ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

EFFETTI AUDIO DIGITALI LINEARI PER STRUMENTI ELETTRICI A CORDA

Relazione finale in Mobile Web Design

Relatore
MIRKO RAVAIOLI

Presentata da
SILVIO OLIVASTRI

Sessione II

Anno Accademico 2013/2014

INDICE

INTRODUZIONE

L'obbiettivo di questa tesi è quello di studiare le tecnologie e i metodi necessari alla simulazione degli effetti audio lineari, normalmente utilizzati per strumenti a corda elettrici, ed implementarla sullo smartphone. I vantaggi di questa idea sono evidenti nella versatilità e nella comodità di utilizzo, rispetto ai classici dispositivi impiegati dai musicisti (come gli effetti a pedali).

Per fare ciò è necessaria la conoscenza delle tecniche di rappresentazione digitale di un segnale, come la trasformata di Fourier, il processo di campionamento e la trasformata Z, esposte nel Capitolo 1. Il Capitolo 2 continua l'introduzione trattando dei metodi utilizzati per creare effetti audio lineari tramite lo studio dei filtri FIR e IIR.

Nel capitolo 3 sarà disponibile una classificazione degli effetti più utilizzati, seguiti dal procedimento di sviluppo di due sistemi: un equalizzatore a 10 bande e un delay, che saranno implementati nello smartphone.

L'ultimo Capitolo, il quarto, spiega come è sviluppato il progetto, perché iOS è l'unico sistema operativo che permetta di farlo, ed indica le principali classi che necessitano di essere utilizzate.

CAPITOLO 1
RAPPRESENTAZIONE DIGITALE DEI SEGNALI AUDIO

La comprensione dei metodi utilizzati per creare effetti audio lineari esige la conoscenza di base delle tecniche di rappresentazione digitale di un segnale audio. Tale studio è analogo a quello della teoria dei segnali e a quello della elaborazione numerica le quali verranno introdotte in questo capitolo. Laddove non sarà specificato, le considerazioni svolte per un segnale generico saranno valide anche per il segnale audio specifico.

1.1 I NUMERI COMPLESSI

I numeri complessi (il cui insieme è denominato $\mathbb{C}$) sono un'estensione dei numeri reali, permettono di determinate equazioni non risolvibili in $\mathbb{R}$, ad esempio: $x^2 + 1 = 0$. Tale estensione viene introdotta inserendo la variabile immaginaria j, che consente di rappresentare un numero complesso come la somma di una parte reale a ed una immaginaria jb (1.1.1), con $a, b \epsilon \mathbb{R}$ e $c \epsilon \mathbb{C}$. [S6]

(1.1.1) $$c = a + jb$$

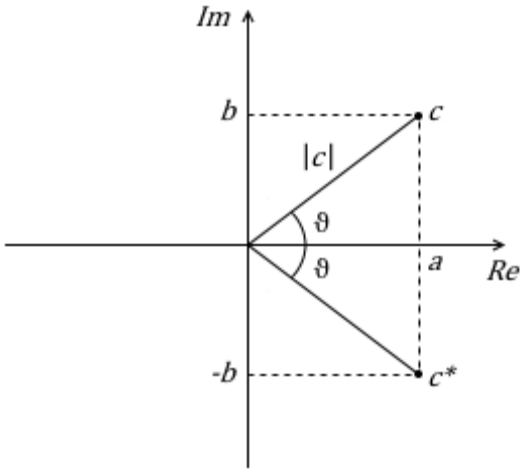

Figura 1.1.1: Rappresentazione cartesiana di numero complesso

La Fig.1.1.1 descrive la 1.1.1 nel piano cartesiano complesso chiamato anche piano di Argand-Gauss, in cui viene definito anche il coniugato c^*. La somma di un numero complesso con il suo coniugato restituisce il numero reale $2a$ (1.1.2). [S6]

(1.1.2) $$c + c^* = a + jb + a - jb = 2a$$

Sia ora la formula di Eulero enunciata in 1.1.3,

(1.1.3) $$e^{j\vartheta} = \cos\vartheta + j\sin\vartheta$$

è possibile esprimere un numero complesso in forma polare come in 1.1.4, la cui formula permette di identificare modulo (1.1.5) e fase (1.1.6) di c. [S6]

$$c = a + jb = |c|(\cos\vartheta + j\sin\vartheta) = |c|e^{j\vartheta} \Rightarrow$$

(1.1.4) $$c = |c|e^{j\vartheta}$$

(1.1.5) $$|c| = \sqrt{a^2 + b^2} = \sqrt{c \cdot c^*}$$

(1.1.6) $$\vartheta = arg\{c\} = \begin{cases} arctg\left(\frac{b}{a}\right) & a > 0 \\ \pi + arctg\left(\frac{b}{a}\right) & a < 0 \end{cases}$$

1.2 DEFINIZIONE E CLASSIFICAZIONE DI SEGNALE

Con "segnale" si indica una grandezza fisica che varia nel tempo. La natura di tale fenomeno può essere diversa: la pressione sonora rilevata da un microfono, l'accelerazione di un corpo, l'intensità luminosa di una scena televisiva. Nel campo dell'audio ci occuperemo sostanzialmente di dati prodotti da uno strumento o da un microfono, rappresentati dalla tensione elettrica al variare nel tempo. [B5][S7] E' possibile suddividere i segnali in due categorie: segnali determinati e segnali aleatori. I primi sono rappresentabili (o

approssimabili) con funzioni matematiche che ne determinano l'andamento in ogni instante; i secondi, dei quali sono note soltanto alcune grandezze statistiche, necessitano del ricorso a modelli probabilistici. Di norma, quest'ultima differenziazione non impone vincoli nello studio degli effetti audio perché, più che studiarne l'andamento, ci si preoccupa di capire come modificare il segnale, per cui ci si riferirà ad esso indipendentemente dalla categoria d'appartenenza. [B5]

Un segnale può essere espresso come una funzione $x(t)$ che può assumere tutti i valori appartenenti ad un certo intervallo o solamente alcuni, analogamente t (il tempo) può essere una successione numerabile di valori oppure innumerabile (continua), per cui possiamo classificare i segnali come:

a) Funzioni continue nei valori e tempo-continue
b) Funzioni discrete nei valori e tempo-continue
c) Funzioni continue nei valori e tempo-discrete
d) Funzioni discrete nei valori e tempo-discrete. [B1]

1.3 RAPPRESENTAZIONE DEI SEGNALI TEMPO-CONTINUI

La frequenza (f) è una grandezza rappresentativa della ripetizione di un evento. Un altro modo in cui è possibile esprimere questa variabile è la pulsazione (1.3.1).

(1.3.1) $$\omega = 2\pi f$$

Essendo f moltiplicata solo per una costante, la 1.3.1 esprime comunque una frequenza, perciò è possibile riferirsi ad essa come tale.

A seconda delle applicazioni, i segnali possono essere processati nel dominio tempo o nel dominio delle frequenze. E' importante, quindi, conoscere gli strumenti matematici principali che permettano di

passare da un dominio all'altro, ovvero, la trasformata di Fourier (1.3.2) e l'antitrasformata di Fourier (1.3.3). [B1]

(1.3.2) $$X(\omega) = \int_{-\infty}^{+\infty} x(t)\, e^{-j\omega t} dt$$

(1.3.3) $$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\omega)\, e^{j\omega t} d\omega$$

$$X(\omega), x(t) \in \mathbb{C}$$

E' importante precisare che tra $X(\omega)$ e $x(t)$ è valida la teoria dell'unicità della trasformata, ovvero che la corrispondenza tra le due funzioni è biunivoca. Non è possibile, infatti, che un segnale abbia più trasformate o che si possano ricavare più funzioni nel dominio del tempo da una delle frequenze. [B1]
Se $x(t) \in \mathbb{R}$, come sempre accade nei segnali audio, allora è applicabile la Simmetria Hermitiana (1.3.4),

(1.3.4) $$X(-\omega) = X^*(\omega) \Rightarrow \begin{cases} |X(-\omega)| = |X(\omega)| \\ arg\{X(-\omega)\} = -arg\{X(\omega)\} \end{cases}$$

ciò permette, ricordando che $X(\omega) \in \mathbb{C}$ ed applicando ad essa la 1.1.3 e la 1.1.4, di delineare una nuova forma della 1.3.3:

$$x(t) = Re\left\{\frac{1}{2\pi}\int_{-\infty}^{+\infty} X(\omega)\, e^{j\omega t} d\omega\right\} = Re\left\{\frac{1}{2\pi}\int_{-\infty}^{+\infty} |X(\omega)|\, e^{j arg\{X(\omega)\}} e^{j\omega t} d\omega\right\} =$$

$$= \frac{1}{2\pi} Re\left\{\int_{-\infty}^{+\infty} |X(\omega)| \cos(arg\{X(\omega)\} + \omega t)\, d\omega + j\int_{-\infty}^{+\infty} |X(\omega)| \sin(arg\{X(\omega)\} + \omega t)\, d\omega\right\} =$$

$$= \frac{1}{2\pi} 2 \int_{0}^{+\infty} |X(\omega)| \cos(arg\{X(\omega)\} + \omega t)\, d\omega$$

ponendo $V(\omega) = \frac{|X(\omega)|}{\pi}$ (densità spettrale di ampiezza) e $\varphi(\omega) = -arg\{X(\omega)\}$ (spettro di fase) con $\omega \geq 0$ si ottiene la relazione chiamata "Integrale di Fourier" (1.3.5) che è di fondamentale importanza perché permette di identificare non solo il concetto di "spettro di un segnale" ($V(\omega)$

o $\varphi(\omega)$) ma anche di banda B_ω, ovvero la parte (o le parti) dove lo spettro non è nullo. Un esempio è esposto in Fig.1.3.1. [B1]

(1.3.5) $$x(t) = \int_0^{+\infty} V(\omega)\cos\big(\omega t - \varphi(\omega)\big)\, d\omega$$

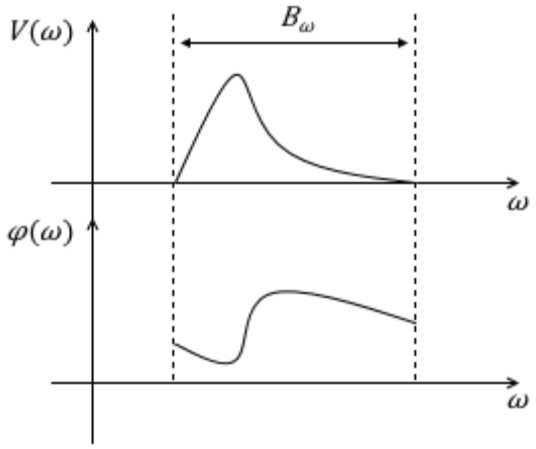

Figura 1.3.1: Esempio di banda (B_ω), spettro in ampiezza ($V(\omega)$) e spettro in fase ($\varphi(\omega)$) di un segnale reale.

Nel campo dei sistemi dinamici tempo-continui, che permettono la realizzazione degli effetti audio lineari analogici, si preferisce usare una generalizzazione della 1.3.2 ottenibile introducendo una nuova variabile complessa: $s = j\omega$.

(1.3.6) $$X(s) = \int_{-\infty}^{+\infty} x(t)\, e^{-st} dt$$

La 1.3.6 è definita come trasformata di Laplace. [S9]

1.4 RAPPRESENTAZIONE DEI SEGNALI TEMPO-DISCRETI

I segnali possono essere in forma analogica o numerica. E' possibile digitalizzare un segnale analogico tramite la conversione A/D (da analogico a digitale). Questa tecnica permette a un calcolatore, come

ad esempio lo smartphone, di prendere e sintetizzare i dati da una fonte come quella di un microfono. Una volta elaborato, il segnale potrà essere ricondotto in analogico tramite conversione D/A (da digitale ad analogico) e spedito agli altoparlanti che permetteranno all'utilizzatore di percepire il suono. In questa tesi saranno modellati i segnali audio in forma numerica, per cui è importante capire i tre procedimenti della conversione A/D.
La prima fase è il campionamento e consiste nel prelevare, ad intervalli regolari di tempo T, i campioni dal segnale tempo-continuo. Il risultato sarà di trasformare $x(t)$ in un segnale tempo-discreto $x_c(nT)$ chiamato "serie temporale" (Fig.1.4.1). [B1]

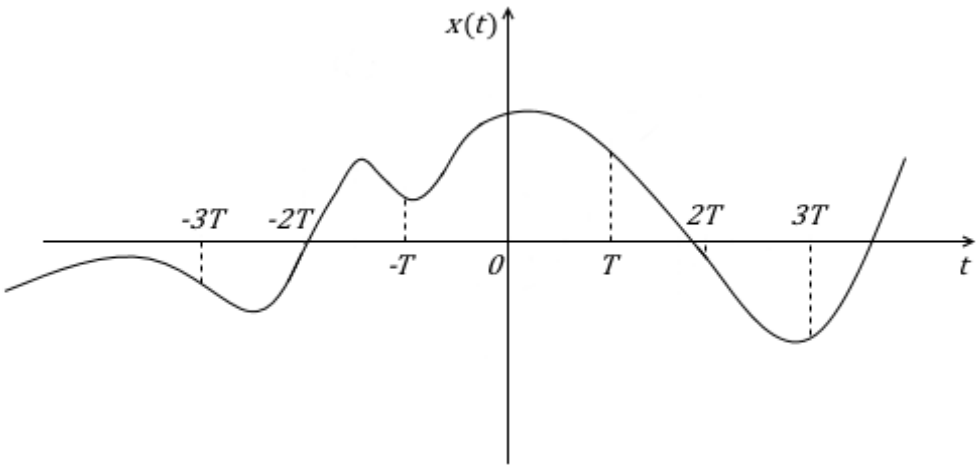

Figura 1.4.1: La successione dei punti presi ad ogni istante T dalla funzione $x(t)$ formano la serie temporale $x_c(nT)$.

In questa operazione è importante enunciare il Teorema di Shannon il quale asserisce che la conoscenza della serie temporale ottenuta da un campionamento equivale a quella della funzione campionata se (condizione sufficiente) la frequenza di campionamento ω_c è almeno il doppio della frequenza massima del segnale campionato ω_m. Se il teorema è soddisfatto, è possibile applicare la relazione dello sviluppo in serie di Shannon (1.4.1), altrimenti se $\omega_c < 2\omega_m$, $x(t)$ non sarà più ricomponibile dalla serie temporale $x_c(nT)$ il che pregiudica l'uscita del sistema in esame. Tale condizione, che deve essere

assolutamente evitata, è chiamata "distorsione da campionamento lento" (aliasing). [B1]

(1.4.1) $$x(t) = \sum_{n=-\infty}^{+\infty} x_c(nT)\, sinc\left(\frac{t - nT}{T}\right)$$

Alla fase di campionamento ne consegue quella di quantizzazione. Affichè il segnale venga convertito in digitale, e quindi in bit, non è possibile che la fascia numerabile dei valori sia infinita, poiché i calcolatori operano su cifre finite. Per cui, in questo procedimento si delimita un intervallo globale $[-M, M]$ suddividendolo in L porzioni (o intervalli di quantizzazione), a seguito del quale ogni valore di $x_c(nT)$ sarà identificato come il valor medio $x(nT)$ dell'intervallo di quantizzazione in cui ricade (Fig.1.4.3). [B1]

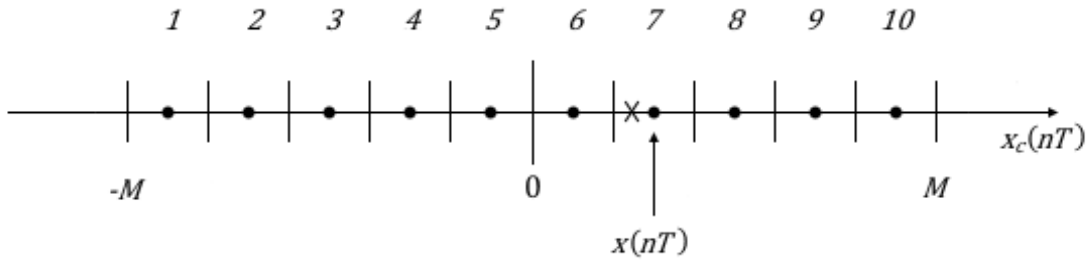

Figura 1.4.3: Esempio di quantizzazione con $L = 10$. Si osserva che il valore campionato $x_c(nT)$ (segnato con una "x") sarà sostituito dal valore quantizzato $x(nT)$ che approssima un certo intervallo di valori continui.

Questo arrotondamento provoca inevitabilmente una perdita d'informazione conosciuta come "errore di quantizzazione".
Come ultima fase, la codifica, porterà a convertire in bit tutti i valori quantizzati $x(nT)$, il segnale risultante da questo processo è chiamato Pulse Code Modulation (PCM). [B1]
La 1.3.2 è valida per segnali tempo-continui, ma per analogia, ponendo $t = nT$ si ottiene la 1.4.2 chiamata trasformata di Fourier tempo-discreta, in inglese Discrete-Time Fourier Transform (DTFT). [B1]

(1.4.2) $$X_s(\omega) = \sum_{n=-\infty}^{+\infty} x(nT)\, e^{-j\omega nT}$$

Per completezza l'antitrasformata è espressa in 1.4.3.

(1.4.3) $$x(nT) = \frac{T}{2\pi} \int_{-\frac{\pi}{T}}^{+\frac{\pi}{T}} X_s(\omega) e^{j\omega nT}\, d\omega$$

Un'altro strumento molto utilizzato per l'analisi dei segnali campionati è la trasformata Z definita in 1.4.4, ed essendo T costante da ora in avanti verrà applicata la sostituzione formale $x(n) = x(nT)$.

(1.4.4) $$X(z) = Z\{x(n)\} = \sum_{n=-\infty}^{+\infty} x(n)\, z^{-n}$$

Si noti che per $z = e^{j\omega T}$ si ottiene la 1.4.2, per cui la trasformata Z è una generalizzazione della trasformata di Fourier per segnali tempo-discreti. L'esistenza della trasformata Z è delimitata dalla sua regione di convergenza, definita per un certo $z = r\, e^{j\alpha}$ se $|X(z)| < \infty$ sul piano complesso di Argand-Gauss (Fig.1.1.1). La trasformata inversa della 1.4.4 è enunciata in 1.4.5,

(1.4.5) $$x(n) = Z^{-1}\{X(z)\} = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz$$

dove l'integrale è calcolato in senso antiorario su un contorno C, interno alla regione di convergenza che contiene l'origine $(z = 0)$.
Una proprietà importante, che viene citata perché utile nei capitoli che seguiranno, è quella della transazione temporale in 1.4.6.[B3][S9]

$$X'(z) = \sum_{n=-\infty}^{+\infty} x(n-k)\, z^{-n}$$

$$m = n - k \Rightarrow$$

$$X'(z) = \sum_{m=-\infty}^{+\infty} x(m)\, z^{-m}\, z^{-k} = z^{-k}X(z) \Rightarrow$$

(1.4.6) $$X(z)z^{-k} = Z\{x(n-k)\}$$

1.5 ALGORITMI PER CALCOLATORI

Le soluzioni di rappresentazione spettrale studiati nei paragrafi 1.3 e 1.4 non sono implementabili nel calcolatore perché il loro intervallo non è finito. La trasformata discreta di Fourier, anche detta DFT (dall'acronimo inglese Discrete Fourier Transform), permette di lavorare con un numero finito di valori, rendendo applicabile la teoria enunciata finora.
Data la successione finita di N numeri complessi $X_n = \{x_0, x_1, \dots, x_{N-1}\}$ e la trasformata di questa successione $X_q = \{x'_0, x'_1, \dots, x'_{N-1}\}$, la funzione che determina tale trasformazione è la 1.5.1.

(1.5.1) $$x'_q = \sum_{n=0}^{N-1} x_n\, e^{-jnq\frac{2\pi}{N}} \qquad 0 \le q \le N-1$$

L'antitrasformazione è data dalla 1.5.2.

(1.5.2) $$x_n = \frac{1}{N}\sum_{q=0}^{N-1} x'_q\, e^{jnq\frac{2\pi}{N}} \qquad 0 \le n \le N-1$$

La 1.5.1 e la 1.5.2 hanno un costo computazionale di $O(N^2)$, per cui sono stati sviluppati metodi più veloci ed efficaci che diminuiscono il costo a $O(N\log(N))$. L'algoritmo in questione si chiama "Fast Fourier Transform" o anche conosciuto come FFT. [B1]
I metodi di applicazione delle DFT o FTT vanno oltre l'argomentazione di questa tesi, quindi per ulteriori approfondimenti si rimanda a [B3].

CAPITOLO 2
FILTRI LIEARI DIGITALI

Una volta definiti i metodi di rappresentazione di un segnale, è possibile discutere delle basi per comprendere come un suono può essere trasformato linearmente. Tramite i sistemi lineari tempo-invarianti, aventi la caratteristica principale di non distorcere in frequenza il segnale in ingresso, è possibile definire delle strutture apposite che sintetizzano il suono in vari modi: i filtri lineari. Essendo i segnali considerati nel progetto tempo-discreti, l'analisi sarà effettuata spesso tramite variabili tempo-discrete (anche se è possibile fare considerazioni analoghe per variabili tempo-continue).

2.1 DEFINIZIONE DI SISTEMA LTI

Un sistema lineare discreto è un blocco che accetta una sequenza di valori discreti in ingresso $x(n)$ e produce in uscita un'altra sequenza $y(n) = T[x(n)]$, come è indicato in Fig.2.1.1, dove $T[.]$ indica una trasformazione lineare. [B1]

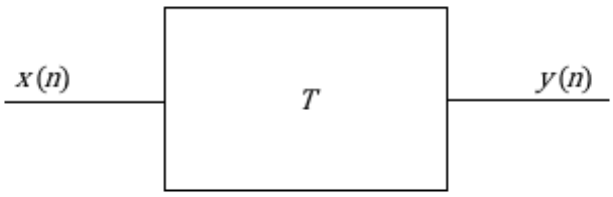

Figura 2.1.1: Sistema lineare tempo-discreto

Un sistema si dice lineare se, dati due ingressi qualsiasi $x_1(n)$ e $x_2(n)$ tali che $y_1(n) = T[x_1(n)]$ e $y_2(n) = T[x_2(n)]$, e un terzo ingresso $x_3(n) = c_1x_1(n) + c_2x_2(n)$ con c_1 e c_2 costanti, vale la proprietà descritta in 2.1.1. [B1]

(2.1.1)
$$y_3(n) = T[c_1x_1(n) + c_2x_2(n)] = \\ = c_1T[x_1(n)] + c_2T[x_2(n)] = c_1y_1(n) + c_2y_2(n)$$

Un sistema si dice tempo-invariante se, dato un segnale d'ingresso $x(n)$ tale che $y(n) = T[x(n)]$ per qualsiasi costante k e dato il segnale ritardato $x_1(n) = x(n+k)$ si ha che

(2.1.2)
$$y_1(n) = T[x_1(n)] = \\ = T[x(n+k)] = y(n+k)$$

Se un sistema rispetta la 2.1.1 e la 2.1.2 allora è lineare tempo-invariante (LTI). [B1]

2.2 FUNZIONE DI TRASFERIMENTO E RISPOSTA IMPULSIVA

Sia $\delta(n)$, determinato come in 2.2.1,

(2.2.1)
$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$

un impulso chiamato "Delta di Dirac", si definisce "Risposta impulsiva" (2.2.2) di un sistema LTI l'uscita avente tale impulso in ingresso. [B1]

(2.2.2)
$$h(n) = T[\delta(n)]$$

Il ruolo della risposta impulsiva è quello di caratterizzare il comportamento ingresso-uscita di un sistema nel dominio del tempo, infatti, un ingresso qualsiasi $x(n)$ può essere visto come il prodotto di convoluzione tra se stesso e $\delta(n)$ (2.2.3),

(2.2.3)
$$x(n) = x(n) * \delta(n) = \sum_{i=-\infty}^{+\infty} x(i)\delta(n-i)$$

per cui l'uscita $y(n)$, può essere ricavata tramite la convoluzione fra l'ingresso e la risposta impulsiva (2.2.4). [B1]

$$y(n) = T[x(n)] = T[x(n) * \delta(n)] = T\left[\sum_{i=-\infty}^{+\infty} x(i)\delta(n-i)\right] = \tag{2.2.4}$$
$$= \sum_{i=-\infty}^{+\infty} x(i)T[\delta(n-i)] = \sum_{i=-\infty}^{+\infty} x(i)h(n-i) = x(n) * h(n)$$

Nei sistemi ideali, la risposta impulsiva può essere definita per tutto l'asse dei tempi, ma per quanto riguarda i sistemi realizzabili fisicamente la risposta impulsiva è definita solo per $n > 0$ (nel caso continuo per $t > 0$). Questo perché i sistemi reali soddisfano la condizione di causalità, ovvero l'uscita $y(n_0)$ dipende solo dai valori d'ingresso $x(n)$ per $n \leq n_0$. Se non si rispetta tale condizione il sistema non è realizzabile perché implica che si possa avere risposta ad un impulso prima che esso entri in ingresso. [B1]
Tramite il teorema della convoluzione (dimostrato in []), il quale afferma che la trasformata del prodotto di convoluzione di due funzioni, è il prodotto delle rispettive trasformate (2.2.5),

$$Y(z)\,X(z) = Z\{x(n) * y(n)\} \tag{2.2.5}$$

dalla 2.2.4, è possibile denominare la "Risposta in frequenza" o "Funzione di trasferimento" (F.d.t.) del sistema applicando la trasformata Z all'uscita (2.2.6). [B1]

$$Y(z) = Z\{y(n)\} = Z\{x(n) * h(n)\} = X(z)H(z) \Rightarrow$$
$$H(z) = \frac{Y(z)}{X(z)} \tag{2.2.6}$$

La F.d.t può essere definita anche come la trasformata Z della 2.2.2, per cui è possibile esprimerla come in 2.2.7. [B1]

(2.2.7) $$H(z) = Z\{h(n)\} = \sum_{n=-\infty}^{+\infty} h(n)\, z^{-n}$$

La F.d.t caratterizza il sistema LTI nel dominio delle frequenze.
Si possono effettuare tutti i ragionamenti fatti finora nel dominio continuo delle pulsazioni, in questo caso la Funzione di trasferimento sarà indicata come $H(\omega)\,\epsilon\,\mathbb{C}$.

2.3 SISTEMA NON DISTORCENTE E FILTRI BASE ANALOGICI

Si intende per distorsione un fenomeno dannoso e indesiderato a causa del quale un segnale viene a perdere alcune delle sue caratteristiche originarie.
Un sistema si dice non distorcente se, dato un ingresso $x(n)$, l'uscita $y(n)$ è modificata soltanto in apiezza $A\,\epsilon\,\mathbb{R}$ e risulta traslata di un valore finto $k > 0$ (2.3.1).

(2.3.1) $$y(n) = A\, x(n \pm k)$$

Nel caso in cui il segnale e il sistema sono continui nel tempo si possono applicare le seguenti sostituzioni alla 2.1.3: $y(n) \to y(t)$, $k \to t_0$, $x(n \pm k) \to x(t \pm t_0)$. [B1]
Ciò implica che il sistema non modifica l'andamento in frequenza ma solo quello in fase e ampiezza. Un sistema LTI ideale presenta questa caratteristica (valida sono nella banda della F.d.t), anche se in realtà tutti i sistemi lineari realizzati fisicamente presentano una distorsione seppur minima o trascurabile.
Si prenda in considerazione un sistema LTI tempo-continuo non distorcente in una certo intervallo di banda (banda passante) dove al difuori di essa lo spettro è nullo (banda attenuata). Un segnale che attraversa tale sistema verrà riprodotto indistorto solo nella parte che coincide con quella di banda passante, infatti, lo spettro uscente sarà nullo al difuori di essa. Tale sistema è definito filtro e consente la separazione di un segnale utile dai disturbi più vari

sovrapposti al segnale originario. Esistono quattro tipi di filtri base ideali:

1) Passa Basso (Fig.2.3.1 a)
2) Passa Alto (Fig.2.3.1 b)
3) Passa Banda (Fig.2.3.2 a)
4) Elimina banda (Fig.2.3.2 b)

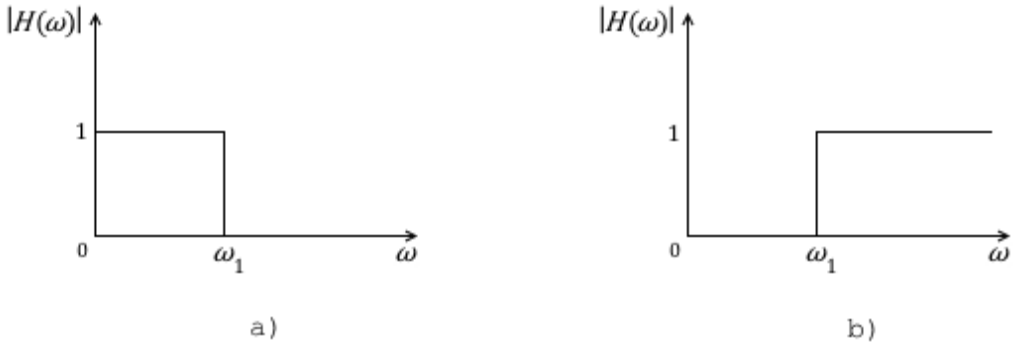

Figura 2.3.1: a) Filtro ideale passa basso, banda passante $[0,\omega_1]$, banda attenuata $[\omega_1,+\infty]$. b) Filtro ideale passa alto, banda passante $[\omega_1,+\infty]$ banda attenuata $[0,\omega_1]$.

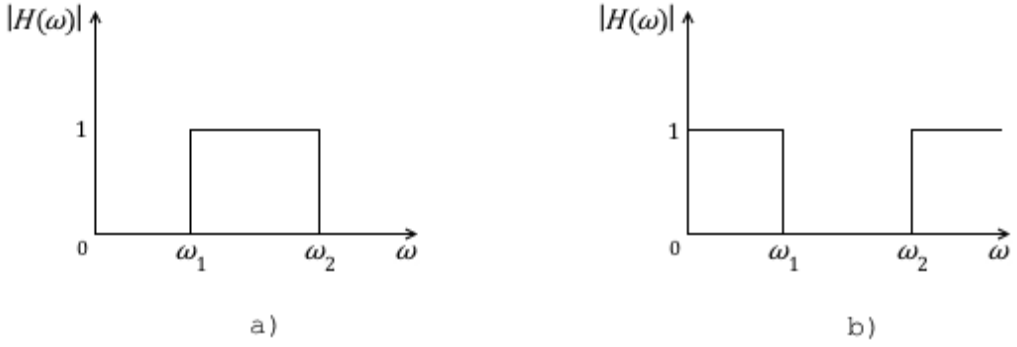

Figura 2.3.2: a) Filtro ideale passa banda, banda passante $[\omega_1,\omega_2]$, banda attenuate $[0,\omega_1]\cup[\omega_2,+\infty]$. b) Filtro ideale elimina banda, banda passante $[0,\omega_1]\cup[\omega_2,+\infty]$ banda attenuata $[\omega_1,\omega_2]$.

I Filtri 1,2,3,4 sono chiamati ideali perché non sono fisicamente realizzabili, infatti, la risposta impulsiva (ricavata, ad esempio, antitrasformando la F.d.t.) sarebbe positiva anche per $t < 0$ (o $n < 0$) e perciò infrangerebbe il vincolo di sistema causale. [B1]
Per tale motivo i filtri analogici reali presentano una parte tra la banda attenuata e quella passante chiamata "raccordo" che può avvicinarsi o meno al filtro, a seconda del grado della F.d.t, meglio noto come ordine del filtro (Fig.2.3.3).

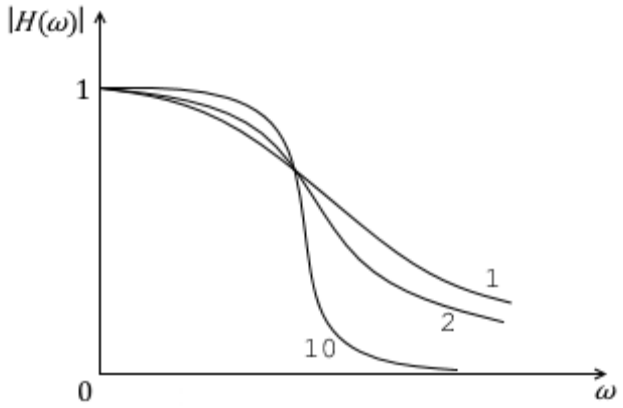

Figura 2.3.3: Tre filtri passa basso con ordini differenti (1,2,10). Si osserva che aumentando l'ordine del filtro la funzione tende a quella ideale in Fig.2.3.1 a.

Un metodo per implementare un filtro tempo-continuo è quello di calcolare la F.d.t partendo da un circuito analogico, ad esempio un filtro passa basso può essere implementato da un circuito RC (resistenza più un condensatore in serie).

2.4 FILTRI FIR E IIR DIGITALI

La base dell'implementazione di filtri digitali ricade su due tipologie che danno la possibilità di creare qualsiasi tipo di filtro: quelli a risposta impulsiva infinita (IIR, infinite impulse response) e quelli a risposta finita (FIR, finite impulse response).

Si definisce filtro ricorrente, un sistema formato da linee di ritardo T in cui vengono prelevate versioni diversamente ritardate dell'ingresso e dell'uscita sommate tra loro, non prima di essere state moltiplicate per una costante appropriata (Fig.2.4.1). [B1][B3]

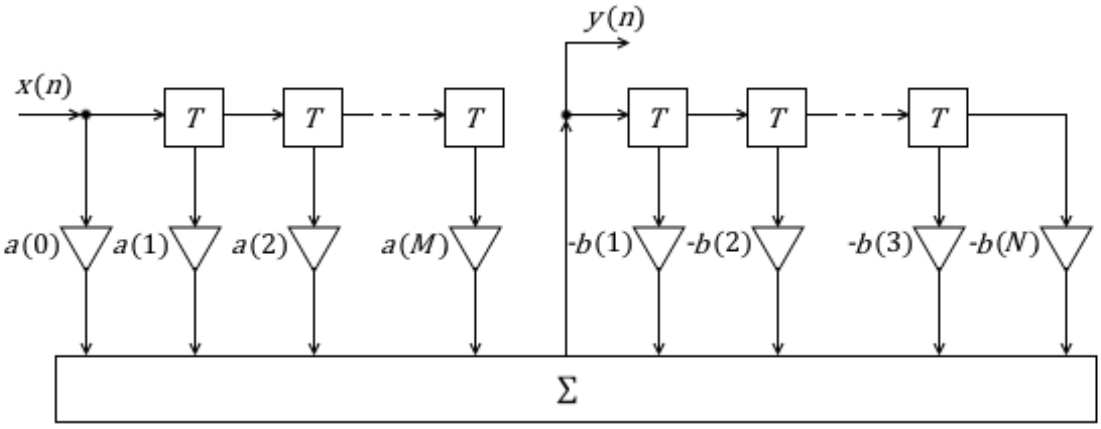

Figura 2.4.1: Schema di un filtro IIR generico

La relazione ingresso-uscita della Fig.2.4.1 può essere calcolata nel dominio delle frequenze (2.4.2) partendo dalla relazione ingresso-uscita nel tempo (2.4.1),

(2.4.1) $$y(n) = \sum_{k=0}^{M} a(k)\, x(n-k) - \sum_{k=1}^{N} b(k)\, y(n-k)$$

$$Z\{y(n)\} = Z\left\{\sum_{k=0}^{M} a(k)\, x(n-k) - \sum_{k=1}^{N} b(k)\, y(n-k)\right\} \Rightarrow$$

$$Y(z) = \sum_{k=0}^{M} a(k)\, X(z) z^{-k} - \sum_{k=1}^{N} b(k)\, Y(z) z^{-k} \Rightarrow$$

(2.4.2) $$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} a(k)\, z^{-k}}{1 + \sum_{k=1}^{N} b(k)\, z^{-k}}$$

con N l'ordine del filtro. [B1][B3][B4]

Questo tipo di sistema LTI ha la particolarità di avere una risposta impulsiva infinita, da cui prende il nome di filtro IIR, per tale regione può soffrire di problemi di stabilità. [B3]
La progettazione di questi filtri non ricade sulla ricerca della risposta impulsiva, bensì su quella dei valori dei coefficienti $a(k)$ e $b(k)$, da cui è possibile ricavare una forma del tipo descritto in 2.4.1. Un metodo è quello di partire dal circuito analogico, corrispondente al filtro desiderato, e ricavarne l'equazione ingresso-uscita nel dominio dei tempi. Ad esempio, un passa basso può essere creato partendo da un circuito RC e ricavando la funzione di trasferimento nel dominio tempo-continuo $H(s)$. Dopodiché $H(s)$ viene trasformata tramite un'opportuna corrispondenza tra il piano s e il piano z. Una relazione che permette ciò (ne esistono diverse) è la trasformazione bilineare (2.4.3). [B3]

(2.4.3) $$s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \Leftrightarrow z = \frac{1+\frac{sT}{2}}{1-\frac{sT}{2}}$$

La (2.4.3) garantisce la corrispondenza fra i due domini, la razionalità della funzioni $H(z), H(s)$ (se quella di partenza era razionale) e la stabilità del sistema (se quello di partenza era stabile), ma altera la risposta in frequenza comprimendola verso l'origine (2.4.4). [B3]

(2.4.4) $$\omega_{analog} = \frac{2}{T}\tan(\frac{T\omega_{digital}}{2})$$

Ulteriori specifiche saranno trattate nel capitolo successivo, quando si esplicheranno i metodi di progetto di un equalizzatore.
Riprendendo lo schema in Fig.2.4.1, è possibile eliminare tutte le linee di ritardo del segnale di uscita ponendo $b(k) = 0$ per ogni k. Il risultato ottenuto è un filtro trasversale (filtro FIR) di ordine M

(Fig.2.4.2) avente la caratteristica di una risposta impulsiva finita, per cui non ha problemi di stabilità. [B3]

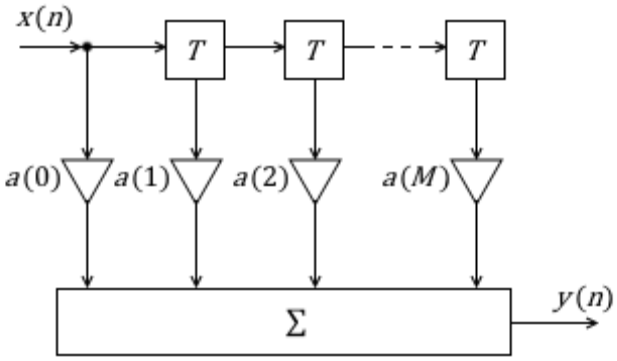

Figura 2.4.2: Schema di un filtro IIR generico

La relazione ingresso-uscita della Fig.2.4.2 (2.4.5) non è altro che il prodotto di convoluzione discreto tra gli ingressi e $a(k)$, per cui, per la (2.2.4) tali coefficienti rappresentano la risposta impulsiva del sistema. [B4]

(2.4.5) $$y(n) = \sum_{k=0}^{M} h(k)\, x(n-k)$$

Con un procedimento analogo a quello della (2.4.2) è possibile ricavare la F.d.t del filtro trasversale (2.4.6).

$$Y(z) = \sum_{k=0}^{M} h(k)\, X(z) z^{-k} \Rightarrow$$

(2.4.6) $$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{M} h(k)\, z^{-k}$$

Tre metodi di progetto dei filtri FIR sono: metodo delle finestre, metodo del campionamento in frequenza e criterio di Chebychev.

Tali tecniche, non essendo state utilizzate, vengono solo passate in rassegna. [B3]

2.5 FILTRI COMB

E' importante considerare una famiglia di filtri molto usata per la creazione di effetti audio: i filtri comb. Questi sistemi hanno la particolarità che la loro F.d.t. in ampiezza ricorda proprio la forma di un pettine (in inglese "comb") e sono implementabili aggiungendo al segnale una sua versione ritardata di un certo numero di passi. Possono essere di tipo FIR o IIR. [B4]
Quelli di tipo FIR sono espressi tramite la relazione ingresso-uscita in 2.5.1,

(2.5.1) $$y(n) = x(n) + g\,x(n-D)$$

dove g è un'opportuna costante che determina il "guadagno" (gain) della ripetizione, e D è il ritardo della ripetizione (delay).

(2.5.2) $$H(z) = 1 + g\,z^{-D}$$

(2.5.3) $$|H(\omega)| = \sqrt{2(1+\cos(\omega D))}$$

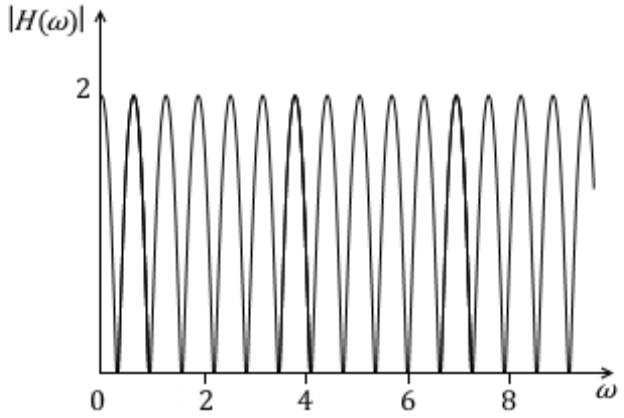

Figura 2.5.2: Spetto della 2.5.3

La 2.5.2 esprime la F.d.t. della 2.5.1. A scopo illustrativo, posto $g = 1$ alla 2.5.1 (per semplificare i calcoli), si ottiene lo spettro di ampiezza continuo 2.5.3 che assume l'andamento in Fig.2.5.2. [B4] I filtri comb di tipo IIR presentano un ritardo D anche dell'uscita, un paio di esempi sono citati in 2.5.4 e in 2.5.5. [B4]

(2.5.4) $$y(n) = x(n - D) - g\, y(n - D)$$

(2.5.5) $$y(n) = x(n) - g\, y(n - D)$$

Un'ultima categoria non utilizzata nel progetto ma che è giusto riportare sono i filtri comb universali o anche detti all-pass, i quali modificano la caratteristica in fase del segnale mantenendo inalterata l'ampiezza. [B4]

CAPITOLO 3
EFFETTI PER STRUMENTI ELETTRICI A CORDA

I metodi esposti nel capitolo 2 introducono le basi per la realizzazione di effetti lineari, una parte dei quali viene utilizzata per creare sistemi che trasformano il suono prodotto da uno strumento elettrico a corda. In questo capitolo sarà disponibile una classificazione di tali effetti seguita dal procedimento di sviluppo di due sistemi: un equalizzatore a 10 bande e un delay, che saranno implementati nello smartphone.

3.1 DEFINIZIONE E CLASSIFICAZIONE

Il termine generico "effetto" indica il cambiamento di percezione risultante da una causa, di conseguenza un "effetto audio" può essere definito come una trasformazione del suono prodotto da una qualsiasi fonte.
Esistono varie tecniche e tipologie di sintesi, come ad esempio i Vocoder, utilizzati per la riproduzione della voce umana, o gli standard MPEG adottati per la compressione dei file multimediali. Questa tesi pone come obbiettivo lo studio della parte degli effetti audio solitamente utilizzati per modificare il segnale in uscita da uno strumento elettrico a corda, come il basso elettrico o la chitarra elettrica. [B6]
Esistono due tipi di classificazioni ufficiali: una basata su come l'effetto viene percepito dall'orecchio umano, l'altra sul metodo utilizzato per produrre l'algoritmo. Partendo dalle classificazioni in [B6], si è scelto di adottare un tipo di classificazione più generico e adatto allo scopo, all'interno del quale si cerca di inglobare tutti gli effetti utilizzati nell'ambito di interesse studiato. Da ciò ne deriva la suddetta distinzione:

1) EFFETTI LINEARI: Equalizzatore, Wha, Phaser, Delay, Franger, Chorus, Riverbero

2) EFFETTI NON LINEARI: Distorsore, Compressore.

E' possibile che si utilizzino effetti derivati da combinazioni di due o più trasformazioni citate oppure in uso in altre discipline.

3.2 PROGETTAZIONE DI UN EQUALIZZATORE A 10 BANDE

Il primo progetto sviluppato è un equalizzatore (EQ) a 10 bande, dove il numero indica le fasce di spettro nelle quali la trasformazione agisce.
L'equalizzatore è un effetto che nel campo audio viene utilizzato per adattare il suono all'ambiente, compensando le carenze di alcuni elementi della catena audio (ad esempio aumentando alcune frequenze attenuate dall'altoparlante), oppure per dare una caratteristica al suono (riducendo le basse frequenze si ottiene un suono più metallico).

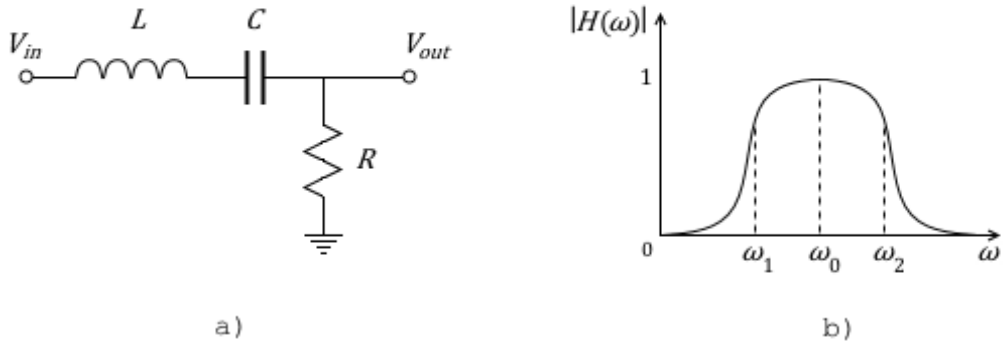

Figura 3.2.1: a) Circuito RLC di un passa banda di secondo ordine. b) Esempio di risposta in frequenza di un filtro passa banda, dove ω_0 è la frequenza di risonanza e ω_1, ω_2 le frequenze di taglio.

La strategia di implementazione dell'algoritmo è quella di inserire dieci filtri IIR passa banda in parallelo con frequenze centrali (dette anche "frequenze di risonanza") differenti. Il filtro viene realizzato partendo dal modello analogico di un passa banda del

secondo ordine (Fig.3.2.1), per poi essere convertito in digitale mediante la trasformazione bilineare, come già discusso nel paragrafo 2.4.

Si osserverà in seguito che l'adozione di questa tecnica comporta l'utilizzo di poche istruzioni, che non incidono significativamente sulle performance del dispositivo.

Esaminando il circuito in Fig.3.2.1 a, è possibile ricavare la relazione ingresso-uscita (3.2.1) e di conseguenza la F.d.t (3.2.2). Ricordando che $s = j\omega$ si ottiene la 3.2.3.

(3.2.1) $$V_{out} = \frac{R}{R + \frac{1}{j\omega C} + j\omega L} V_{in}$$

(3.2.2) $$H(\omega) = \frac{V_{out}}{V_{in}} = \frac{j\omega R}{j\omega R + (j\omega)^2 L + \frac{1}{C}}$$

(3.2.3) $$H(s) = \frac{sR}{sR + s^2 L + \frac{1}{C}}$$

Essendo il sistema di secondo ordine, dalla 2.4.2, si presume che la F.d.t. nel dominio Z sia del tipo descritto in 3.2.4,

(3.2.4) $$H(z) = \frac{a(0) + a(1)z^{-1} + a(2)z^{-2}}{1 + b(1)z^{-1} + b(2)z^{-2}}$$

l'operazione successiva, infatti, sarà quella di trovare i coefficienti applicando la 2.4.3 alla 3.2.3 ottenendo la caratteristica in frequenza nel dominio, valido per segnali tempo-discreti (3.2.5)

$$H(z) = \frac{\frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} R}{\frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} R + \left(\frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}}\right)^2 L + \frac{1}{C}} =$$

$$= \frac{2R(1 - z^{-1})}{2R(1 - z^{-1}) + \frac{4}{T}\frac{(1 - z^{-1})^2}{1 + z^{-1}} L + \frac{T(1 + z^{-1})}{C}} =$$

$$= \frac{2RT(1+z^{-1})(1-z^{-1})}{2RT(1+z^{-1})(1-z^{-1}) + 4L(1-z^{-1})^2 + \frac{\left(T(1+z^{-1})\right)^2}{C}} =$$

$$= \frac{2RT(1-z^{-2})}{2RT(1-z^{-2}) + 4L(1-2z^{-1}+z^{-2}) + \frac{T^2}{C}(1+2z^{-1}+z^{-2})} =$$

$$= \frac{\frac{RT}{2}(1-z^{-2})}{\frac{1}{2}\left(RT+2L+\frac{T^2}{2C}\right) - (2L-\frac{T^2}{2C})\,z^{-1} + (L-\frac{RT}{2}+\frac{T^2}{4C})\,z^{-2}} \Rightarrow$$

(3.2.5)
$$H(z) = \frac{\alpha(1-z^{-2})}{\frac{1}{2} - \gamma z^{-1} + \beta z^{-2}}$$

con i relativi coefficienti (3.2.6). [B2]

(3.2.6)
$$\alpha = \frac{\frac{RT}{2}}{RT+2L+\frac{T^2}{2C}}$$

$$a(1) = 0$$

$$\gamma = \frac{2L-\frac{T^2}{2C}}{RT+2L+\frac{T^2}{2C}}$$

$$\beta = \frac{L-\frac{RT}{2}+\frac{T^2}{4C}}{RT+2L+\frac{T^2}{2C}}$$

E' opportuno notare che la differenza tra il formalismo espresso in 3.2.4 e quello in 3.2.5 non comporta alcun cambiamento ai fini pratici.

Questo tipo di progetto comporta una relazione non lineare tra il dominio analogico e quello digitale (2.4.4) che deve essere presa in considerazione. E' ben noto che per valori molto piccoli ($\vartheta < \pi/4$), una funzione trigonometrica può essere espressa come in 3.2.7, 3.2.8 e 3.2.9. [B2]

(3.2.7) $$\sin(\vartheta) \cong \vartheta$$

(3.2.8) $$\cos(\vartheta) \cong 1 - \frac{\vartheta^2}{2}$$

(3.2.9) $$\tan(\vartheta) \cong \vartheta$$

Questa approssimazione, detta Small-Angle Approximation (SAA), applicata alla 2.4.4 permette di avere una corrispondenza diretta tra i due domini (analogico e digitale). In questa applicazione la maggior parte delle frequenze prese in considerazione rispetta tale principio. E' importante fare attenzione quando la si mette in pratica poiché il filtro IIR in analisi dipende molto da piccole differenze di numeri. [B2][S8]

Ricordando che $z = e^{j\omega T}$ e la 1.1.3 si può riscrivere la 3.2.5 come in 3.2.10.

(3.2.10) $$H\left(e^{j\omega T}\right) = \frac{\alpha(1 - e^{-j2\omega T})}{\frac{1}{2} - \gamma e^{-j\omega T} + \beta e^{-j2\omega T}} = \frac{\alpha(e^{j2\omega T} - 1)}{\frac{1}{2}e^{j2\omega T} - \gamma e^{j\omega T} + \beta} =$$
$$= \frac{\alpha(\cos(2\omega T) - 1) + j\sin(2\omega T)}{\frac{1}{2}\cos(2\omega T) - \gamma\cos(\omega T) + \beta + j(\frac{1}{2}\sin(2\omega T) - \gamma\sin(\omega T))}$$

Dopodiché, dalla 1.1.5, il modulo della F.d.t (chiamato anche "guadagno") si modifica come in 3.2.11.

$$|H(\omega)| = \sqrt{H(e^{j\omega T}) \cdot H^*(e^{j\omega T})} \Rightarrow$$

(3.2.11) $$|H(\omega)| = \frac{2\alpha\sin(\omega T)}{\sqrt{((\frac{1}{2} - \beta)\sin(\omega T))^2 + ((\frac{1}{2} + \beta)(\cos(\omega T) - \cos(\omega_0 T)))^2}}$$

(3.2.12) $$\cos(\omega_0 T) = \frac{\gamma}{\frac{1}{2} + \beta} \Rightarrow \gamma = (\frac{1}{2} + \beta)\cos(\omega_0 T)$$

(3.2.13) $$H(\omega_0) = \frac{2\alpha}{\frac{1}{2} + \beta} = 1 \Rightarrow \alpha = \frac{1 + 2\beta}{4}$$

La 3.2.12 rappresenta il coseno della frequenza centrale. [B2]

Esaminando l'equazione 3.2.11 si possono fare le seguenti osservazioni:

1) Il guadagno è proporzionale ad α.
2) La 3.2.13 esprime il guadagno alla frequenza di risonanza.
3) La larghezza di banda è regolata dal parametro β.
4) E' simmetrica in scala logaritmica. [B2]

La 4 è ancora più visibile applicando la SAA alla 3.2.11 (3.2.14).

$$|H_a(\omega)| = \frac{H(\omega_0)}{\sqrt{1+\left(\frac{\frac{1}{2}+\beta}{\frac{1}{2}-\beta}\right)^2 \cdot \left(\frac{(\omega_0 T)^2-(\omega T)^2}{2\omega T}\right)^2}} \qquad (3.2.14)$$

Sostituendo $\omega = k\omega_0$ oppure $\omega = \omega_0/k$ si ha un'equazione equivalente della 3.2.14, che mette in risalto la simmetria logaritmica. Il pedice "a" indica che è stata applicata la SAA. [B2]
Osservando la Fig.3.2.1 b, è ben noto che $|H_a(\omega_1)| = |H_a(\omega_2)| = H(\omega_0)/\sqrt{2}$, inoltre, grazie alla simmetria logaritmica, si ha $\omega_1 = \omega_0/k$ e $\omega_2 = k\omega_0$. Definito Q (3.2.15) il "fattore di merito" che determina l'ampiezza della banda passante,

$$Q = \frac{\omega_0}{\omega_2 - \omega_1} = \frac{\omega_0}{k\omega_0 - \frac{\omega_0}{k}} = \frac{k}{k^2-1} \qquad k > 1 \qquad (3.2.15)$$

è possibile ricavare β_a (3.2.16) dalla 3.4.14 studiando la F.d.t in una delle due frequenze di taglio.

$$\frac{H(\omega_0)}{\sqrt{1+\left(\frac{\frac{1}{2}+\beta}{\frac{1}{2}-\beta}\right)^2 \cdot \left(\frac{(\omega_0 T)^2-(\omega_1 T)^2}{2\omega_1 T}\right)^2}} = \frac{H(\omega_0)}{\sqrt{2}} \Rightarrow$$

$$\left(\frac{\frac{1}{2}+\beta_a}{\frac{1}{2}-\beta_a}\right)^2 \cdot \left(\frac{(\omega_0 T)^2-(\omega_1 T)^2}{2\omega_1 T}\right)^2 = 1 \Rightarrow$$

$$\frac{\frac{1}{2}+\beta_a}{\frac{1}{2}-\beta_a} = \frac{2}{\omega_0 T}\left(\frac{k}{k^2-1}\right) = \frac{2Q}{\omega_0 T} \Rightarrow$$

(3.2.16)
$$\beta_a = \frac{Q-\frac{\omega_0 T}{2}}{2Q+\omega_0 T}$$

Grazie alle equazioni 3.2.12, 3.2.13 e 3.2.16 si possono ricavare, conoscendo la frequenza di risonanza e il fattore di merito, i coefficienti del filtro IIR per $f < f_s/8$, con f_s la frequenza di campionamento. In questo esperimento $f_s = 44100$ Hz, quindi è possibile coprire lo spettro fino a 5512.5 Hz. Nella Tab.3.2.1 vengono espressi la suddivisione e il calcolo dei coefficienti, imponendo arbitrariamente $Q = 1.4$. Le ultime due righe della tabella, non soddisfacenti la SAA, vengono estrapolate da [B2] in cui viene citato anche il metodo di calcolo.
Una volta resi noti i coefficienti, è possibile ricavare l'equazione differenziale del filtro in analisi (3.2.17) antitrasformando la 3.2.5.

$$Z^{-1}\{H(z)\} = Z^{-1}\left\{\frac{Y(z)}{X(z)}\right\} \Rightarrow$$

$$Z^{-1}\{X(z)\alpha(1-z^{-2})\} = Z^{-1}\left\{Y(z)\left(\frac{1}{2}-\gamma z^{-1}+\beta z^{-2}\right)\right\} \Rightarrow$$

(3.2.17)
$$y(n) = 2(\alpha x(n) - \alpha x(n-2) + \gamma y(n-1) - \beta y(n-2))$$

Come in Fig.3.2.2, si inseriscono dieci filtri IIR passa banda (PB) in parallelo, ognuno moltiplicato per un coefficiente $-0.2 < g < 1$ che permette un guadagno di ± 14 dB applicato nello spazio circostante la frequenza centrale. I vari sistemi vengono successivamente sommati tra loro formando l'uscita equalizzata $y(n)$. [B2]

Frequenza centrale (Hz)	α	β	γ
31	0.000787462865	0.498425074	0.998415336
62	0.00157244917	0.496855102	0.996816209
125	0.00316016172	0.493679677	0.993522095
250	0.00628062774	0.487438745	0.986812425
500	0.0124054279	0.475189144	0.972715729
1000	0.0242101804	0.451579639	0.941937749
2000	0.0461841095	0.407631781	0.871031797
4000	0.0845577687	0.330884463	0.699565951
8000	0.1199464	0.2601072	0.3176087
16000	0.159603	0.1800994	−0.4435172

Tabella 3.2.1: Tabella dei coefficienti del filtro passa banda rappresentato nel dominio tempo-discreto dall'equazione differenziale 3.2.17.

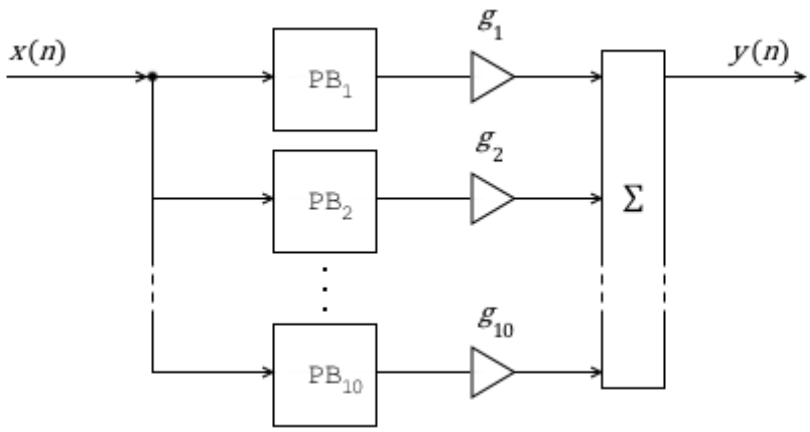

Figura 3.2.2: Schema dell'equalizzatore a 10 bande.

3.3 PROGETTAZIONE DELL'EFFETTO DELAY

Il secondo effetto implementato, molto utilizzato nel campo musicale, è il delay: trattasi di una trasformazione audio che registra il suono

in ingresso riproducendolo con un determinato ritardo temporale. In pratica vengono prese più versioni del segnale in ingresso, le quali vengono reintrodotte in uscita in intervalli di tempo differenti.[S5] La sua realizzazione è possibile mediante un filtro comb FIR (2.5.1) per ogni ritardo concepito: in questo caso saranno disponibili tre ritardi di 10000, 20000 e 30000 campioni regolabili proporzionalmente dall'utilizzatore tramite una costante $10 \leq k \leq 30$ (3.3.1).

$$D_1 = \frac{1}{3}k \cdot 1000 \Rightarrow 3334 \leq D_1 \leq 10000$$
$$D_2 = \frac{2}{3}k \cdot 1000 \Rightarrow 6667 \leq D_2 \leq 20000 \quad (3.1.1)$$
$$D_3 = k \cdot 1000 \Rightarrow 10000 \leq D_3 \leq 30000$$

Rispettivamente, l'ampiezza dei tre segnali è diminuita da fattori arbitrari g (3.1.2) per dare eco al suono.

$$g_1 = 0.6;$$
$$g_2 = 0.3; \quad (3.1.2)$$
$$g_3 = 0.1;$$

Richiedendo di salvare una stringa di dati, il delay impone la progettazione di un buffer circolare di grandezza pari al ritardo massimo del filtro. L'algoritmo di una singola linea di ritardo è espresso nel diagramma di flusso in Fig.3.3.1. Tale procedimento dovrà essere ripetuto per tutti i ritardi desiderati.
Anche se l'impressione dell'utilizzatore è quella di modificare il delay in tempo reale, come è osservabile in Fig.3.3.1, il ridimensionamento del buffer è posto a fine giro, questo per evitare che il suono si spezzi in tronco rendendo l'ascolto spiacevole. Un altro accorgimento che viene adottato successivamente alla riorganizzazione del buffer è quello di annullare tutti i campioni dopo l'ultimo utilizzato, evitando, in caso si aumenti D, l'introduzione di suoni sporchi rimasti in memoria.

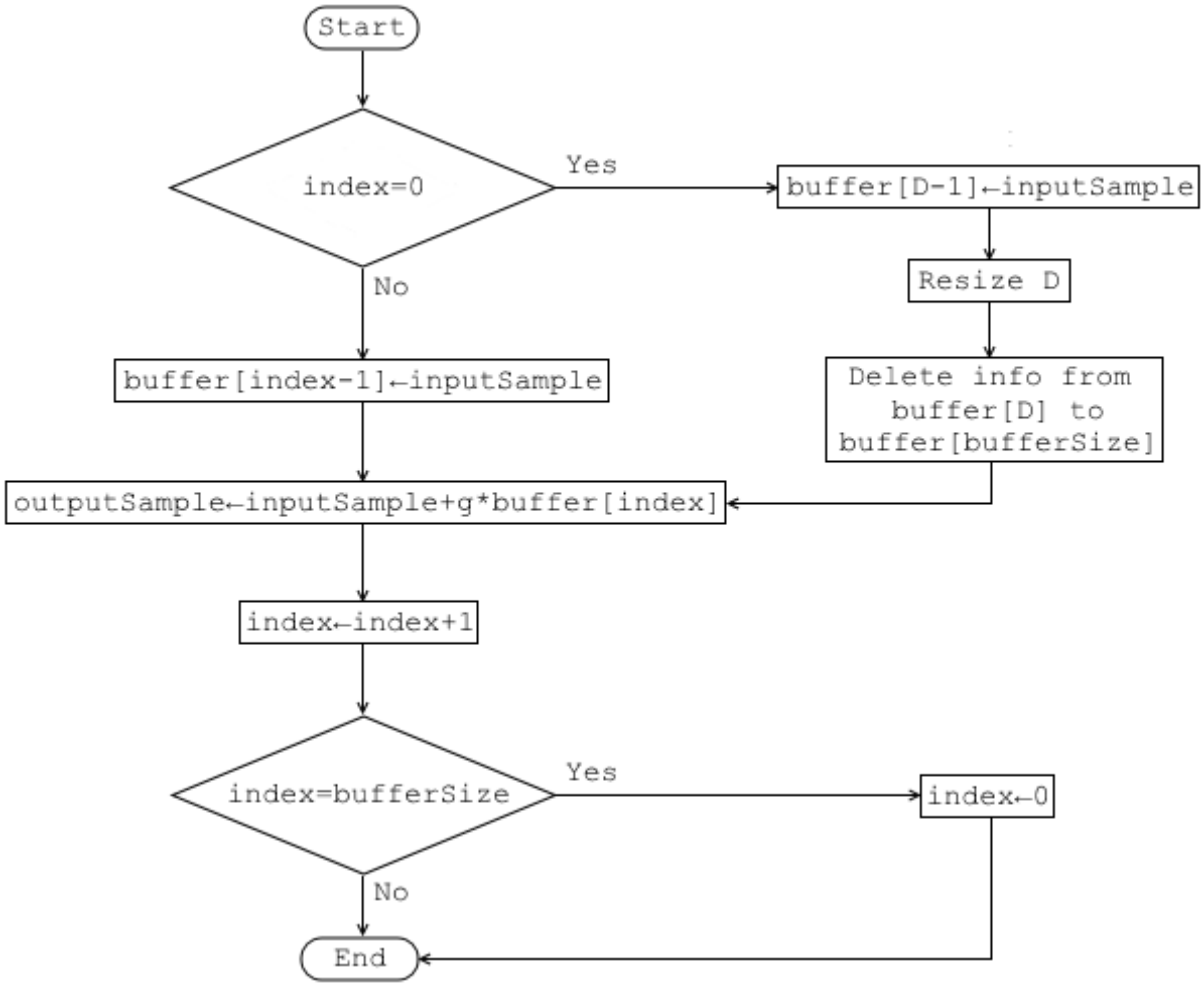

Figura 3.3.1: Diagramma di flusso di una linea di ritardo del delay

CAPITOLO 4
PROGETTAZIONE PER SMARTPHONE

In questo capitolo viene messo in pratica ciò che è stato esposto nei capitoli precedenti, partendo da un'introduzione generale si arriverà a descrivere, più dettagliatamente, le classi utilizzate. Non sarà trascritto in toto il codice dell'applicazione ma solo le parti più significative, accompagnate da commenti che ne introducono la funzione. Alcune considerazioni descritte vengono estrapolate empiricamente dal progetto stesso.

4.1 INTRODUZIONE ALLA PROGETTAZIONE

Il motivo per il quale è stata scelta l'implementazione sullo smartphone è quello di rendere portatile una tecnologia che fino a pochi anni fa non lo era. Normalmente, gli effetti audio per strumenti vengono costruiti tramite apposite pedaliere che comportano un costo elevato, oltre che un dispendio di spazio. Grazie all'implementazione eseguita è possibile inglobare tutti gli effetti in un solo dispositivo, che è il medesimo utilizzato quotidianamente dalla maggioranza delle persone. I vantaggi principali sono i costi ridotti (un terzo del corrispondente fisico) e la versatilità. Ovviamente servirsi di uno strumento non appositamente creato per quello scopo porta anche a degli svantaggi, ad esempio la potenza in uscita dal dispositivo non permette l'utilizzo in concerti, e la latenza tra ingresso e uscita, per quanto accettabile, è sempre maggiore a quella di un pedale. Ad oggi, tutto ciò rende auspicabile avvalersi di questa applicazione per le prove casalinghe, in studio di registrazione, o per provare un componente prima di comprarlo.
La prima problematica da affrontare è quella di capire in che modo è possibile interfacciare il dispositivo con uno strumento elettrico a corda. Uno smartphone ha generalmente due canali ingresso-uscita: il connettore jack femmina da 3.5 mm (Mini-jack) e la porta USB (o Lightning se si parla di iPhone 5s). Interfacciare la chitarra tramite

porta USB impedirebbe all'utilizzatore di mettere in carica il dispositivo, per questo motivo si è preferito optare per l'altro canale, la porta Mini-jack. Gli strumenti come la chitarra elettrica, aventi un jack output da 6.3 mm maschio, necessitano di un adattatore che non solo permette lo scorrere dell'informazione dallo strumento verso lo smarphone, ma anche l'inserimento di un dispositivo, come le cuffie, per sentire l'audio prodotto in tempo reale. La soluzione è stata trovata utilizzando l' iRig (IK Multimedia): questo dispositivo soddisfa i requisiti necessari a svolgere tutte le funzioni cercate grazie ad un cavo Mini-jack maschio, un ingresso da 6.3 mm per lo strumento ed uno da 3.5 per le cuffie (Fig.4.1.1).

Figura 4.1.1: iRig. (Copyright immagine: IKMultimedia Production srl)

Una volta identificato l'adattatore da utilizzare, bisogna scoprire quali sono i sistemi operativi (OS, Operative System) per i dispositivi mobili che possono supportare questa tecnologia. Sono stati presi in considerazione i due principali OS: Android (versione 4.4) e iOS (versione 7.1). Dalle ricerche svolte però è emerso che solo iOS è in grado di implementare la sintesi audio in tempo reale nelle circostanze studiate. Le operazioni impiegate nel tentativo di creare librerie apposite per Android non hanno prodotto risultati soddisfacenti nei test di latenza. Per queste ragioni si è scelto di

lavorare sul sistema operativo della Apple e utilizzare XCode come IDE (Integrated Development environment). [S1][S4][S11]
L'applicazione "Effects", sviluppata per l'occasione, consente all'utente di collegare il proprio strumento all'iPhone (4s o superiore), permettendo di sentire il suono prodotto e modificarne l'andamento tramite un equalizzatore a 10 bande e un effetto delay. Nell'applicativo, che dovrà opportunamente funzionare in tempo reale, sarà visibile anche lo spettro in frequenza in decibel, di cui si parlerà nel paragrafo 4.3. L'interfaccia grafica è visibile in Fig.4.1.2 e sarà descritta brevemente nel paragrafo 4.5.

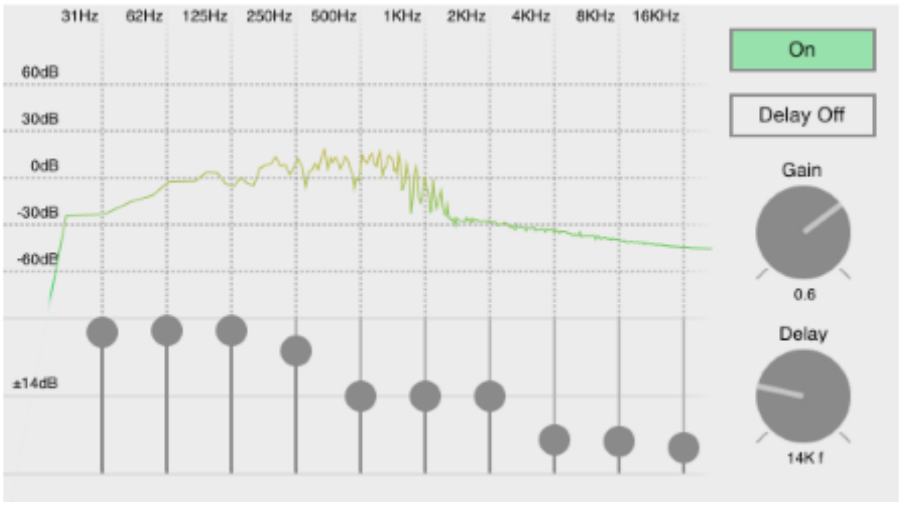

Figura 4.1.2: Interfaccia grafica dell'applicazione "Effects" sviluppata in questa tesi.

Lo sviluppo è stato concepito tramite la creazione di varie classi, scritte in linguaggio Objective-C (che permette di inserire codice anche di linguaggio C), le quali vengono passate in rassegna:

1) **MyAudioUnit:** gestisce il flusso audio in entrata e in uscita dal dispositivo.
2) **FastFourierTransform:** permette la trasformazione di un vettore dal dominio del tempo al dominio delle frequenze (in decibel).
3) **Equalizer10Band:** implementa l'equalizzatore a 10 bande.
4) **DelayEffect:** implementa l'effetto delay.

5) **ViewController:** e' responsabile dell'interfaccia grafica e dell'interazione tra le varie classi.

Per realizzarle sono stati utilizzati i seguenti framework: AVFondation, Accelerate, AudioToolbox, CoreAudio, Foundation, CoreGraphics e UIKit.

4.2 SESSIONE E UNITA' AUDIO

Il contesto audio viene impostato tramite la classe **AVAudioSession**. Questa classe viene utilizzata per: attivare o disattivare una sessione audio, sceglierne il metodo di utilizzo, configurare le impostazioni audio e stabilire quali funzioni devono essere utilizzate a seconda degli eventi audio che si verificano. [S3]

```
// Inizializza l'unità audio
audioUnit = (AudioUnit*)malloc(sizeof(AudioUnit));

// Attiva la sessione audio
if (![[AVAudioSession sharedInstance] setActive:YES error:&error])
{ return 1; }

// Stabilisce la categoria del flusso audio
if (![[AVAudioSession sharedInstance]
setCategory:AVAudioSessionCategoryPlayAndRecord error:&error])
{ return 1; }

// Viene inserita la grandezza preferibile del buffer (in secondi). Non è sicuro
che il sistema la rispetti sempre.
Float32 bufferSizeInSec = 0.02f;
if (![[AVAudioSession sharedInstance]
setPreferredIOBufferDuration:bufferSizeInSec error:&error])
{ return 1; }

// Cambia temporaneamente il percorso audio corrente. Se utilizzato con l'opzione
AVAudioSessionPortOverrideSpeaker e la categoria della sessione audio
AVAudioSessionCategoryPlayAndRecord impone l'utilizzo dell'altoparlante e del
microfono indipendentemente dalle altre impostazioni.
if (![[AVAudioSession sharedInstance]
overrideOutputAudioPort:AVAudioSessionPortOverrideSpeaker error:&error])
{ return 1; }
```

Codice 4.2.1: Parte della funzione **initAudioSession** situata nella classe **MyAudioUnit**

L'oggetto creato agisce da intermediario tra l'applicazione e i servizi multimediali del sistema, che a loro volta dialogano direttamente con l'hardware audio sottostante, garantendo ottime prestazioni. AVAudioSession, però, non lascia allo sviluppatore il controllo diretto delle impostazioni: chiede al sistema determinati requisiti che possono essere offerti o meno, ed esempio in caso di conflitti con altre applicazioni. [S3]
Nella classe MyAudioUnit viene implementata la funzione initAudioSession, che inizializza tutta la sessione audio e un oggetto dell' AudioUnit, una classe che definisce l'unità audio utilizzata nel flusso ingresso-uscita (Cod.4.2.1). [S2]
Nella funzione initAudioStreams, invece, vengono definiti tutti i parametri del flusso, associati all' AudioUnit, come la frequenza di campionamento a 44.1 KHz (scelta arbitrariamente), il formato del buffer (PCM, vedi paragrafo 1.4) e il tipo di canale (mono).

```
// Struttura che identifica un unità audio in run-time
AudioComponentDescription componentDescription;
componentDescription.componentType = kAudioUnitType_Output;
componentDescription.componentSubType = kAudioUnitSubType_RemoteIO;
componentDescription.componentManufacturer = kAudioUnitManufacturer_Apple;
componentDescription.componentFlags = 0;
componentDescription.componentFlagsMask = 0;

// Definisce il flusso audio con determinate caratteristiche
AudioStreamBasicDescription streamDescription;
streamDescription.mSampleRate = 44100;
streamDescription.mFormatID = kAudioFormatLinearPCM;
streamDescription.mFormatFlags = kAudioFormatFlagIsSignedInteger |
kAudioFormatFlagsNativeEndian |
kAudioFormatFlagIsPacked;
streamDescription.mBitsPerChannel = 16;
streamDescription.mBytesPerFrame = 2;
streamDescription.mChannelsPerFrame = 1;
streamDescription.mBytesPerPacket = streamDescription.mBytesPerFrame
*streamDescription.mChannelsPerFrame;
streamDescription.mFramesPerPacket = 1;
streamDescription.mReserved = 0
```

Codice 4.2.2: Parte della funzione initAudioStreams situata nella classe MyAudioUnit

La parte principale della classe in analisi è la funzione renderCallback. L'informazione viene convertita da analogico in digitale, per poi

essere immagazzinata in una coda di buffer. La funzione in questione ha la possibilità di leggere ogni singolo buffer (suddiviso in frame), per un'unità di tempo limitata. Questo passaggio consente la modifica del contenuto dell'informazione, perciò le operazioni che vengono introdotte nel renderCallback devono essere relativamente veloci, ad esempio non è consigliabile l'allocazione di molte risorse o dialoghi con metodi che richiedono tempi di risposta lunghi. Una volta scaduto il timer, il buffer verrà spedito nella coda di uscita, pronto ad essere ascoltato dall'utente. Viene fatto notare che nel Cod.4.2.3, i buffer vengono convertiti da PCM a floating point per permettere l'elaborazione numerica. [S2]
Nella classe MyAudioUnit si adotta una metodologia che consente la delega della funzione, al fine di convergere alla classe ViewController tutte le operazioni che richiedono tali dati.

```
OSStatus renderCallback(void *userData, AudioUnitRenderActionFlags *actionFlags,
const AudioTimeStamp *audioTimeStamp, UInt32 busNumber, UInt32 numFrames,
AudioBufferList *buffers)
{
OSStatus status = AudioUnitRender(*audioUnit, actionFlags, audioTimeStamp, 1,
numFrames, buffers);
if(status != noErr) { return status; }
// Inizializza il buffer

if(convertedSampleBuffer == NULL) {
convertedSampleBuffer = (float*)malloc(sizeof(float)*numFrames) }
SInt16 *inputFrames = (SInt16*)(buffers->mBuffers->mData);

// Conversione da PCM in floating point
for(int i = 0; i < numFrames; i++) {
convertedSampleBuffer[i] = (float)inputFrames[i] / 32768; }

// Delegate per il View Controller
convertedSampleBuffer=delegate(convertedSampleBuffer, numFrames);

// Conversione da floating point in PCM
for(int i = 0; i < numFrames; i++) {
inputFrames[i] = (SInt16)(convertedSampleBuffer[i] * 32767); }
return noErr;
}
```

Codice 4.2.3: Funzione renderCallBack situata nella classe MyAudioUnit.

4.3 IMPLEMENTAZIONE DELLO SPETTRO IN AMPIEZZA

L'esecuzione dell'algoritmo, presentato in 1.5, è disponibile tramite delle funzioni situate nel framework "Accelerate". Quest'ultime impongono che il vettore da processare sia di una grandezza fissa e pari ad una potenza di 2, per ottimizzare le performance.
La funzione `FFT`, che implementa l'intero algoritmo, prende in input un vettore di 4096 valori e ne restituisce un'altro di medesima grandezza, contente lo spettro in ampiezza in dB (4.3.1) del segnale in ingresso.

(4.3.1) $$|X(\omega)|_{dB} = 20\log_{10}|X(\omega)|$$

Come si evince dal paragrafo 4.2, il flusso audio è suddiviso in vari buffer di lunghezza variabile (in media 512), per cui nell'implementazione dovrà essere creato un vettore circolare di 4096 elementi, così da essere sicuri di avere abbastanza frames ed evitare la perdita di informazione.

```
// Converte un vettore formato da numeri complessi in un due vettori: uno con i
valori d'ampiezza e l'altro con i valori in fase
vDSP_ctoz((COMPLEX *) in_real, 2, &split_data, 1, fftSizeOver2);

// Calcola la trasformata di Fourier
vDSP_fft_zrip(fftSetup, &split_data, 1, log2n, FFT_FORWARD);

// Conversione in Decibel
float zero = 1.0;
vDSP_vdbcon(magnitude, 1, &zero, magnitude, 1, fftSizeOver2, 1);
```

Codice 4.3.1: Parte della funzione `FFT` situata nella classe `FastFourierTransform`.

4.4 IMPLEMETAZIONE DEGLI EFFETTI

L'implementazione dell'equalizzatore a 10 bande, studiato nel paragrafo 3.2, è visibile in Cod.4.4.1.

```
void EQ(float *buffer, int index)
{
    sum = 0;
    x[0] = buffer[index];

    for (int j=0; j<10; j++)
    {
        y[0][j] = 2*(c[0][j]*(x[0] - x[2]) + c[2][j]*y[1][j] - c[1][j]*y[2][j]);

        // Crea il nuovo elemento del buffer
        sum= sum + (y[0][j] * getGain(j));

        // Aggiorna le variabili di uscita
        y[2][j]=y[1][j];
        y[1][j]=y[0][j];
    }

    // Buffer di uscita modificato
    buffer[index]=sum;

    // Aggiorna le variabili di ingresso
    x[2]=x[1];
    x[1]=x[0];
}
```

Codice 4.4.1: Funzione **EQ** situata nella classe **Equalizer10Band**.

Tramite un oscilloscopio ne è stata provata l'effettiva funzionalità: i test (risultati positivi) consistevano nell'inserire un segnale a frequenza fissa, corrispondente a quella di risonanza di un filtro passa banda, e nel verificare se, cambiando il guadagno g, si avesse una variazione di ± 14 dB. La Fig.4.4.1 e la Fig.4.4.2 rappresentano un esempio di test svolto alla frequenza centrale di 250 Hz.

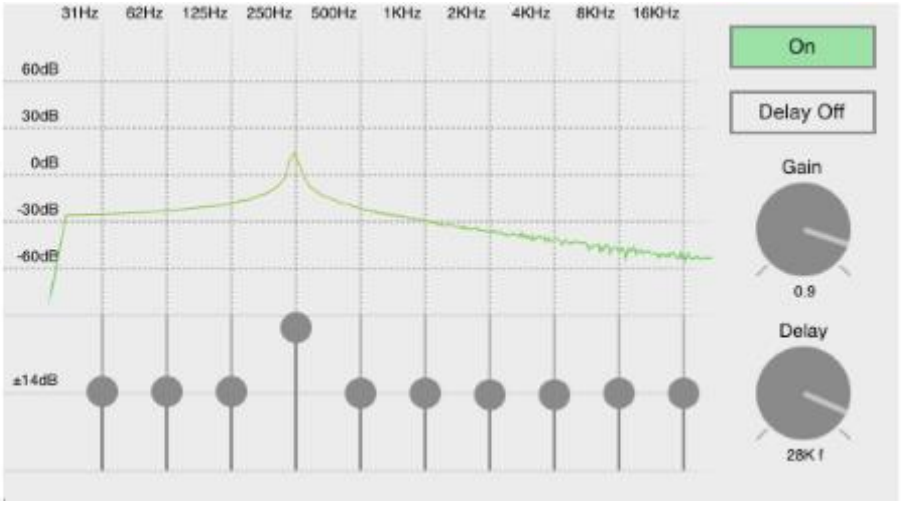

Figura 4.4.1: Spettro del segnale di 250 Hz prodotto da un oscilloscopio. Posto $g = 1$, il guadano sarà di $+14$ dB dall'andamento originale.

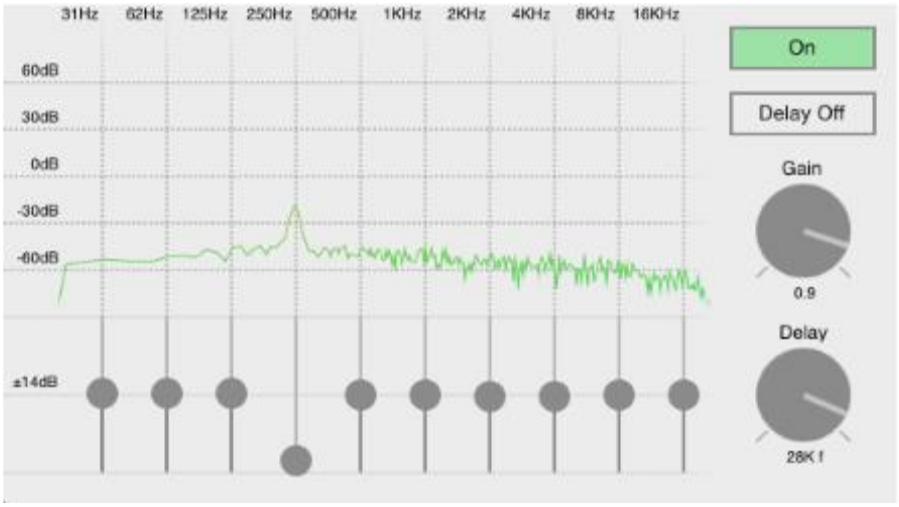

Figura 4.4.2: Spettro del segnale di 250 Hz prodotto da un oscilloscopio. Posto $g = -0.2$, il guadano sarà di -14 dB dall'andamento originale.

L'effetto delay viene implementato direttamente dall'algoritmo in Fig.3.3.1 in Cod.4.4.2.

```
void delayEffect(float *buffer, int index)
{
    if (i1==0)
    {
        x1[bufferSize1-1] = buffer[index];
        bufferSize1 = (int) (size/3)*1000;
        for (int i=bufferSize1; i<10000; i++) { x1[i]=0; }
    }
    else { x1[i1-1] = buffer[index]; }

    if (i2==0)
    {
        x2[bufferSize2-1] = buffer[index];
        bufferSize2 = (int) 2*(size/3)*1000;
        for (int i=bufferSize2; i<20000; i++) { x2[i]=0; }
    }
    else { x2[i2-1] = buffer[index]; }

    if (i3==0)
    {
        x3[bufferSize3-1] = buffer[index];
        bufferSize3 = (int) size*1000;
        for (int i=bufferSize3; i<30000; i++) { x3[i]=0; }
    }
    else { x3[i3-1] = buffer[index]; }

    // Buffer di uscita modificato
    buffer[index]=buffer[index]+gain1*x1[i1]+gain2*x2[i2]+gain3*x3[i3];

    i1++; i2++; i3++;

    if (i1==bufferSize1) { i1=0; }
    if (i2==bufferSize2) { i2=0; }
    if (i3==bufferSize3) { i3=0; }
}
```

Codice 4.4.2: Funzione **delayEffect** situata nella classe **DelayEffect**.

4.5 INTERFACCIA GRAFICA E VIEW CONTROLLER

La classe **ViewController** implementa sia il **callbackDelegate** sia l'interfaccia grafica.

La funzione **callbackDelegate** (Cod.4.5.1) della classe **MyAudioUnit** permette la condivisione dei dati del flusso audio. Tale delega consente di disegnare lo spettro in ampiezza e, all'utente, di controllare le impostazioni degli effetti. Un aspetto rilevante del Cod.4.5.1 riguarda la funzione che calcola la trasformata di Fourier, che viene

posta in background per non incidere sulle prestazioni della sessione audio.

```
-(float*)callbackDelegate:(float*)buffer numberFrames:(int)numFrames
{
    if (buffer!=NULL)
    {
        for(int i=0; i<numFrames; i++)
        {
            EQ(buffer, i);
            buffer[i]=buffer[i]*self.gainKnob.value;

            if (delayOnOff) { delayEffect(buffer, i); }

            //Buffer circolare per FFT
            circularBuffer[indexCircularBuffer]=buffer[i];
            indexCircularBuffer++;
            if(indexCircularBuffer==4096) { indexCircularBuffer=0; }
        }

        // La FTT viene calcolata in background
        dispatch_async(dispatch_get_global_queue(
                       DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
            FFT(circularBuffer, bufferMagnitude, bufferPhase);
        });
    }
    return buffer;
}
```

Codice 4.5.1: Metodo `callbackDelegate:numberFrames:` situato nella classe `ViewController`.

L'interfaccia grafica viene realizzata interamente tramite le classi `BezierPath`, `CAShapeLayer` e `CAGradientLayer`. Per creare un oggetto `UIImage` si utilizza il metodo descritto in Cod.4.5.2, il quale converte un contesto grafico in un'immagine.
Per conferire maggiore fluidità al movimento, lo spettro viene animato tramite la classe `CABasicAnimation`. Il calcolo dello spettro e l'aggiornamento dell'animazione dovranno necessariamente coincidere temporalmente per evitare che altre modifiche dell'interfaccia grafica, dovute ad esempio al tocco di uno slider, ne appesantiscano l'aggiornamento, rendendo lo spettro meno fluido (Cod.4.5.3 e Cod.4.5.4).
Le manopole che cambiano le impostazioni del volume e del delay vengono realizzate dalla classe `MHRotaryKnob`.

```
- (UIImage*) createImageFromBezierPath:(UIBezierPath*)bezierPath withColor:
(UIColor*)color size:(CGSize)size fill:(bool)fill
{
// Crea un contesto grafico bitmap-based
UIGraphicsBeginImageContext(size);

if (fill)
{
    // Crea un BezierPath fill (pieno)
    [color setFill];
    [bezierPath fill];
}

else
{
    // Crea un BezierPath  stroke (solo bordi)
    [color setStroke];
    [bezierPath stroke];
}

// Inserisce il contesto grafico creato dentro un oggetto UIImage
UIImage* image = UIGraphicsGetImageFromCurrentImageContext();

// Chiude il contesto grafico (il primo della pila)
UIGraphicsEndImageContext();
return image;
}
```

Codice 4.5.2: Metodo `createImageFromBezierPath:withColor:size:fill:` situato nella classe `ViewController`.

```
spectrumAnimation = [CABasicAnimation animationWithKeyPath:@"path"];
spectrumAnimation.duration = 0.1;
```

Codice 4.5.3: Parte del metodo `initSpectrumLayer` situato nella classe `ViewController`.

```
drawSpectrumTime = [NSTimer scheduledTimerWithTimeInterval:0.1 target:self
selector:@selector(drawSpectrum) userInfo:nil repeats:YES];
```

Codice 4.5.4: Parte del metodo `startStop` situato nella classe `ViewController`.

BIBLIOGRAFIA

[B1] Calandrino L, Immovilli G, *Schemi delle lezioni di comunicazioni elettriche*, Bologna, Pitagora Editrice, 1991.

[B2] Motorola, *Digital Stereo 10-Band Graphic Equalizer Using the DSP56001*, Phoenix (Arizona), Motorola Literature Distribution, 1988.

[B3] Oppenheim A V, Schafer R W, *Elaborazione numerica dei segnali*, Milano, FrancoAngeli, 1994.

[B4] Uncini A, *Audio Digitale*, Milano, McGraw-Hill, 2006.

[B5] Verrazzani L, *Teoria dei segnali determinati*, Pisa, ETS Università, 1982.

[B6] Zolzer U, *DAFX: Digital Audio Effects* (seconda edizione), Chichester (UK), WYLEY, 2011.

SITOGRAFIA

[S1] Autori di Android Annoiances, *Support for real-time low latency audio; synchronous play and record [UPDATED]*, Android Annoiances, ultima consultazione in data 20/09/2014. http://www.androidannoyances.com/post/38

[S2] Autori di Apple Inc, *Audio Unit Hosting Guide for iOS*, Apple Inc, ultima consultazione in data 19/09/2014. https://developer.apple.com/Library/ios/documentation/MusicAudio/Conceptual/AudioUnitHostingGuide_iOS/Introduction/Introduction.html#//apple_ref/doc/uid/TP40009492-CH1-SW1

[S3] Autori di Apple Inc, *Audio Session Service Reference*, Apple Inc, ultima consultazione in data 19/09/2014.

https://developer.apple.com/Library/ios/documentation/MusicAudio/Conceptual/AudioUnitHostingGuide_iOS/Introduction/Introduction.html#//apple_ref/doc/uid/TP40009492-CH1-SW1

[S4] Autori di Heat Synthesizer, *High Performance/Low Latency Audio on Android…why it still doesn't work*, Heat Synthesizer, ultima consultazione in data 20/09/2014.
http://heatvst.com/wp/2013/11/30/high-performance-low-latency-audio-on-android-why-it-still-doesnt-work/

[S5] Autori di Wikipedia, *Delay*, Wikipedia, L'enciclopedia libera, ultima consultazione in data 06/08/2014, http://it.wikipedia.org/wiki/Delay

[S6] Autori di Wikipedia, *Numero Complesso*, Wikipedia, L'enciclopedia libera, ultima consultazione in data 31/07/2014.
http://it.wikipedia.org/wiki/Numero_complesso

[S7] Autori di Wikipedia, *Segnale Audio*, Wikipedia, L'enciclopedia libera, ultima consultazione in data 31/07/2014.
http://it.wikipedia.org/wiki/Segnale_audio

[S8] Autori di Wikipedia, *Small-angle approximation*, Wikipedia, L'enciclopedia libera, ultima consultazione in data 05/08/2014.
http://en.wikipedia.org/wiki/Small-angle_approximation

[S9] Autori di Wikipedia, *Trasformata di Laplace*, Wikipedia, L'enciclopedia libera, ultima consultazione in data 31/07/2014.
http://it.wikipedia.org/wiki/Trasformata_di_Laplace

[S10] Autori di Wikipedia, *Trasformata zeta*, Wikipedia, L'enciclopedia libera, ultima consultazione in data 31/07/2014.
http://it.wikipedia.org/wiki/Trasformata_zeta

[S11] Gabor Szanto, *Music Creation and Low Latency Audio on Android and iOS*, Superpowered, ultima consultazione in data 20/09/2014. http://superpowered.com/low-latency-audio-android-ios/