mostra come uno schema a blocchi costituito da varie ugens viene trasformato in uno script all'interno del linguaggio Music V. Come si può vedere, la codifica era di basso livello cioè con un tipo di linguaggio **machine oriented** e quindi non semplice da utilizzare. Nelle versioni attuali di tali programmi il linguaggio si è molto semplificato (**man oriented**)

# Csound

Csound proviene da uno sviluppo di Music V ed è stato concepito all'interno dei laboratori EMS (Electronic Music Studio) del MEDIA Lab MIT da [Barry Vercoe](#) (1985)



## Aspetti innovativi di Csound

- Ridefinisce la modalità di interpretazione dei dati d'ingresso
- Scritto in C, permette di usare script di programmazione C-like semplici all'uso
- Portabilità (da Unix a Dos , Windows, OS, OSX, Linux)
- Maggiore efficienza, migliorano e aumentano le UGs specializzate

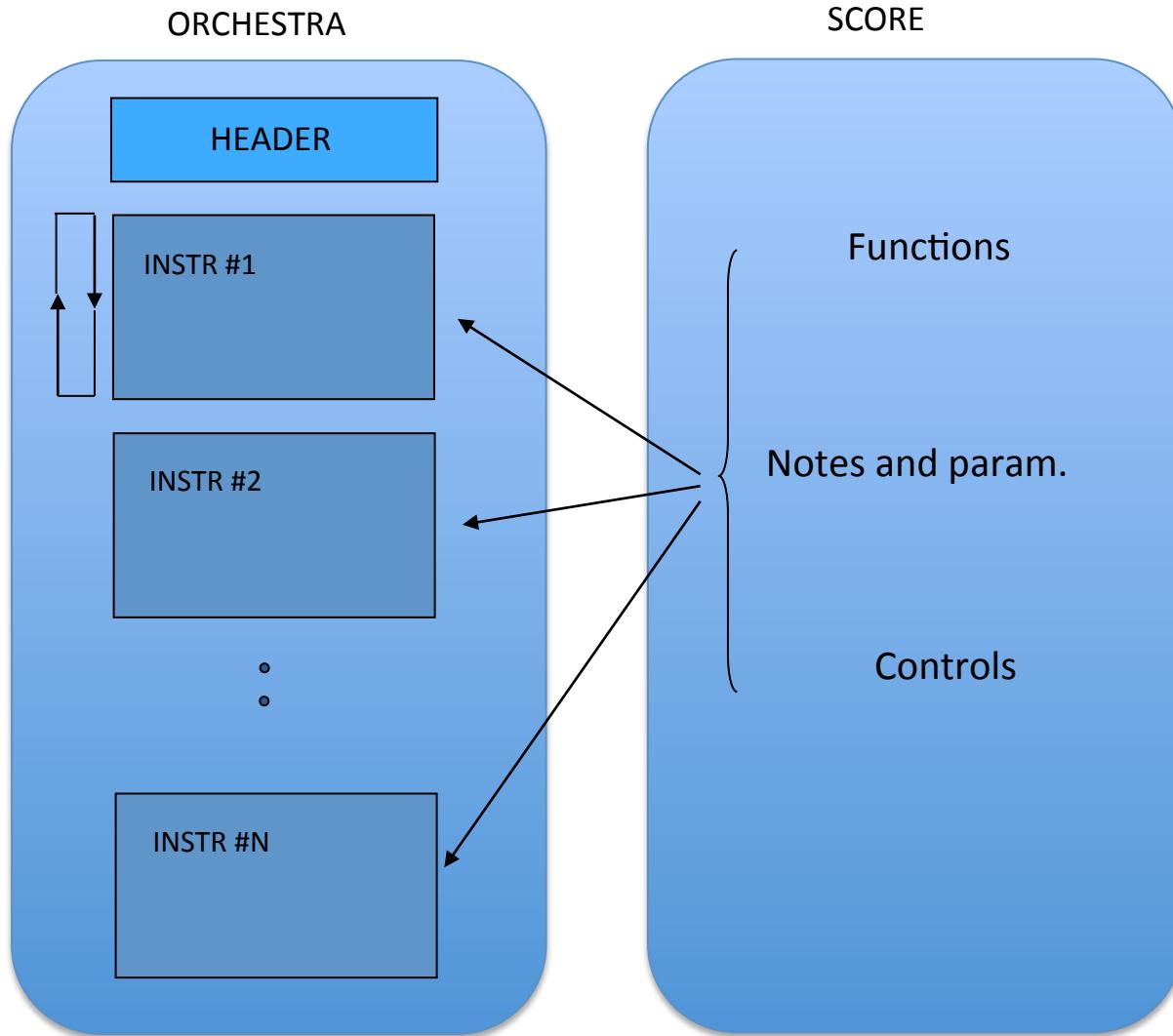
# Struttura dei file orchestra e score in Csound

L'ORCHESTRA è costituita dall'insieme degli strumenti che costituiscono la generazione/elaborazione del suono.

Prima degli strumenti viene specificato un HEADER che definisce alcuni parametri globali (freq. di campionamento, numero di canali, frequenza di controllo).

Ogni strumento è numerato e al suo interno si avvia un ciclo continuo di calcolo del suono: ogni giro equivale ad un frame di campionamento.

Nello SCORE trovano luogo invece la temporizzazione dei vari strumenti (Notes and param.) più controlli e funzioni.



# Aspetto del linguaggio Csound

```
; Initialize the global variables.  
sr = 44100  
kr = 441  
ksmps = 100|  
nchnls = 1  
  
; Instrument #1 - a basic oscillator.  
instr 1  
    kamp = 10000  
    kcps = 440  
    ifn = 1  
  
    a1 oscil kamp, kcps, ifn  
    out a1  
endin  
; Instrument #2.|  
instr 2  
    ; Generate a random number between 220 and 440.  
    kmin init 220  
    kmax init 440  
    k1 random kmin, kmax  
  
    printks "k1 = %f\\n", 0.1, k1  
endin
```

*NOTA : le parole e i simboli chiave sono mostrate in colore azzurro. I commenti in verde e preceduti da punto e virgola*

```
; Table #1, a sine wave.  
f 1 0 16384 10 1  
  
; Play Instrument #1 for 2 seconds.  
i 1 0 2  
; Play Instrument #2 for 5 seconds.  
i 2 1.5 5  
e
```

In questa pagina è mostrato il contenuto di un file ORCHESTRA (a sx) e un file SCORE (dx) indispensabili per fare funzionare l'interprete Csound.

Si possono individuare i due strumenti di cui è formata l'ORCHESTRA: *instr 1* e *instr 2* (si noti che la fine ogni strumento è delimitata dalla parola chiave *endin*).

L'ORCHESTRA contiene anche una intestazione (header) dove sono specificati parametri globali, come frequenza di campionamento (sr) o numero di canali (nchnls)

Il file di dx (SCORE) contiene le informazioni di attivazione dell'orchestra:

i1 0 2 significa attiva strum. 1 al tempo zero e durata 2 sec  
i2 1.5 5 “ “ strum. 2 al tempo 1.5 e durata 5 sec.

# Sviluppi nelle versioni moderne di Csound

- 1984 Barry Vercoe rilascia la prima versione di Csound (primo linguaggio di sintesi audio scritto interamente in C) (MIT)
- 1991 John fficht porta Csound sotto DOS (University of Bath)

Attualmente esistono versioni per tutte le principali piattaforme (Unix, Windows, OSX, Linux)

- Rispetto a molti software commerciali ha un numero molto superiore di unità generative
- Documentazione estesa
- Facile espandibilità
- Uso dell'aritmetica in doppia precisione per il calcolo

# Principali versioni di Csound

Sono disponibili molte versioni di Csound. Le più diffuse:

- CsoundAV            (Gabriel Maldonado)    (Windows) - non manutenuta
- Csound Linux        ( Nicola Bernardini)      (Linux)
- MacCsound           (Ma++Ingalls)              (OSX) - non manutenuta
- Csound5             (Michael Gogins)            Win/OSX
- **Csound6**            (M.Gogins, J. Ffitch et altri.) - versione più recente - 2013

# Frontend di Csound

I frontend sono programmi (applicazioni) che forniscono un'interfaccia grafica per il controllo e la programmazione di Csound. Possono permettere di avere tipicamente di aggiungere caratteristiche di aiuto: colorazione della sintassi, widgets grafici, o strumenti per la generazione algoritmica dello score, che non sono parte di Csound stesso.

- **CsoundQt** (Andres Cabrera) (Windows/OSX)

CsoundQt è una versatile, cross-piattaforma GUI (graphical user interface) che viene fornita in bundle con la distribuzione standard di Csound. Creata e manutenuta da Andres Cabrera, CsoundQt fornisce un multi-tabbed editor, widgets grafici per il controllo real-time del suono, e un sistema di help contestuale che è collegato al manuale. La versione più recente può essere scaricata a <http://qutecsound.sourceforge.net/>.

- **Blue** (Stephen Yi) (Windows/OSX)

E' un front end cross-piattaforma orientato alla composizione scritto by Steven Yi in Java. L'interfaccia grafica fornisce una struttura basata su una timeline in qualche modo simile ad un sistema multitraccia, ma differisce nel fatto che le timelines possono essere incluse timelines (polyObjects). Ciò consente un tipo di organizzazione compositiva nel tempo che molti utilizzatori trovano intuitiva, informativa e flessibile. Ciascuno strumento e sezione di score, in un progetto Blue ha la propria finestra di editing, che rende più semplice l'organizzazione di grandi progetti. Blue può essere scaricato da [Blue Home Page](#).

- **WinXsound** (Stefano Bonetti) (Windows/OSX/Linux)

E' un efficiente front-end multipiattaforma con hilight della sintassi. WinXsound può essere scaricato da [WinXsound Front Page](#).

# Interfaccia grafica (GUI) basata su FLTK



[SimpleKnob.csd](#)



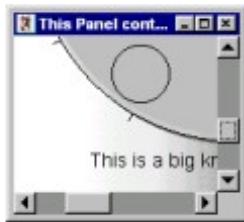
[simpleButton.csd](#)



[ShowValues.csd](#)



[simpleSlider.csd](#)



[simpleScroll.csd](#)



[ModifyAppearance.csd](#)



[simpleRoller.csd](#)



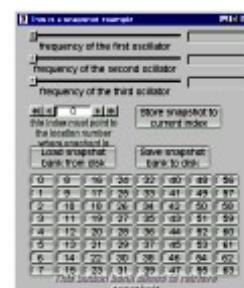
[simpleJoystick.csd](#)



[simpleTabs.csd](#)



[simpleButtonBank.csd](#)



[Savesettings.csd](#)



[FLprintk.csd](#)

Viene impiegato un subset della libreria grafica Fast Light Tool Kit (open source) FLTK che mette a disposizione un elevato numero di "widget" (componente grafico di una interfaccia utente)

## PLUS

- veloce
- supporta lo standard OpenGL
- compatibile con tutte le piattaforme

## MINUS

- occorre scrivere codice per richiamare i widgets grafici



[simpleTextField.csd](#)

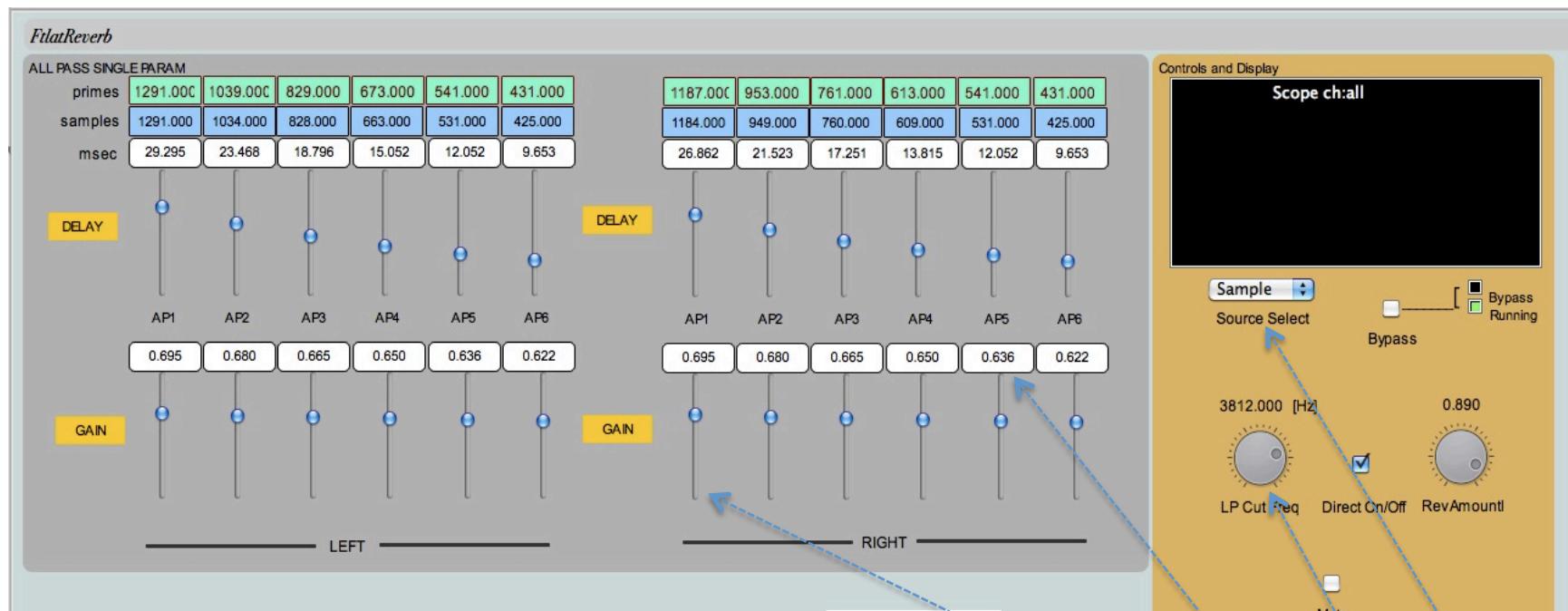


Activate motor 2 and modify its frequency with this Counter

# Interfaccia grafica (GUI) basata su Qt

Attraverso l'uso di una libreria grafica open-source Qt (sviluppata prevalentemente da Nokia) è possibile progettare un'interfaccia utente piazzando rapidamente attraverso il mouse una serie di widgets predefiniti. Tali oggetti possono essere facilmente collocabili, ridimensionabili e personalizzabili (varianti di forma, colore, funzione).

Una delle ultime versioni di Csound è basata sull' engine di generazione (**Csound versione 5**) e su tale interfaccia grafica. In rete è disponibile gratuitamente sotto il pacchetto unico denominato **CsoundQt** (precedentemente apparso col nome di Qtcsound)



Parte di una interfaccia utente costruita in ambiente CsoundQt. Sono visibili "slider" verticali, "displays", "knobs", "menu" e molti altri.

# Caratteri innovativi e sviluppi di Csound5 (ver. 5.xx)

- Licenziato sotto GNU Lesser General Public License (Licenza Open Source)
- Utilizzazione di librerie open source:
  1. libsndfile for soundfile input and output.
  2. PortAudio with ASIO drivers for low-latency, real-time audio input and output.
  3. FLTK for graphical widgets that can be programmed in orchestra code.
  4. PortMidi for real-time MIDI input and output.
- Sistema di bufferizzazione audio semplificato
- MIDI interop opcodes (intercambiabilità flusso dati MIDI)
- Uso di opcode (UG) di tipo “plug in”  
(Python richiamabile all’interno)  
(STK di Perry Cook per la modelazione fisica) ...etc...
- Csound API includibili in C/C++/Python/Java/Lisp
- Rientrante: multiple istanze girano sullo stesso processo
- Riscrittura di un nuovo parser attraverso un *parser generator (j. fficht)*

# Csound6

- Licenziato sotto GNU Lesser General Public License (Licenza Open Source)
- Utilizzazione di librerie open source:
  1. libsndfile for soundfile input and output.
  2. PortAudio with ASIO drivers for low-latency, real-time audio input and output.
  3. FLTK for graphical widgets that can be programmed in orchestra code.
  4. PortMidi for real-time MIDI input and output.
- Sistema di bufferizzazione audio semplificato
- MIDI interop opcodes (intercambiabilità flusso dati MIDI)
- Uso di opcode (UG) di tipo “plug in”  
(Python richiamabile all’interno)  
(STK di Perry Cook per la modelazione fisica) ...etc...
- Csound API includibili in C/C++/Python/Java/Lisp
- Rientrante: multiple istanze girano sullo stesso processo
- Riscrittura di un nuovo parser attraverso un *parser generator* (*j. fficht*)