GENERAZIONE DI EFFETTI SPECIALI SUL SEGNALE AUDIO

INTRODUZIONE

Lo scopo di questo progetto è la realizzazione di un software applicativo che elabori segnali audio digitali (mono e stereofonici) aggiungendovi effetti speciali, quali echi, cori, riverberi,…

Tali effetti sono stati concepiti alla luce di risultati teorici consolidati presenti in letteratura ([1]).

Come formato audio digitale è stato scelto il Wave (*.wav), sottoinsieme delle specifiche RIFF di Microsoft [6] , per la sua enorme diffusione. Per quanto, invece, concerne l'ambiente software di sviluppo si è optato per Matlab 5.3 rilasciato da Mathworks perchè rappresenta un potente strumento di calcolo ed offre, inoltre, una serie di tools utili all'elaborazione numerica dei segnali (densita' spettrale di potenza, spettrogramma,… [7]). Il tutto e' stato disegnato tramite un'interfaccia GUI (Graphic User Interface [8]) permettendo cosi' all'utilizzatore uno studio degli effetti speciali piu' flessibile ma, allo stesso tempo, rigoroso. I filtri che sono stati implementati sono i seguenti:

- **Eco Singolo**
- **Eco Multiplo (o Riverbero Piano)**
- **Coro**
- **Flanging**
- **Phasing**
- **Riverbero Tipo I: Piano**
- **Riverbero Tipo II: Schroeder (o Passatutto)**
- **Riverbero Tipo III: Passabasso**
- **Compressione/Estensione della Dinamica**

1. IL FORMATO WAVE

Il formato di file denominato WAVE (*.wav) rientra, come sottoclasse, nelle specifiche RIFF rilasciate da Microsoft. Quest'ultime comprendono una enorme varieta' di tipologie di dati differenti, ma tutte caratterizzate dal fatto di essere correlate alla multimedialità (audio, video,…). La trattazione seguente non pretende di essere esaustiva ma, piu' semplicemente, di fornire gli strumenti basilari per capire cosa e' un file audio.

1.1 IL SUONO E L'AUDIO DIGITALE

Chiunque abbia qualche nozione di fisica sa che il suono è un'onda acustica, cioè una variazione di pressione che si propaga attraverso un mezzo, sia esso gassoso (come l'aria), liquido o solido. Ogni qualvolta un'onda di questo tipo raggiunge l'orecchio umano, essa produce delle vibrazioni nella sua membrana, creando quella sensazione nota come suono. La risposta del sitema nervoso umano (alle frequenze comprese tra i 16 Hz e i 20 KHz) costituisce il processo noto come udito.

Viene quindi da chiedersi che relazione intercorra tra un'onda elastica quale quella acustica ed un segnale numerico o digitale. Il suono digitalizzato, attraverso una sequenza di numeri, rappresenta le variazioni nel tempo della pressione aerea. Quando registriamo un suono attraverso il microfono di un calcolatore, l'onda di pressione esercitata sulla sua membrana viene convertita in variazioni di tensione elettrica. Una scheda elettronica dedicata all'audio provvede a campionare e quantizzare il segnale in ingresso analogico generando in uscita un segnale numerico. La velocita' con cui viene effettuato il campionamento è espressa in campioni al secondo.

1.2 L'HEADER RIFF PER IL FORMATO WAVE

Come precedentemente anticipato **RIFF** è un insieme di specifiche che definiscono il formato per la memorizzazione elettronica digitale di dati, prevalentemente di tipo multimediale, come audio e video. Esso e' basato sui concetti di **Chunks** (in inglese chunk=pezzo) e **Sub-chunks**. Ogni Chunk è caratterizzato da un tipo ed è identificato da un'etichetta (Tag).

L'intero file RIFF è un grosso Chunk che a sua volta è suddiviso in altri Sub-Chunks. Il formato destinato all'audio è il **WAVE** (*.wav). Esso richiede due tipi di Chunks:

- **FMT CHUNK** , che descrive sample rate, sample width,…
- **DATA CHUNK** , che contiene i campioni.

Wave può anche contenere altri tipi di chunks permessi da RIFF, inclusi i chunks **LIST**, che vengono utilizzati per aggiungere informazioni opzionali quali data, copyright, nome dell'autore. Wave supporta differenti algoritmi di compressione. L'etichetta FORMATO (**Format Tag**) del chunk FMT indica il tipo di codifica utilizzata. Il valore 1 indica la codifica **PCM (Pulse Code Modulation)** , e quindi codifica senza compressione dei campioni. Valori superiori ne indicano, invece, la presenza.

L'header RIFF relativo alla forma canonica Wave è il seguente:

OFFSET	**LUNGHEZZA**	**CONTENUTO**
0	4 Bytes	'RIFF'
4	4 Bytes	< Lunghezza del File >
8	4 Bytes	'WAVE' ← Format Type
12	4 Bytes	'FMT'
16	4 Bytes	0x00000010 (Lunghezza del chunk FMT (16 bytes)
20	2 Bytes	0x0001 (Format Tag: 1=PCM)
22	4 Bytes	< Channel > (Channels: 1=Mono, 2=Stereo)
24	4 Bytes	< Sample Rate > (Campioni al secondo, ES: 44100 Hz)
28	4 Bytes	< Bytes/Seconds> (Sample Rate*Block Align)
32	2 Bytes	< Block Align > (Channels*Bits/Sample)
34	2 Bytes	< Bits/Sample > (8 o 16 bits)
36	4 Bytes	'DATA'
40	4 Bytes	< Lunghezza del chunck DATA >
44	… Bytes	< Campioni >

I campioni a 8 bit sono memorizzati come bytes senza segno nel range 0…255, mentre quelli a 16 bit come interi con segno in complemento a 2, nel range -32768…+32767. Tutti i valori sono digitalizzati secondo le specifiche Intel Little-Endian, cioè dal Byte meno significativo al piu' significativo.

Una descrizione più dettagliata delle specifiche RIFF e del formato WAVE sono disponibili consultando la documentazione Microsoft Win32 Multimedia API [6].

2. EFFETTI SPECIALI AUDIO

In questo capitolo vengono discussi il funzionamento del programma AudioFx, gli algoritmi per elaborare il segnale audio con l'inserimento di effetti speciali, il codice Matlab per la loro implementazione ed una serie di esempi d'applicazione.

2.1 AUDIOFX 1.0

Per l'elaborazione e lo studio degli effetti speciali su file audio digitali (formato wave) è stato progettato e realizzato un programma dedicato chiamato AudioFx. L'ambiente software in cui è stato sviluppato e lavora è Matlab 5.3 prodotto da MathWorks. La sua scelta è stata dettata dalla possibilità di sfruttare l'enorme potenza di calcolo ed i numerosi tools utili alla rappresentazione dei segnali numerici. In fig.2.1.1 è raffigurata la finestra principale di AudioFx. Esso è stato concepito con un'interfaccia grafica GUI (Graphic User Interface) permettendo così all'utilizzatore di lavorare in maniera più flessibile ed intuitiva ma, contemporaneamente, rigorosa.

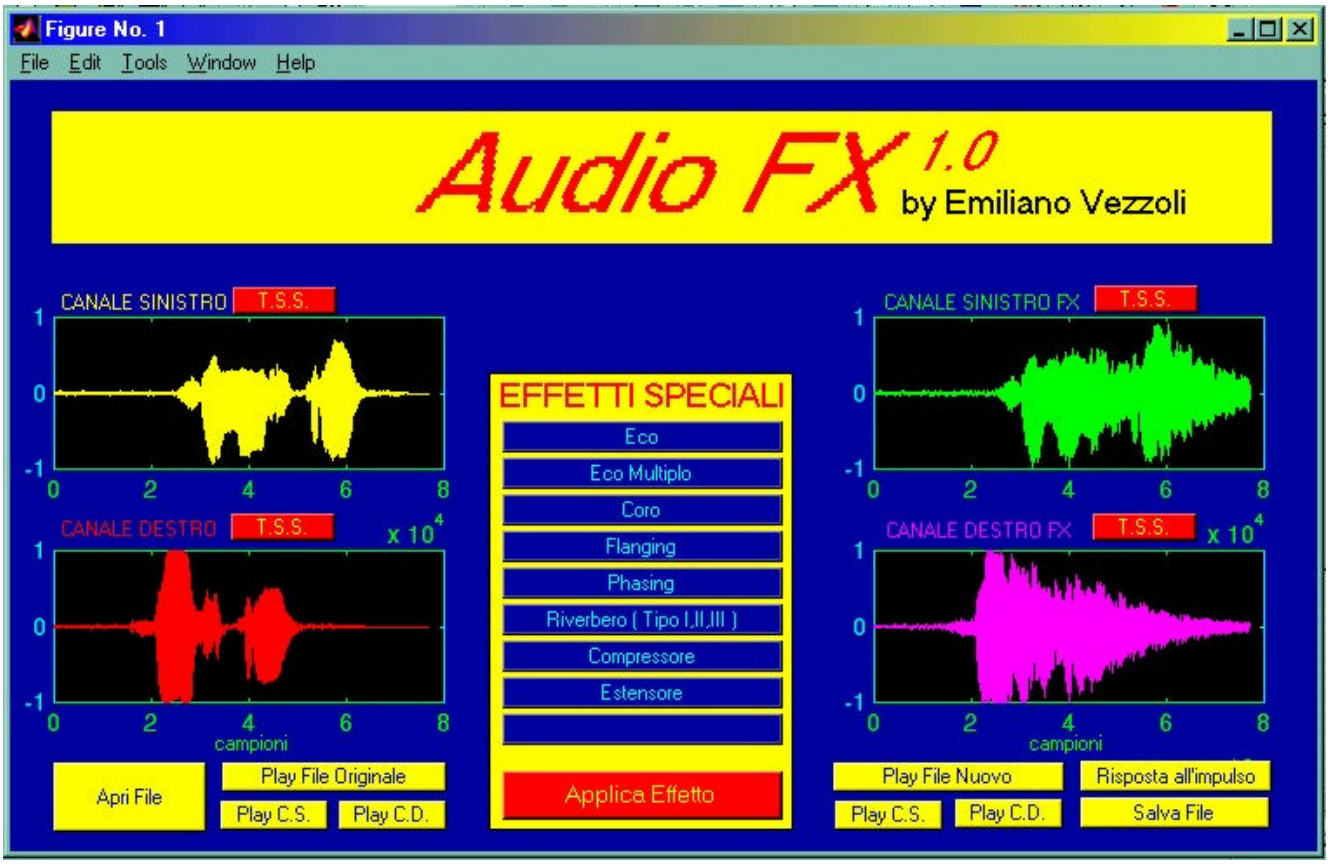


Fig. 2.1.1 Schermata principale del pacchetto applicativo Audiofx 1.0

Sulla sinistra vi sono tutte le informazioni sul file sorgente (o originale) a cui si vuole applicare un effetto. Vengono mostrati i grafici del canale sinistro e destro (se la traccia è stereofonica) con i campioni sull'asse delle ascisse ed i valori (normalizzati per rientrare nel range –1…+1) su quello delle ordinate. Cliccando sul tasto **T.S.S.** viene data la possibilità di visualizzare lo stesso grafico nel dominio temporale, la densità spettrale di potenza e lo spettrogramma. Inoltre si possono ascoltare i canali audio separatamente o insieme. Analogamente per la parte destra dello schermo dove si possono effettuare le stesse operazioni ma sul segnale filtrato, con l'aggiunta della risposta all'impulso e la possibilità di salvare il file. Va notato immediatamente che i tutti i grafici vengono mostrati come funzioni continue ma, in realtà sono segnali numerici!!! Si è preferito questo modo rappresentativo per meglio osservare i vari aspetti del filtraggio. Un discorso analogo va fatto per la risposta all'impulso, dove si considera un ingresso di soli 100 campioni. Per vedere in modo qualitativo il comportamento dei filtri al variare dei vari parametri, all'impulso e' stata preferita una sequenza di 10 campioni unitari dopo i primi cinque nulli, sequita da tutti zeri.

Nella zona centrale è presente il menù con gli effetti speciali disponibili. Prima infatti di applicarli (cliccando il tasto rosso apposito) viene richiesto l'inserimento di tutta una serie di parametri necessari al filtraggio. Come per esempio: ritardo massimo insecondi, attenuazione minima. La trattazione di questi è rimandata ad ogni singolo effetto.

In fig. 2.1.2 viene mostrato un esempio di finestra per l'inserimento dei parametri di un filtro.

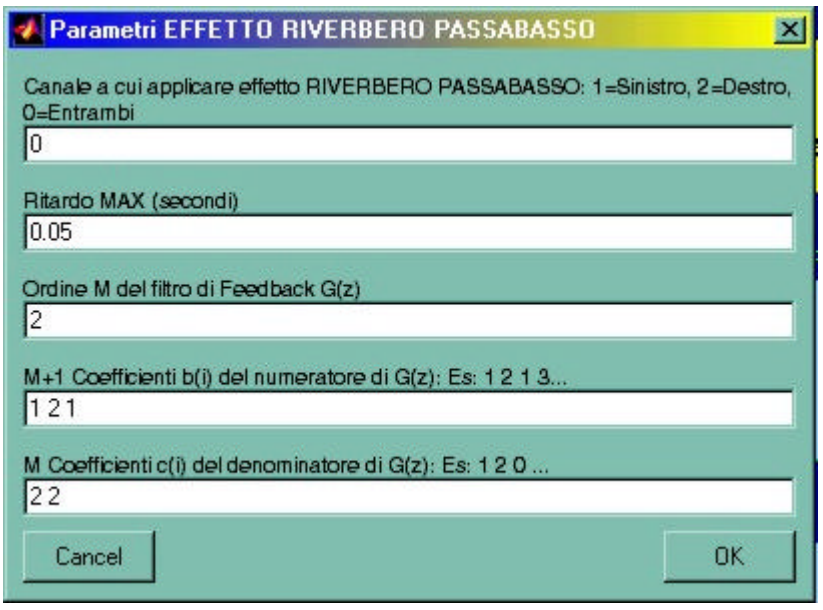


Fig. 2.1.2 Finestra inserimento parametri per l'effetto Riverbero Passabasso.

I tests sul corretto funzionamento di AudioFx sono stati eseguiti su vari segnali audio molto brevi, a 8 e 16 bit, mono e stereofonici. Di seguito vengono riportate le caratteristiche relative al canale sinistro dei più significativi. E' stato interessante provare generi musicali differenti , quali classica e dance, oltre al segnale vocale, perché distinti per contributi frequenziali e spettrali.

Fig. 2.1.3 :	**Brano03.wav**	Segnale Vocale	2 secondi	16 bit	Stereo	44100 Hz
Fig. 2.1.4 :	**BranoPop.wav**	Musica Pop	4 secondi	16 bit	Stereo	22050 Hz
Fig. 2.1.5:	**BranoCla.wav**	Musica Classica	4 secondi	16 bit	Stereo	22050 Hz
Fig. 2.1.6:	**BranoDan.wav**	Musica Dance	4 secondi	16 bit	Stereo	22050 Hz

Va ricordato che la densità spettrale di potenza mostra la potenza del segnale come funzione della frequenza (indicata in Hz) mentre lo spettrogramma rappresenta il contenuto frequenziale del segnale al variare del tempo (i picchi sono evidenziati da macchie più scure , mentre sull'asse delle ascisse si trova la frequenza, normalizzata a valori compresi tra 0 e 1).

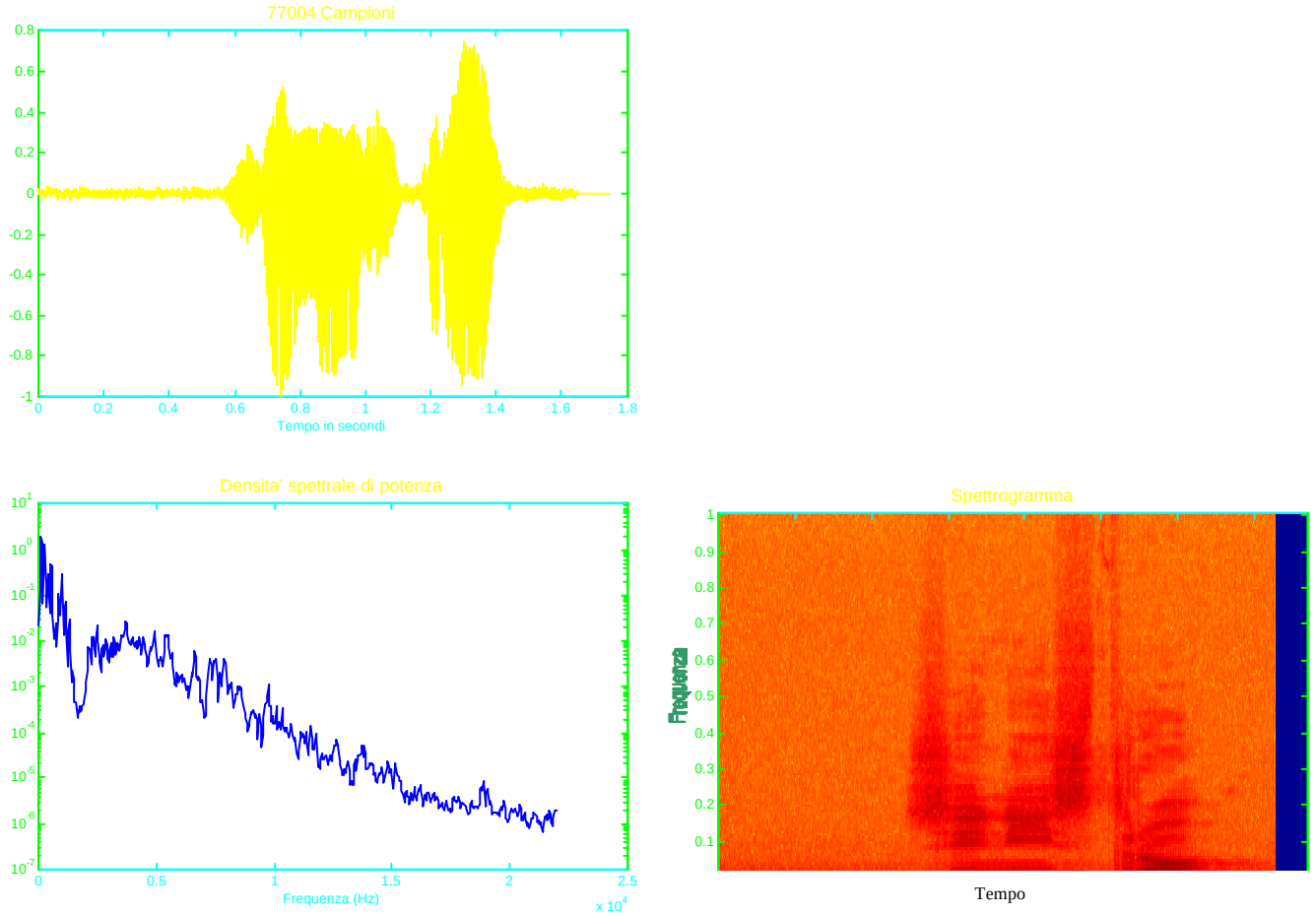
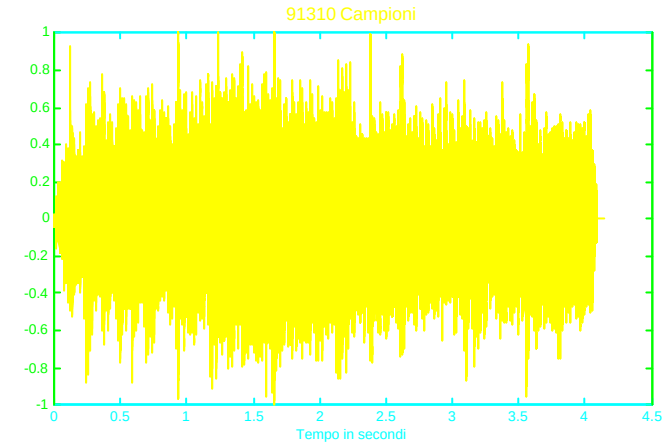
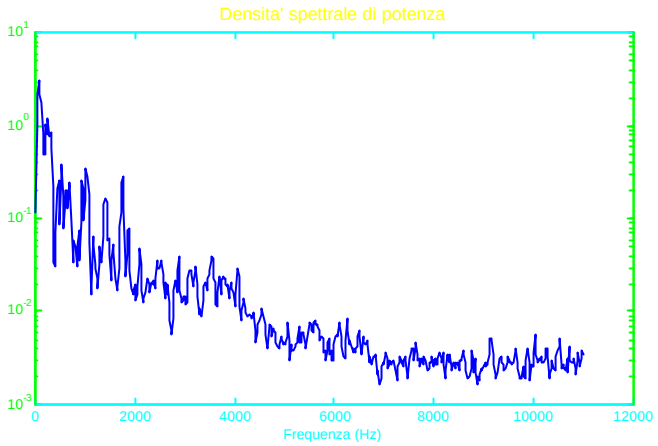
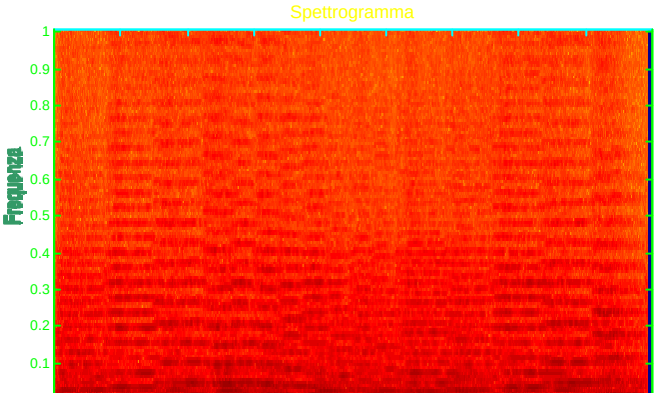


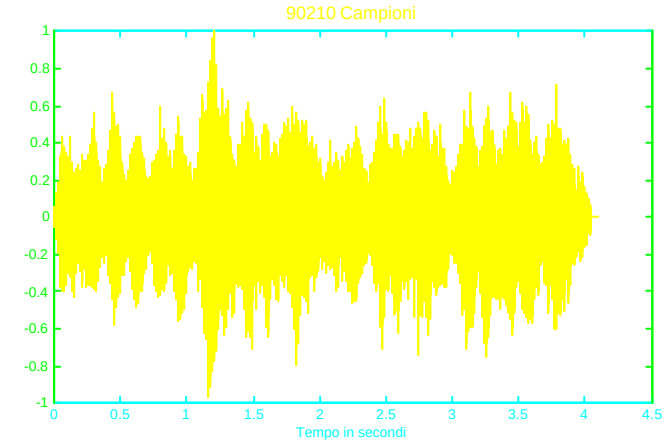
Fig 2.1.3 Canale sinistro di Brano03.wav

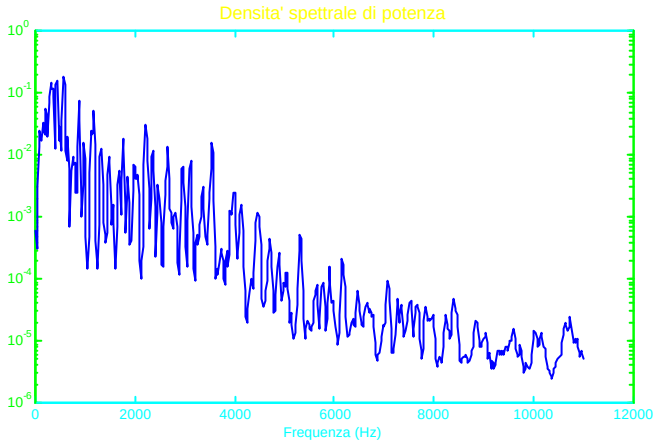


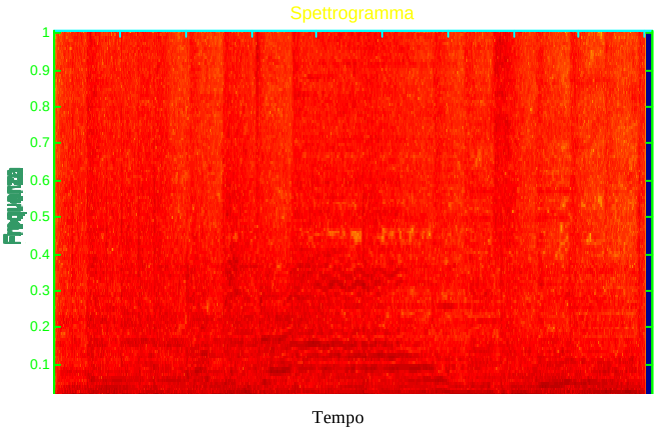




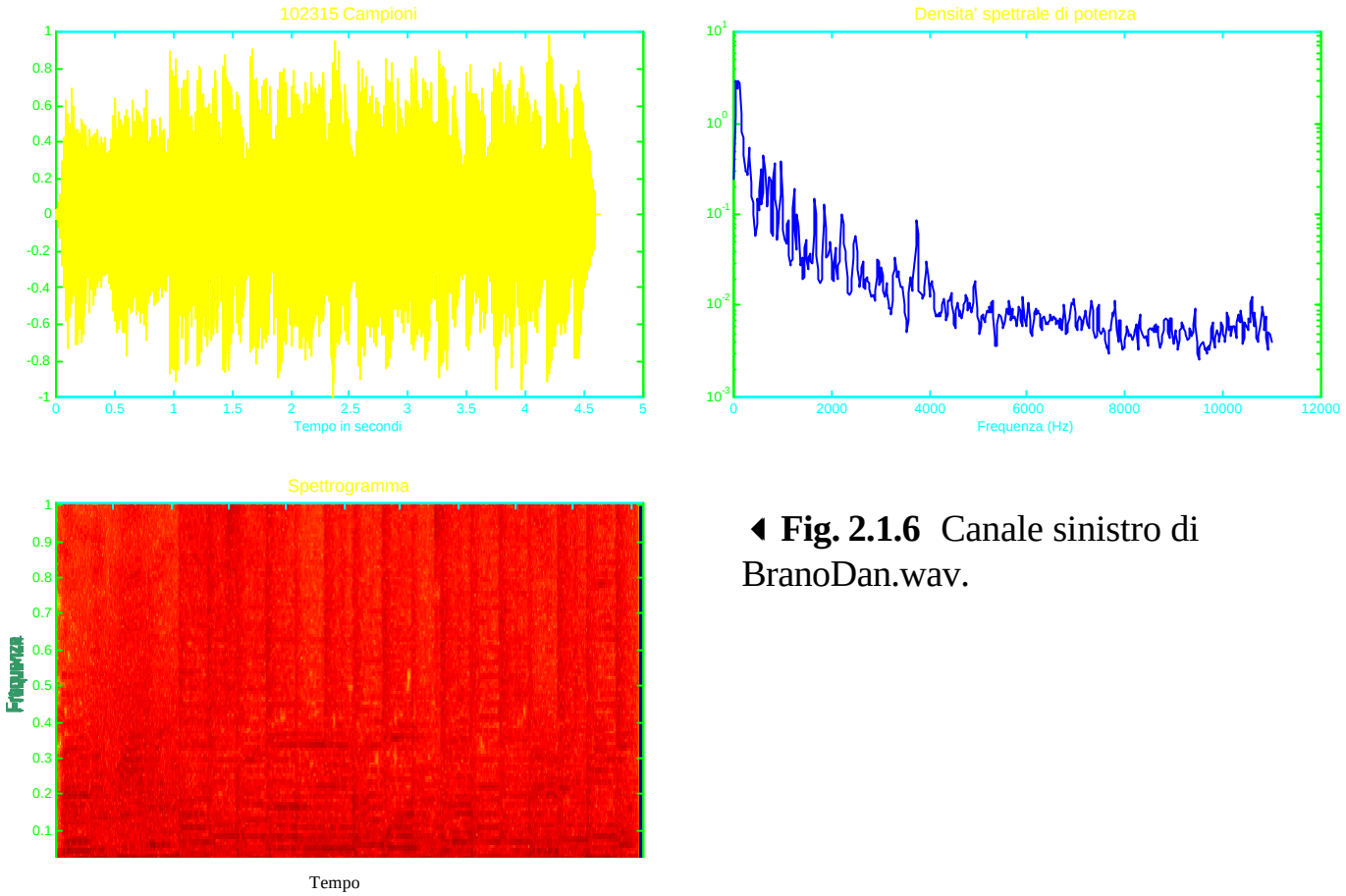
◂ **Fig. 2.1.4** Canale sinistro di BranoPop.wav.







◂ **Fig. 2.1.5** Canale sinistro di BranoCla.wav.



◂ **Fig. 2.1.6** Canale sinistro di BranoDan.wav.

I grafici riportati nelle figure precedenti serviranno come punto di riferimento per eventuali confronti con i nuovi segnali generati attraverso i filtri per effetti speciali. Già adesso, però, può essere utile osservare le differenze tra i brani audio proposti. Nella traccia dance BranoDan.wav , infatti, prevalgono frequenze molto più basse rispetto agli altri generi musicali, in corrispondenza di valori più elevati della densità spettrale di potenza. E' più simile ad un brano di musica pop che non al brano di classica o al segnale vocale dove, addirittura, i contributi frequenziali sono maggiormente distribuiti.

2.2 ECHI (SINGOLI E MULTIPLI)

Nell'ambito dell'elaborazione audio, l'effetto speciale più semplice e costituente il "blocco base" per tutti gli altri effetti è il **ritardo elementare**. Si parla anche di ritardatore. Il suo scopo è simulare il suono risultante dalla somma dell'onda acustica che raggiunge direttamente l'ascoltatore con quelle riflesse da eventuali pareti, oggetti di una stanza o di una sala da concerto. Quest'ultime saranno ritardate ed attenuate rispetto all'originale.

Se si considera un eco semplice composto da una sola riflessione, la sua funzione di trasferimento è del tipo:

$$\mathbf{H(z) = 1 + a \cdot z^{-D}}$$

Dove **D** è il ritardo in campioni (cioè Ritardo in secondi * Frequency Sample) ed **a** è l'attenuazione ($|a| \leq 1$).

In termini di equazione alle differenze si ha:

$$\mathbf{y(n) = x(n) + a \cdot x(n - D)}$$

Come si può notare si tratta di un filtro **FIR** (Risposta all'Impulso Finita) lineare tempoinvariante. Infatti, se $x(n)=x_1(n)+ x_2(n)$ allora:

$$y(n) = x_1(n) + a \cdot x_1(n - D) + x_2(n) + a \cdot x_2(n - D) = y_1(n) + y_2(n)$$

- **Linearità.**

Mentre se applico x(n-k) allora:

$$y\left[x(n - k)\right] = x(n - k) + a \cdot x(n - k - D) = y(n - k)$$

- **TempoInvarianza.**

Si ricorda che un filtro FIR è sempre stabile, per costruzione!!
Un'altra importante considerazione riguarda la causalità del sistema. Dato che stiamo considerando segnali audio, d'ora in avanti considereremo tutti i sistemi come causali. Non avrebbe senso fare il contrario. Un suono non è definito per istanti minori di zero!

In termini di risposta in frequenza, ponendo $z=e^{jw}$:

$$\mathbf{H(w) = 1 + a \cdot e^{-jw}}$$

$$\left|\mathbf{H}(w)\right| = \sqrt{\mathbf{1 + 2a \cdot \cos(w \cdot D) + a^2}}$$

In fig. 2.2.1 sono mostrati uno schema a blocchi del generatore d'eco singolo e la sua risposta all'impulso.

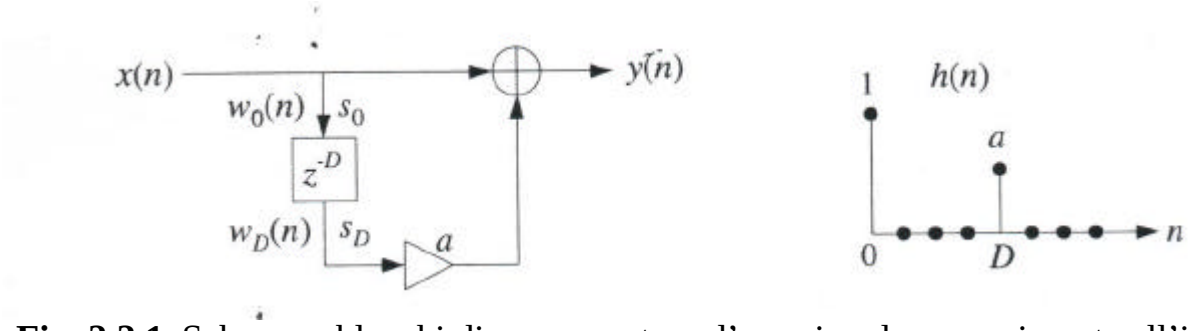


Fig. 2.2.1 Schema a blocchi di un generatore d'eco singolo e sua risposta all'impulso.

Di seguito vengono riportati alcuni esempi applicativi dell'effetto eco, in fig.2.2.2, fig.2.2.3, fig.2.2.4, figg.2.2.5 e 2.2.6.

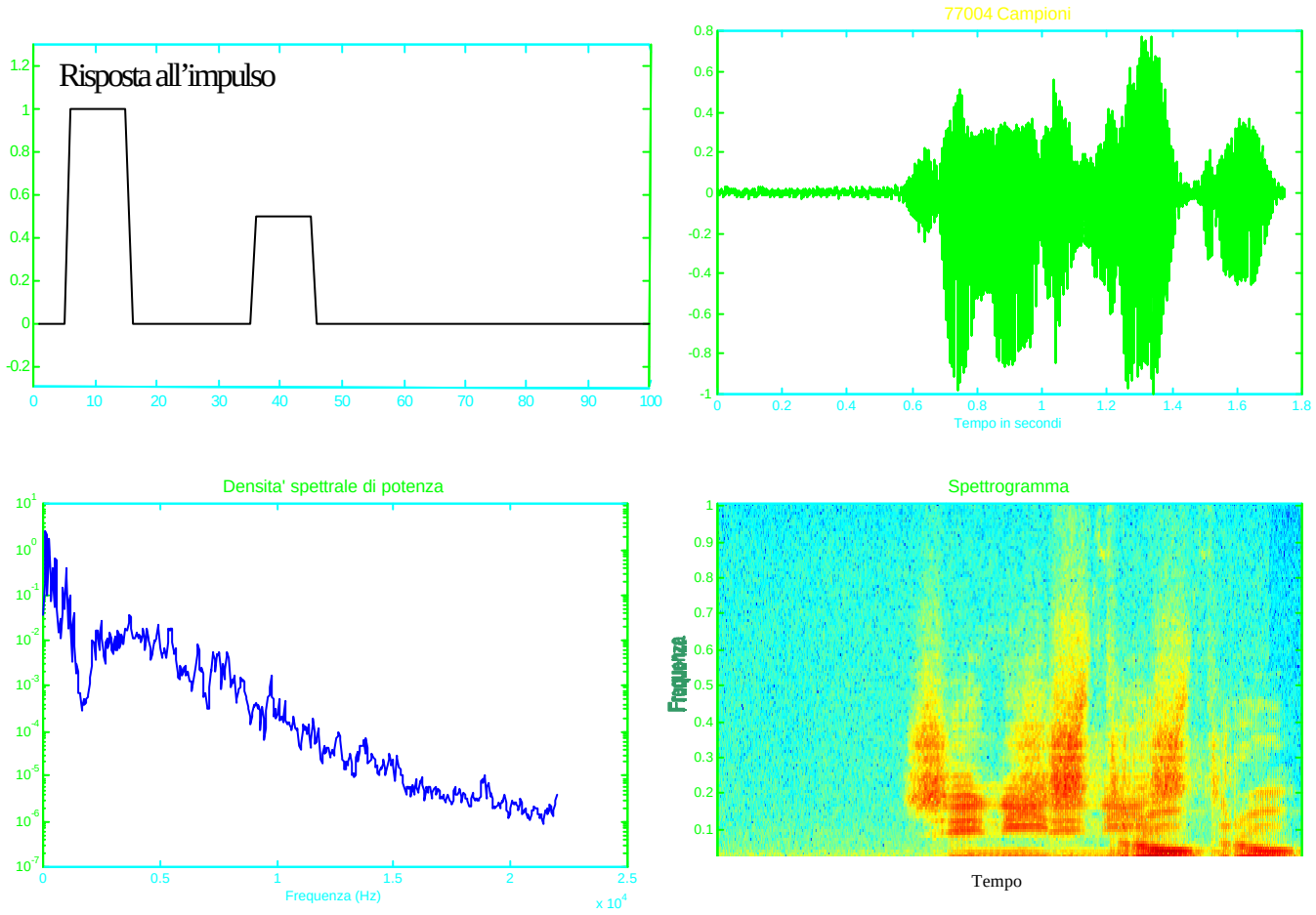


Fig. 2.2.2 Effetto Eco Singolo su Brano03.wav con i seguenti parametri:
Attenuazione **a**=0,5
Ritardo Max **d**=0,3 secondi

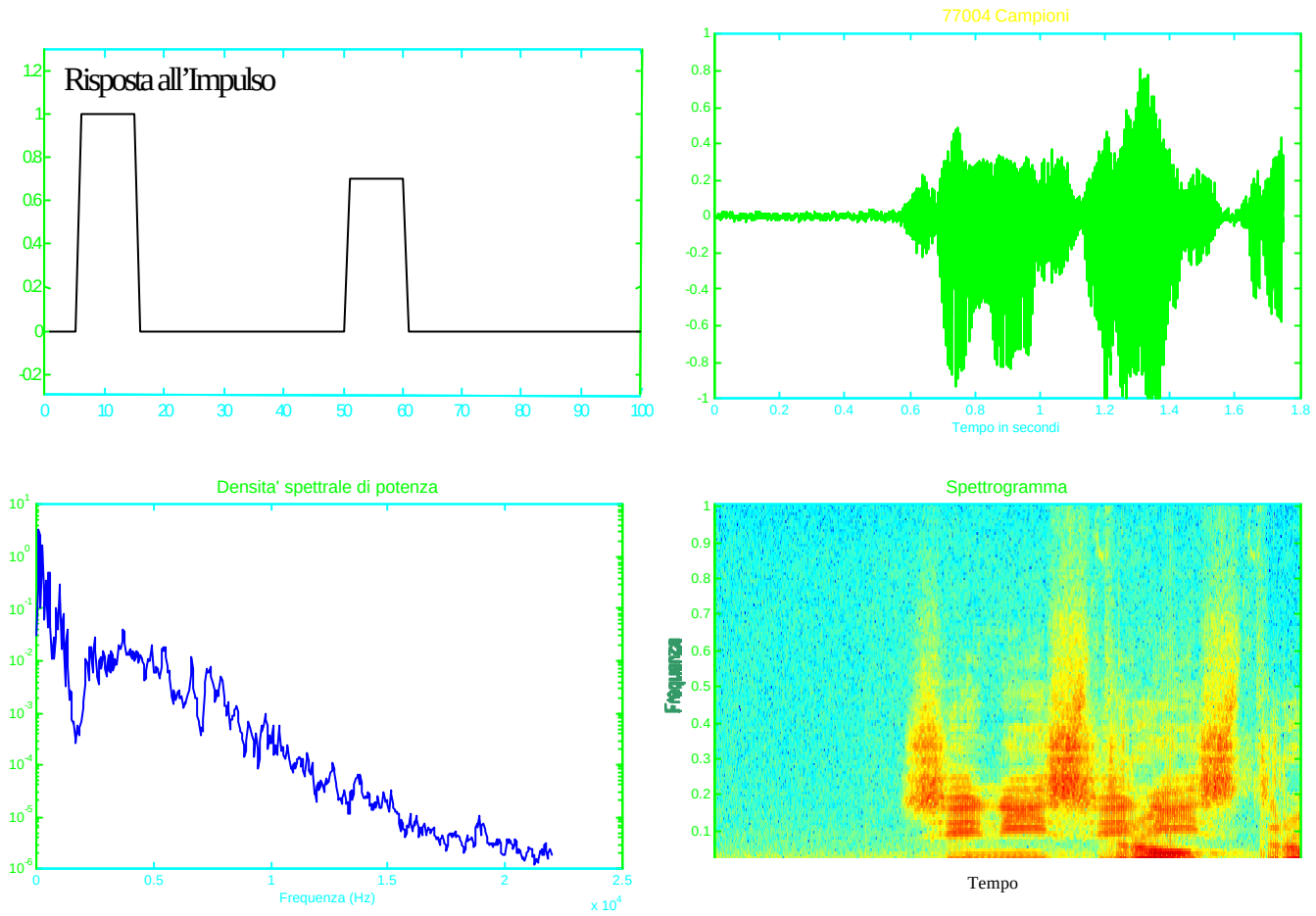
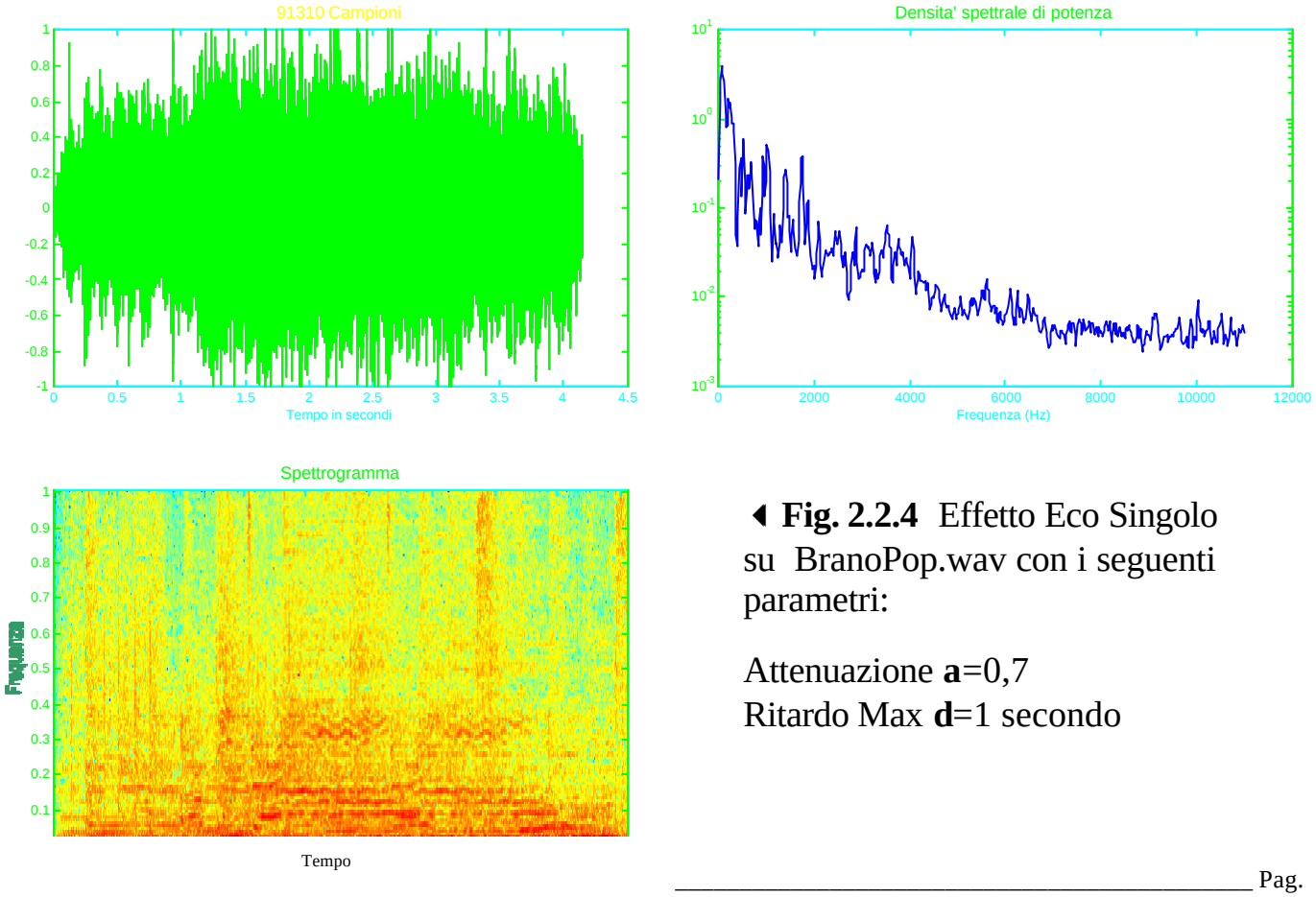
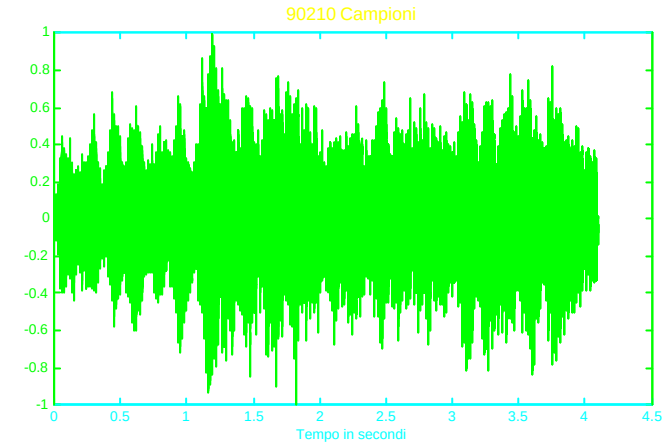


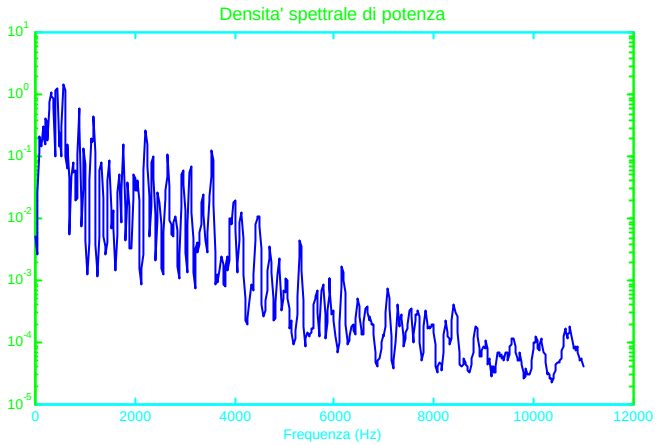
Fig. 2.2.3 Effetto Eco Singolo su Brano03.wav con i seguenti parametri:
Attenuazione **a**=0,7 ; Ritardo Max **d**=0,45 secondi

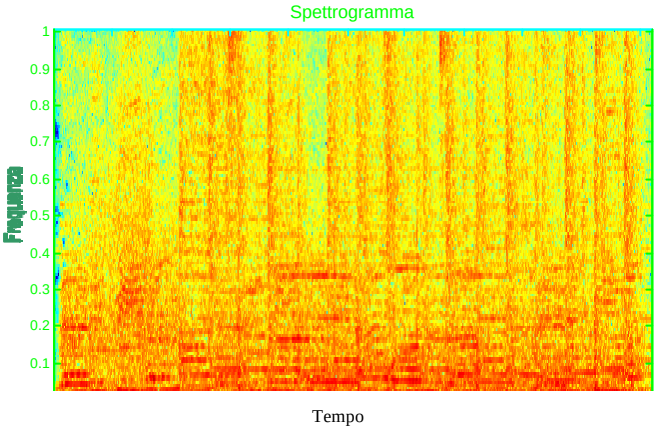


◂ **Fig. 2.2.4** Effetto Eco Singolo su BranoPop.wav con i seguenti parametri:

Attenuazione **a**=0,7
Ritardo Max **d**=1 secondo

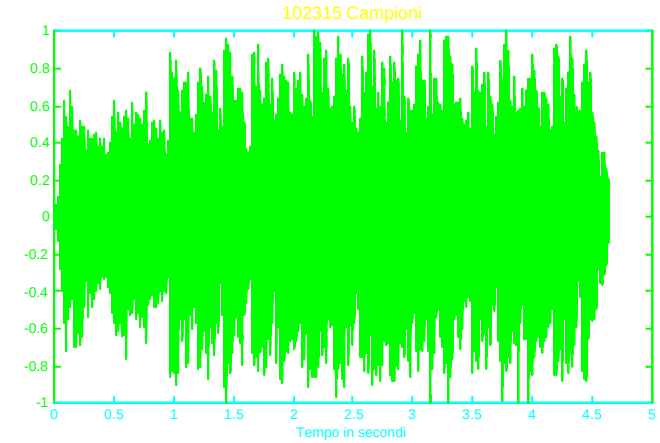


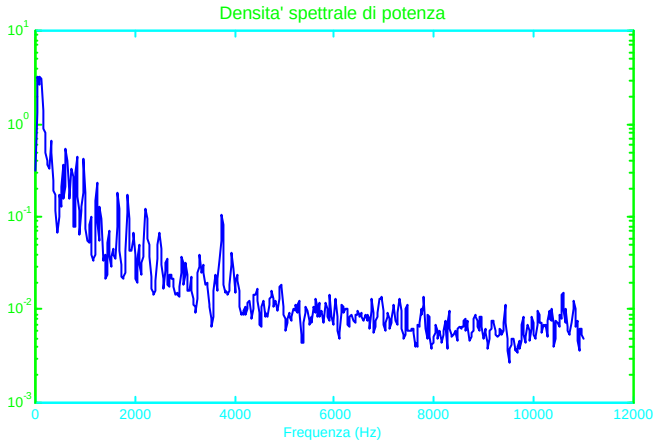


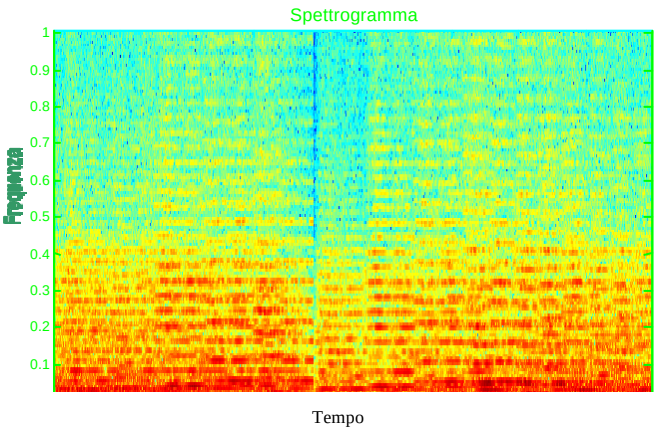


◂ **Fig. 2.2.5** Effetto Eco Singolo su BranoCla.wav con i seguenti parametri:

Attenuazione **a**=0,6
Ritardo Max **d**=0,5 secondi







◂ **Fig. 2.2.6** Effetto Eco Singolo su BranoDan.wav con i seguenti parametri:

Attenuazione **a**=0,5
Ritardo Max **d**=1,2 secondi

Se vogliamo raffinare l'effetto Eco per meglio simulare le infinite riflessioni duvute alle pareti di una stanza otteniamo un effetto chiamato **ECO MULTIPLO** ma più generalmente indicato con il nome di **RIVERBERO PIANO**.

La sua funzione di trasferimento è la seguente:

$$\mathbf{H(z) = \frac{1}{1 - a \cdot z^{-D}}}$$

Dove **a** è l'attenuazione minima ($|a| \leq 1$) e **D** è il ritardo in campioni (cioè Ritardo in secondi * Frequency Sample). In fig. 2.2.7 sono raffigurati lo schema a blocchi e la risposta all'impulso. In termni di equazione alle differenze:

$$\mathbf{y(n) = a \cdot y(n - D) + x(n)}$$

E' evidente, osservando tale espressione, la sua differenza con l'eco singolo. Questo, infatti, è un filtro **IIR** (Infinite Impulse Response), cioè con risposta all'impulso infinita. Affinchè tale sistema sia stabile è necessario che:

$$\mathbf{\sum |h(n)| \leq M_h \leq \infty}$$

AudioFx applica tale filtro sempre su un numero finito di campioni, per cui risulterà sempre stabile. La ripetizione degli Echi ogni D campioni corrisponde alla ripetizione periodica con frequenza $f_1=(f_s/D)$Hz, cioè $\omega_1=2\pi/D$ come mostrato in fig. 2.2.8.

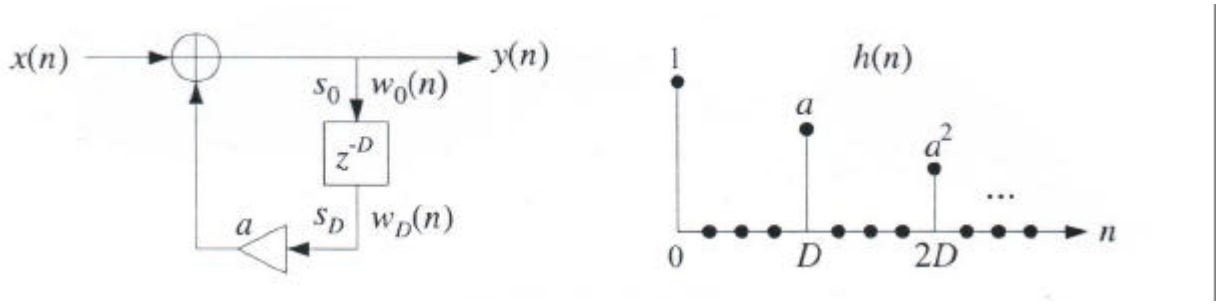


Fig. 2.2.7 Schema a blocchi e Risposta all'Impulso dell'Eco Multiplo.

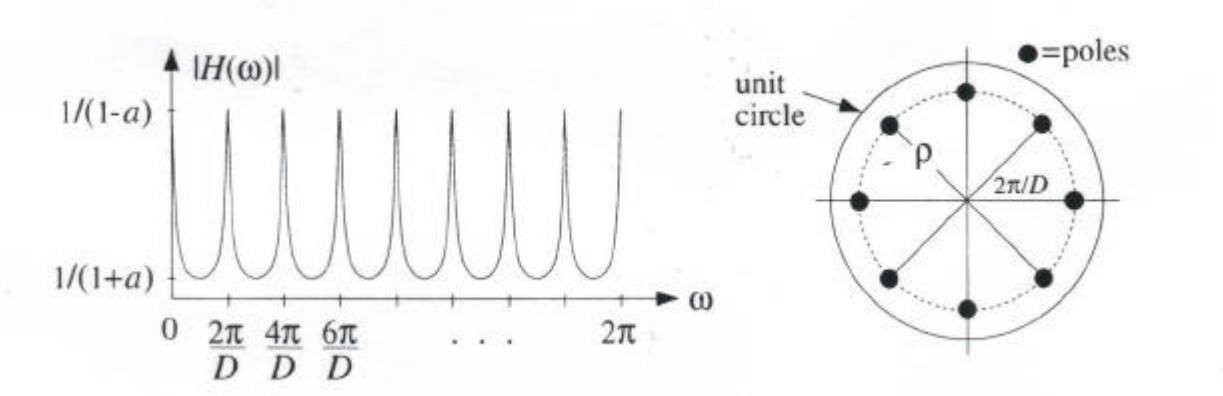


Fig. 2.2.8 $|H(\omega)|$ con i picchi a $\omega_k=2\pi k/D$, k=0,1,…,D-1.

Di seguito un esempio semplificato di implementazione Matlab del filtro che crea l'eco multiplo, mentre nelle figg. 2.2.9 , 2.2.10 e 2.2.11 l'effetto applicato tramite AudioFx al canale sinistro del file Brano03.wav e BranoCla.wav.

```
% Effetto Eco Multiplo – Matlab
  for i=(D+1):cmp        % cmp è il numero dei campioni
    y(i)=a*y(i-D)+x(i);
  end
```

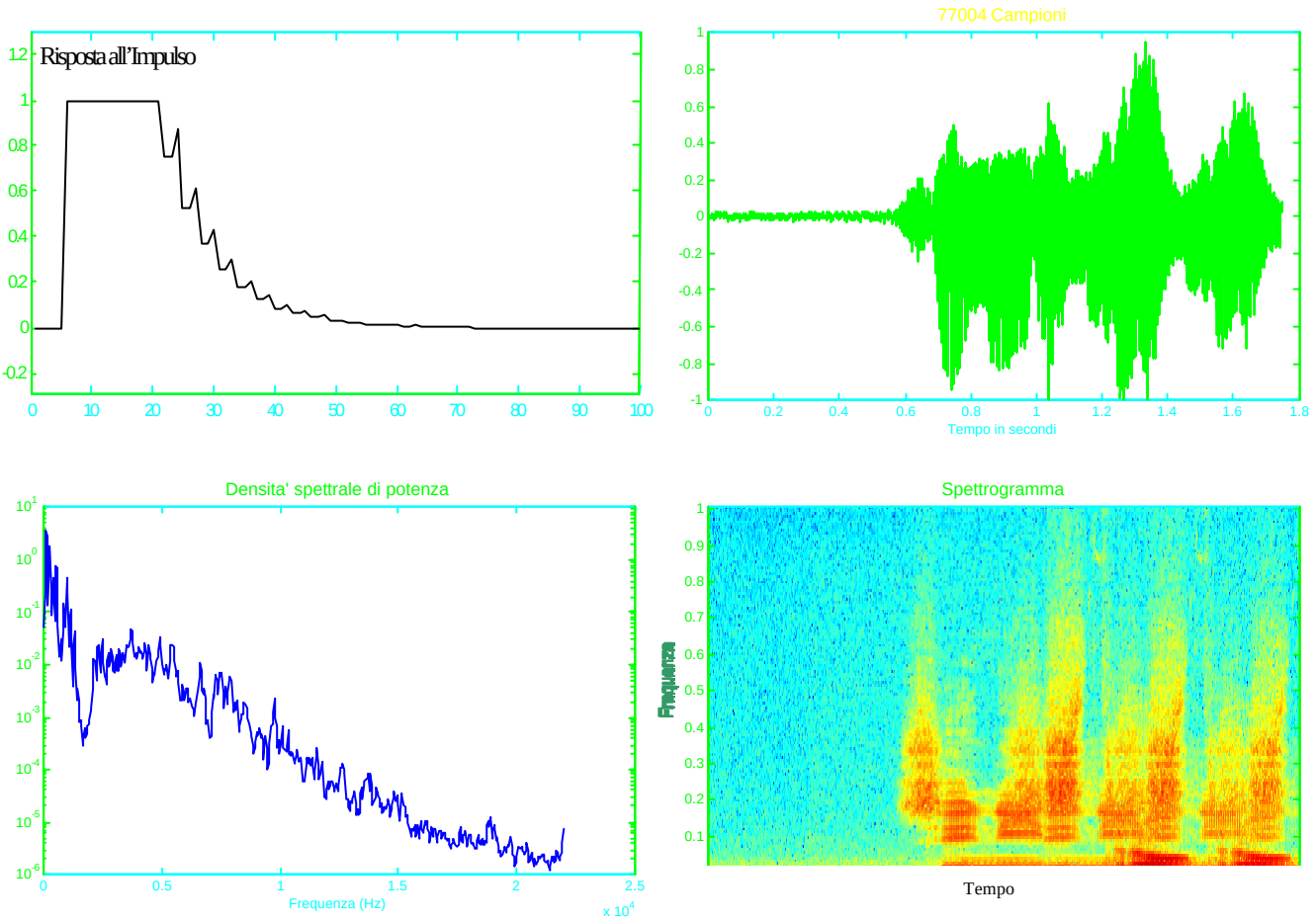
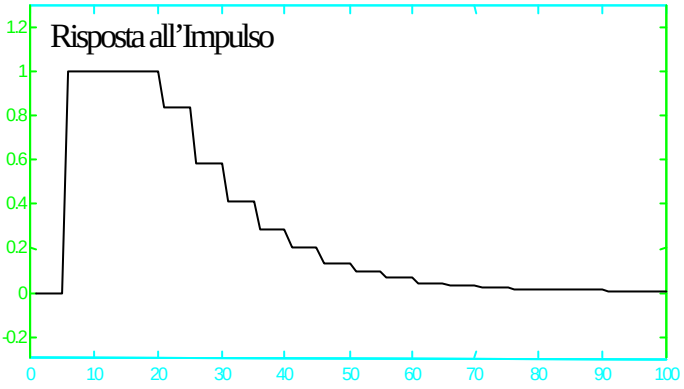
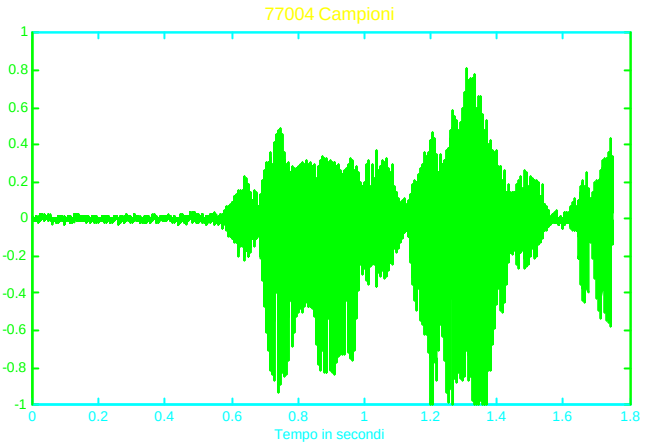
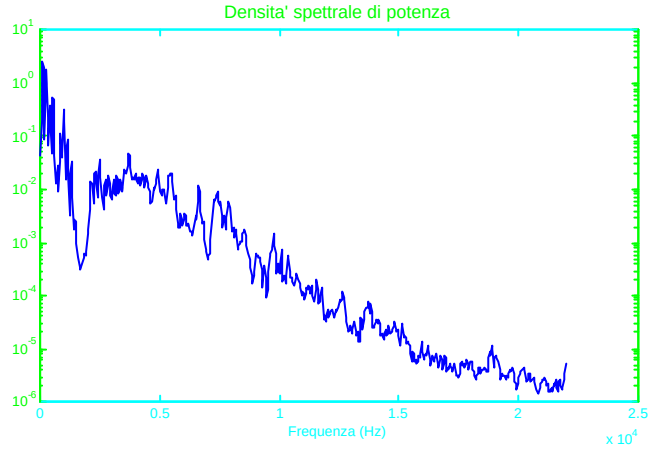


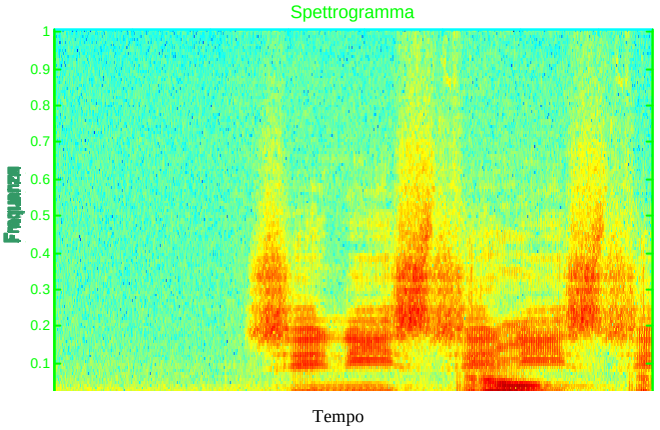
Fig. 2.2.9 Effetto Eco Multiplo su Brano03.wav con i seguenti paramentri:
Attenuazione **a**=0,7
Ritardo Max **d**=0,3 secondi

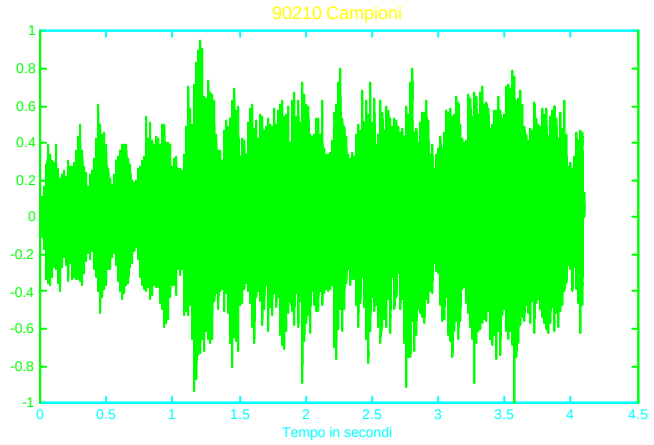
In fig. 2.2.11 l'effetto Eco Multiplo su BranoCla.wav. A differenza del segnale vocale, dove tale effetto è particolarmante apprezzabile, in un brano musicale come quello classico si ha la sensazione di trovarsi in una sala da concerto, dove l'effetto riverberante si riperquote sull'ascoltatore creando una sensazione di profondità dello spazio. I risultati dei tests sui brani dance e pop, invece, non sono stati riportati perche' non significativi. In questi casi, infatti, l'enorme quantita' di strumenti e suoni elettronici presenti già nel suono originale rende poco distinguibile un effetto come l'Eco Multiplo.





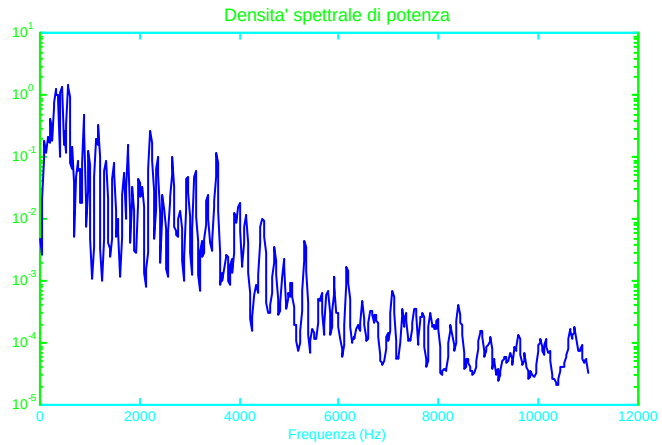


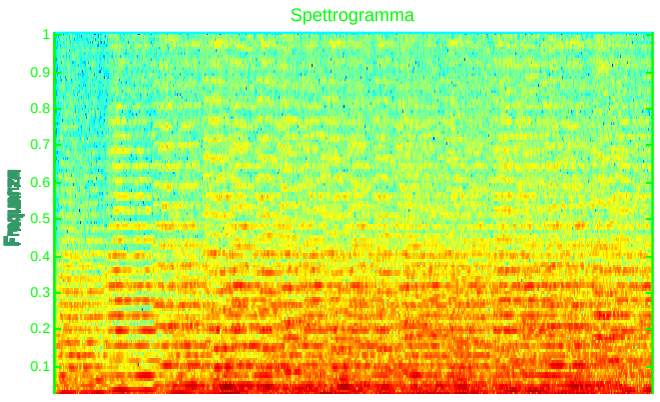




▲ **Fig. 2.2.10** Effetto Eco Multiplo su Brano03.wav con i seguenti paramentri: Attenuazione **a**=0,5 ; Ritardo Max **d**=0,5 secondi

◀ **Fig. 2.2.11** Effetto Eco Multiplo su BranoCla.wav con i seguenti paramentri: Attenuazione **a**=0,7 ; Ritardo Max **d**=0,8 secondi





2.3 FLANGING

Fino ad ora è stato considerato il ritardo D come costante. Se lo facciamo variare nel tempo (in un range finito!!) otteniamo l'effetto **FLANGING**. Questo tipo di effetto conferisce un'impronta quasi "metallica" alla traccia audio tanto che è stato utilizzato spesso per distorcere la voce di alieni ed extraterrestri in videogames e serie TV. Possiamo descriverlo tramite la sequente equazione alle differenze:

$$\mathbf{y(n) = x(n) + a \cdot x(n - d(n))}$$

Con $\mathbf{d(n)=(D/2)(1-\cos(2\pi nF_d))}$, dove $0 \le d(n) \le D$, **D**=Ritardo (in campioni), F_d è la frequenza (in cicli/campione) con cui varia il ritardo d(n). Il suo algoritmo è mostrato nello schema a blocchi di fig. 2.3.1.

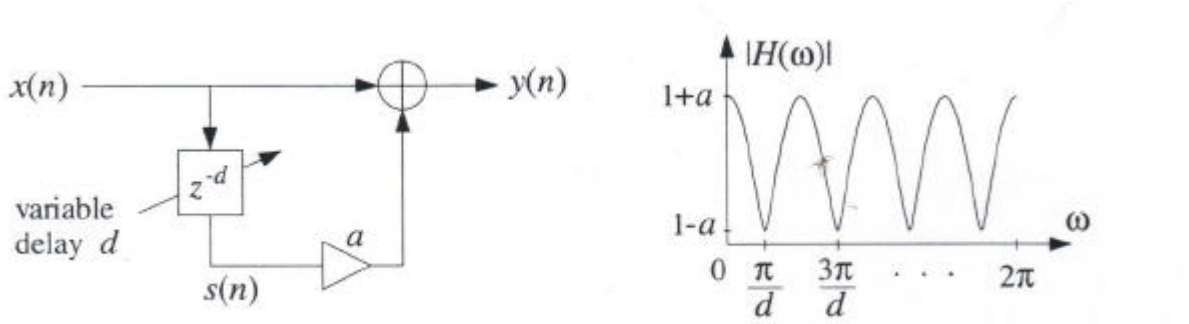


Fig. 2.3.1 Schema a blocchi e modulo della risposta in frequenza del generatore di effetto Flanging.

d(n) può assumere valori non interi compresi tra 0 e D, ma x(n-d(n)) richiede che d(n) sia un intero!! Esistono allora varie tecniche per risolvere questo problema: troncamento, arrotondamento, interpolazione lineare (quest'ultima utile solo per basse frequenze),…

In AudioFx si è optato per l'arrotondamento al valore intero più vicino. Si può osservare che, nonostante la presenza di d(n) variabile, il filtro di Flanging è **LINEARE**. Infatti se $x(n)=x_1(n)+ x_2(n)$ allora:

$$\mathbf{y\left[x_1(n)+x_2(n)\right] = x_1(n)+x_2(n)+a\cdot\left[x_1(n-d(n))+x_2(n-d(n))\right] = y_1(n)+y_2(n)}$$

…ma è **TempoVariante**. Applicando come ingresso x(n-k) ottengo in uscita:

$$y[x(n-k)]=x(n-k)+a\cdot x(n-k-d(n)) \quad . \quad y(n-k)=x(n-k)+a\cdot x(n-k-d(n-k))$$

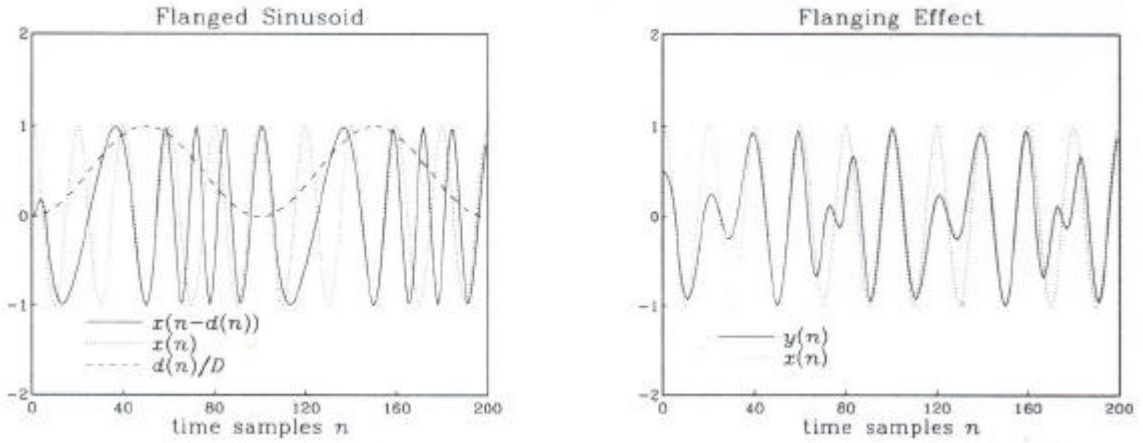


Fig. 2.3.2 Esempio di come agisce il filtro di Flanging. A sinistra la creazione della sinusoide Flanged a periodo variabile, mentre a destra il suo effetto sull'uscita.

Nelle figg. 2.3.3, 2.3.4 e 2.3.5 sono riportati 3 esempi applicativi dell'effetto flanging sul canale sinistro del file Brano03.wav e BranoDan.wav. Interessante osservare come cambia la densità spettrale di potenza.

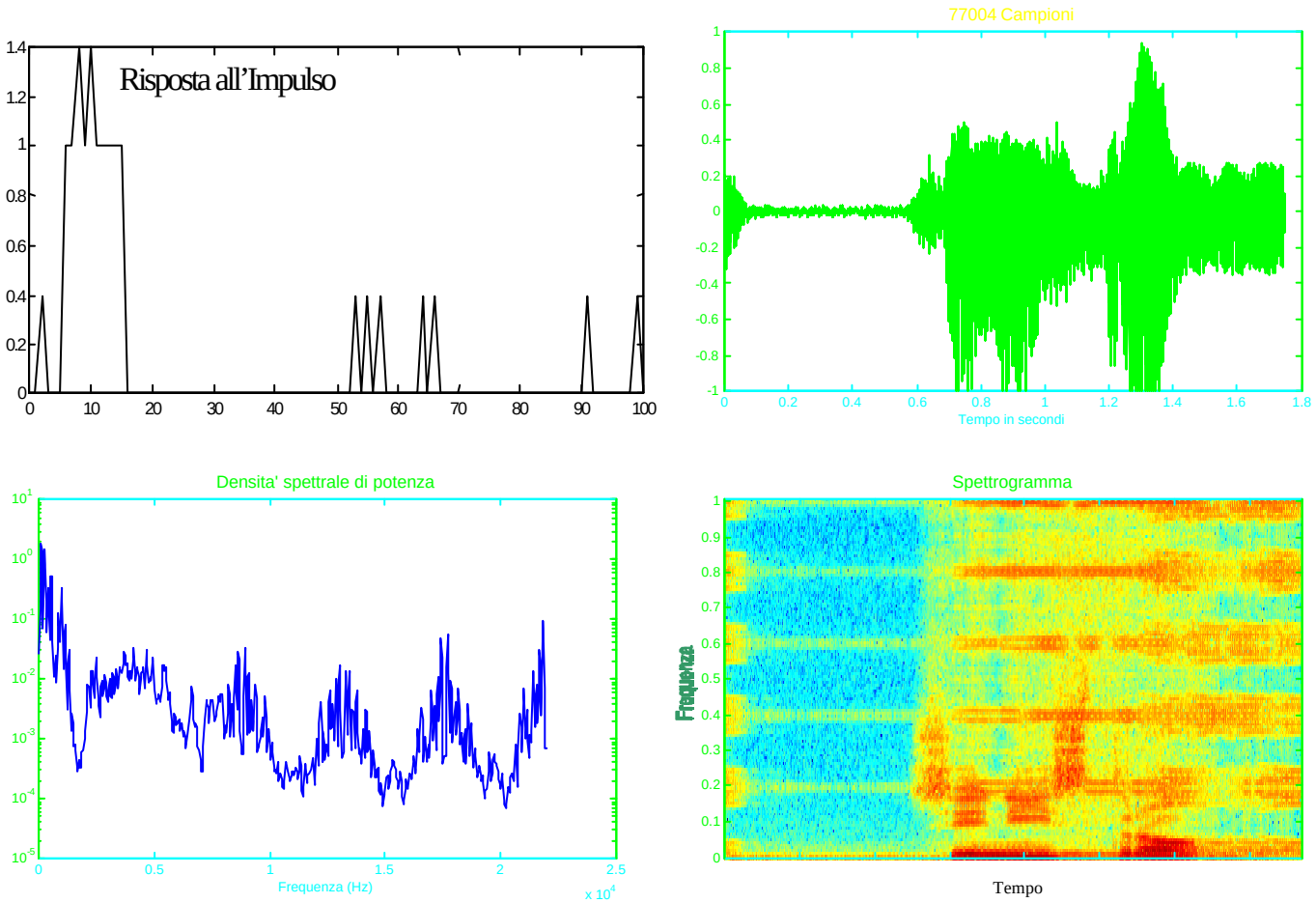


Fig. 2.3.3 Effetto Flanging su Brano03.wav con i seguenti parametri:
Ritardo Max **d**=0,4 secondi
Attenuazione minima **a**=0,4
Frequenza Variazione Ritardi **Freq**=0,1πrad/sec

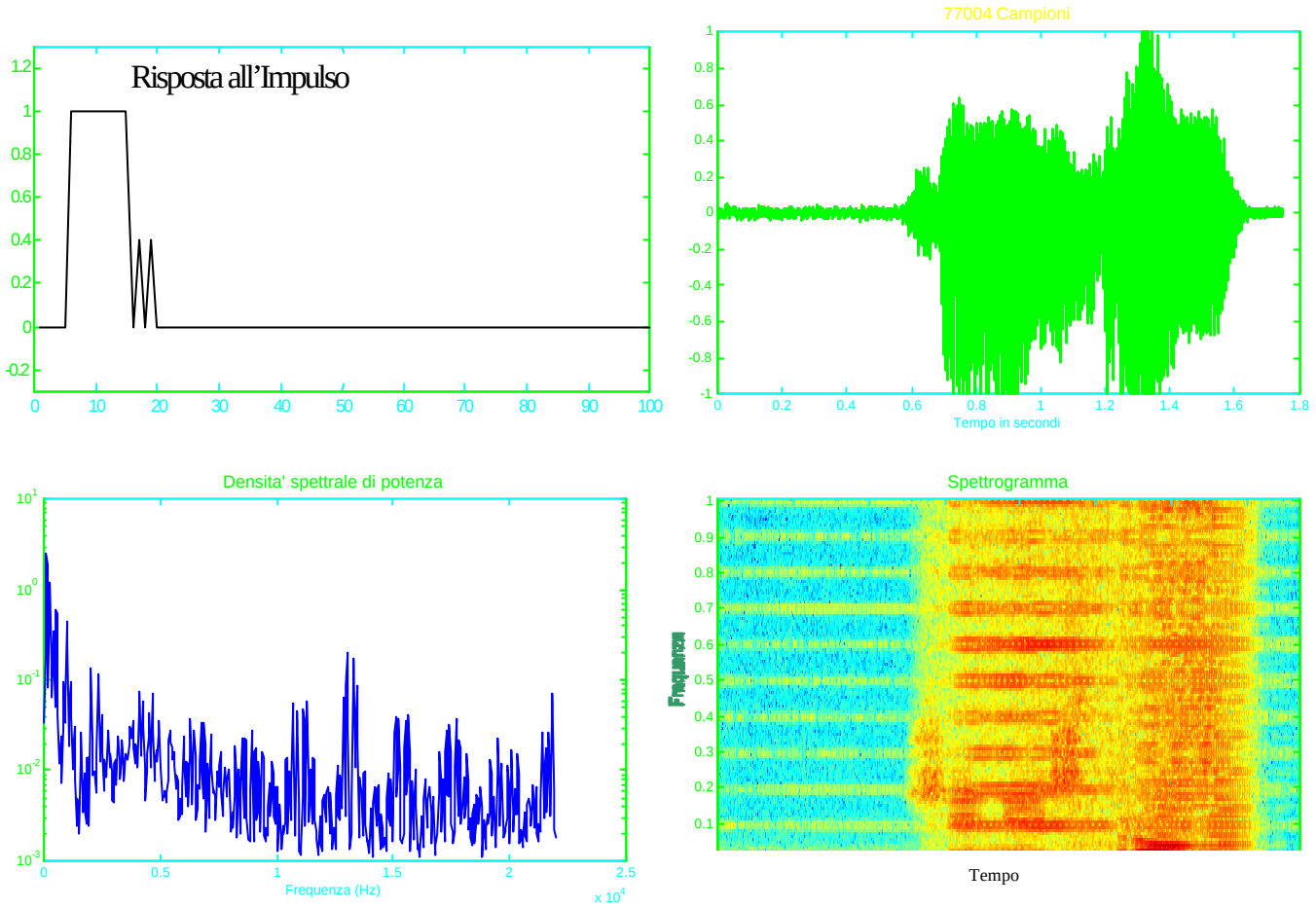


Fig. 2.3.4 Effetto Flanging su Brano03.wav con i seguenti parametri:
Ritardo Max **d**=0,2 secondi
Attenuazione minima **a**=0,8
Frequenza Variazione Ritardi **Freq**=0,05 πrad/sec

Qui sotto viene mostrata una possibile implementazione Matlab di questo effetto speciale.

```
% Effetto Flanging – Matlab
for k=1:cmp
   % Creo un vettore di ritardi variabili nel tempo
   d2(k)=(D/2)*(1-cos(2*pi*Freq*k));
   % Arrotondamento per avere ritardi interi
   d(k)=round(d2(k));
   % Buffer di Ritardi circolari
   indice(k)=k-d(k);
     while ((indice(k)<=0)
        indice(k)=indice(k)+cmp;
     end
   % Filtro Flanging
   y(k)=x(k)+(a*x(indice(k)));
end
```

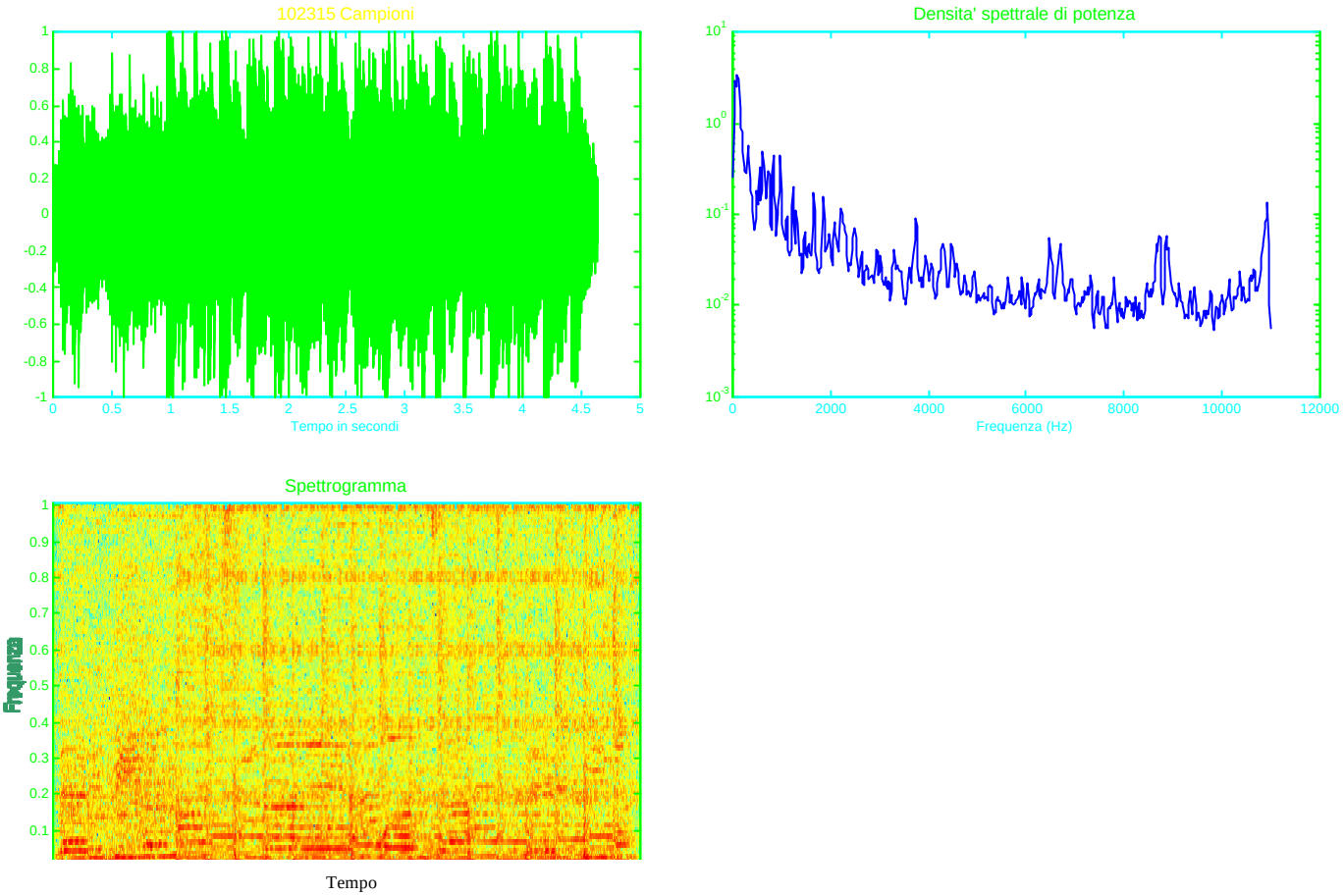


Fig. 2.3.5 Effetto Flanging su BranoDan.wav con i seguenti parametri:
Ritardo Max **d** = 0,4 secondi
Attenuazione minima **a** = 0,4
Frequenza Variazione Ritardi **Freq** = 0,1 πrad/sec

2.4 IL CORO

L'effetto CORO simula la presenza di un gruppo di più voci (musicisti,….) che riproducono lo stesso brano simultaneamente. Tali voci sono più o meno sincronizzate con le altre, eccetto per piccole variazioni in ampiezza e nel tempo. Un possibile esempio di Coro a k voci è il seguente sistema lineare tempovariante:

$$\mathbf{y(n) = x(n) + a_1 \cdot x(n - d_1(n)) + a_2 \cdot x(n - d_2(n)) + \ldots + a_k \cdot x(n - d_k(n))}$$

Dove :

$$\mathbf{d_i(n) = D \cdot (0{,}5 + v(n))}$$

Con **D** ritardo max in campioni e **v(n)** che varia casualmente nell'intervallo –0,5…+0,5.
In fig. 2.4.1 un esempio di schema a blocchi per un generatore di Effetto Coro a sole 3 voci mentre in fig 2.4.2 l'effetto coro sul segnale x(n).

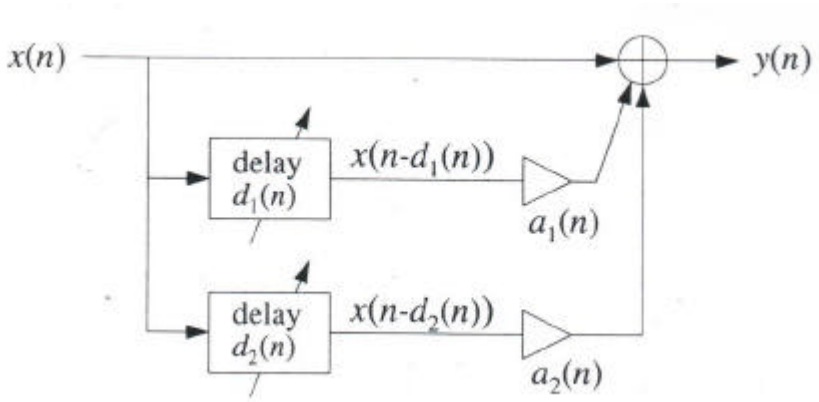


Fig. 2.4.1 Effetto Coro a 3 voci.

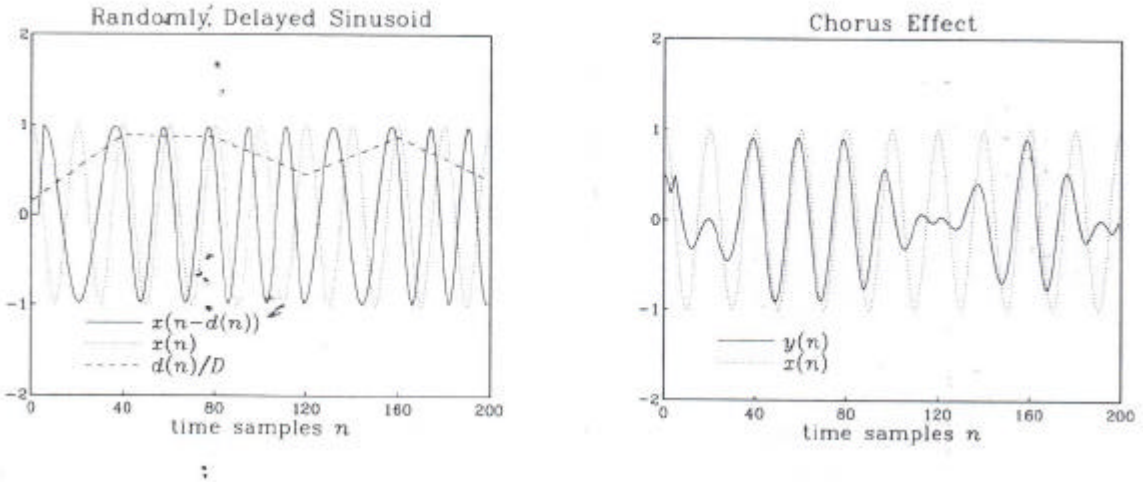


Fig 2.4.2 Come agisce l'effetto coro sui ritardi (a sinistra)
e sul segnale di ingresso (a destra).

Ecco come in AudioFx è stato implementato l'Effetto Coro (a **n** voci):

```
% Effetto Coro – Matlab
periodo=1/freq;          % Periodo del generatore casuale di ritardi
  nperiodi=round(cmp/periodo);       % Numero Periodi in cmp campioni
  for i=1:cmp
    for voc=1:(voci-1)          % voci= numero delle voci nel coro
      d(i,voc)=0;
     a1(i,voc)=0;
    end
  end
% Creo un vettore di ritardi variabili 'd' e amplific variabili 'a1'
% uno per ogni voce nel coro ( esclusa l'originale )
for voc=1:(voci-1)
j=1;
  for n=1:nperiodi
  r(n)=randn(1,1);
  v(n)=r(n)/2;
  % Arrotondamento per creare ritardi interi.
  d2(n)=round(ritardo*(0.5+v(n)));
  % Creo vettore di guadagni variabili ( sempre 0<ai(k)<1 )
  a2(n)=abs(a*(sin(2*pi*(abs(r(n))))));
  for i=1:periodo
    d3(n,i)=d2(n);
    a3(n,i)=a2(n);
    d(j,voc)=d3(n,i);
    a1(j,voc)=a3(n,i);
    j=j+1;
  end
end
end
% Applico il filtro di Coro
% Pongo momentaneamente l'uscita uguale all'ingresso
for i=1:cmp
    y(i)=x(i);
     % aggiunta delle voci
    for voc=1:(voci-1)
        % Creo ritardi circolari
        indice(i,voc)=i-d(i,voc);
          while (indice(i,voc)<=0)
              indice(i,voc)=indice(i,voc)+cmp;
          end
          while (indice(i,voc)>cmp)
              indice(i,voc)=indice(i,voc)-cmp;
          end
```

```
            y(i)=y(i)+(a1(i,voc)*x(indice(i,voc)));
        end
    end
% Riporto il vettore nella forma canonica : 1 colonna, cmp righe.
y=y';
```

Nelle figg. 2.4.3, 2.4.4 e 2.4.5 sono riportate le caratteristiche relative al coro per il canale sinistro di Brano03.wav e BranoPop.wav

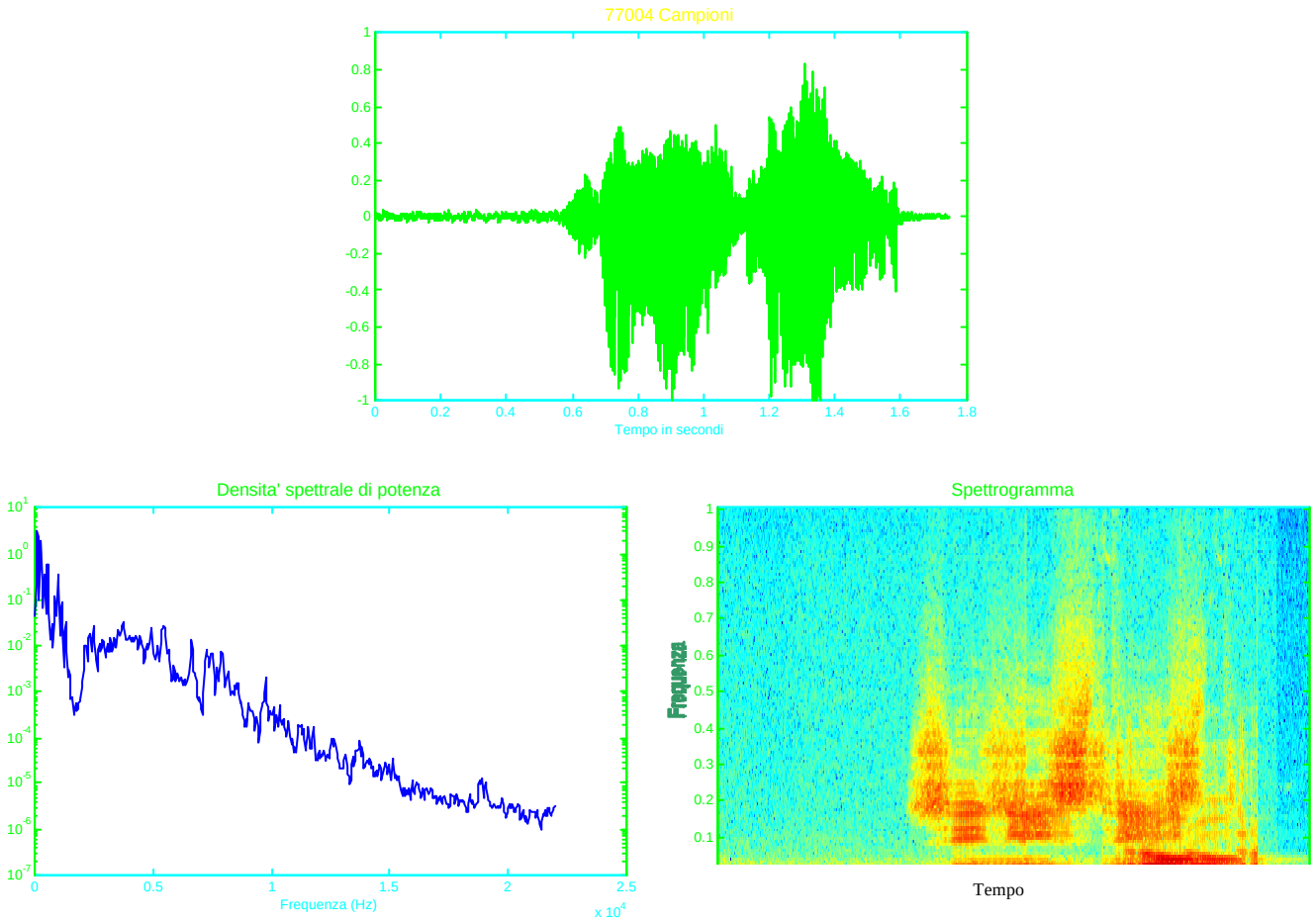
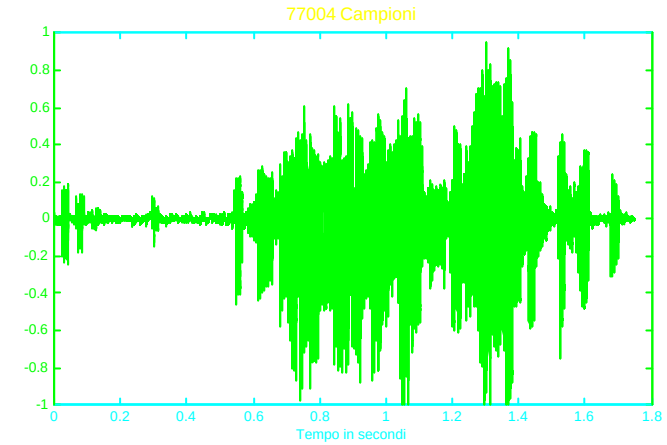
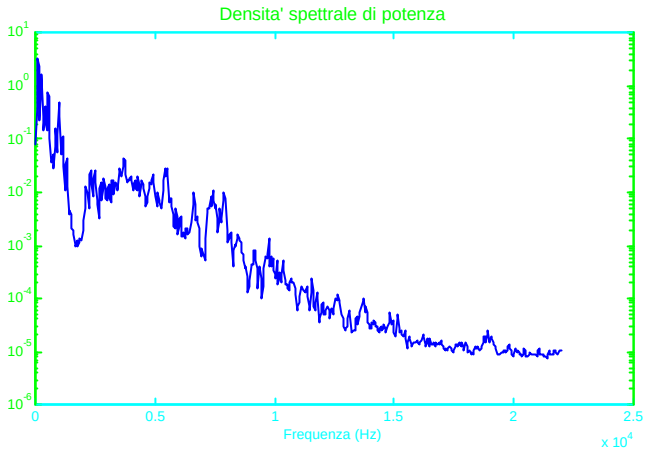
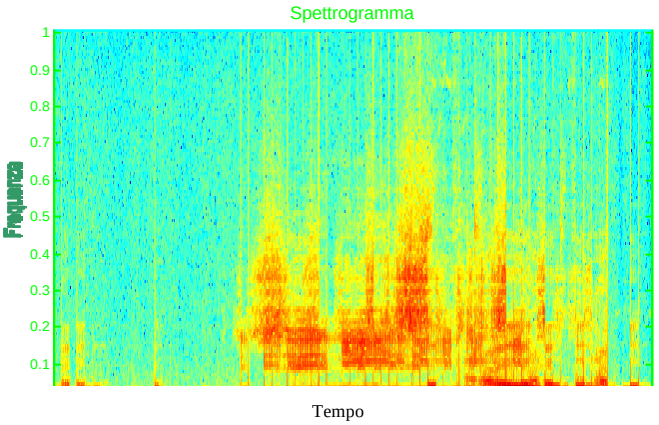


Fig. 2.4.3 Effetto Coro su Brano03.wav con i seguenti parametri:
Ritardo Max **d** = 0,3 secondi
Attenuazione minima **a** = 0,6
Numero **Voci** Coro = 3
Frequenza Desincronizzazione **Freq** = 0,0001 π rad/sec

Come si può osservare in questo e nei grafici seguenti, la frequenza di desincronizzazione, cioè quella che definisce quando variare ritardi e ampiezze nelle varie voci è molto piccola. Più questo valore viene incrementato più il segnale di uscita sarà disturbato ed incomprensibile. Ciò è ovvio perché così facendo le voci del coro divengono troppo diverse. E' come se ognuna di esse ripetesse lo stesso brano ma senza seguire il gruppo. Sul segnale vocale ed in generale sulla voce (del cantante,…) l'effetto coro è particolarmente distinguibile mentre con la musica risulta quasi impercettibile,se lieve, e sgradevole se applicato con numero di voci e frequenza di desincronizzazione elevate.

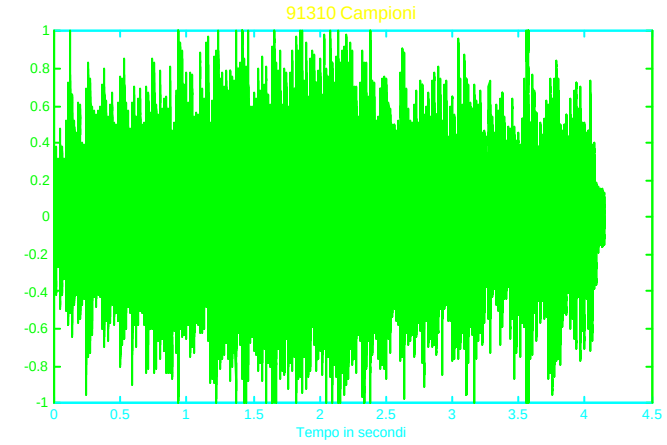


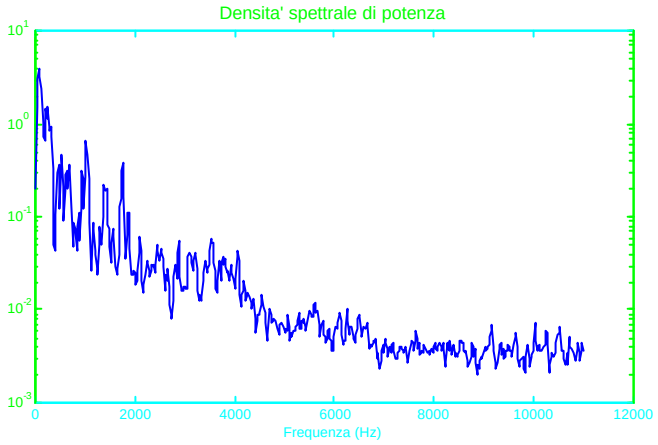


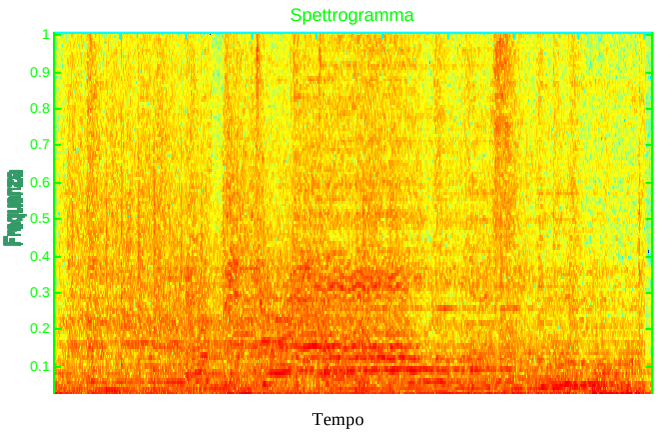


◂ **Fig. 2.4.4** Effetto Coro su Brano03.wav con i seguenti parametri:

Ritardo Max **d** = 0,4 secondi
Attenuazione minima **a** = 0,8
Numero **Voci** Coro = 4
Frequenza Desincronizzazione: **Freq** = 0,001 πrad/sec







◂ **Fig. 2.4.5** Effetto Coro su BranoPop.wav con i seguenti parametri:

Ritardo Max **d** = 0,5 secondi
Attenuazione minima **a** = 0,6
Numero **Voci** Coro = 3
Frequenza Desincronizzazione: **Freq** = 0,0005 πrad/sec

2.5 PHASING O SHIFT DI FASE

Particolarmente noto ai chitarristi, lo SHIFT DI FASE è un effetto speciale che consiste nel far attraversare al segnale audio sorgente un filtro NOTCH molto stretto e combinandone l'uscita con il suono originale, come mostrato nello schema a blocchi di fig. 2.5.1.

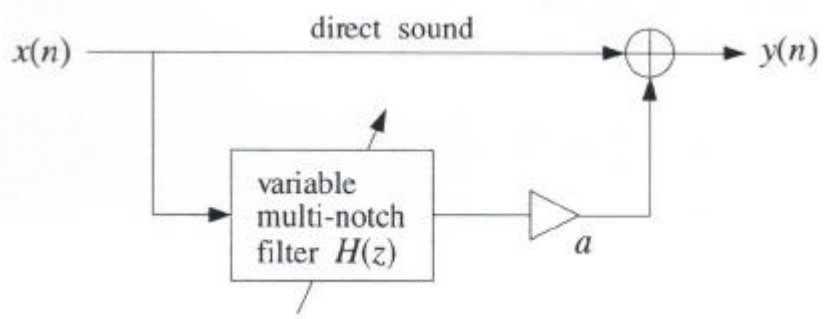


Fig 2.5.1 Schema a blocchi dell'effetto phasing.

Attraverso un'opportuno algoritmo viene fatta variare (ad una frequenza w_{sweep}) la frequenza di notch w_0 all'interno di un intervallo prefissato $w_1 \leq w_0 \leq w_2$ cosi' da ottenere un risultato simile a quanto riportato in figura 2.5.2.

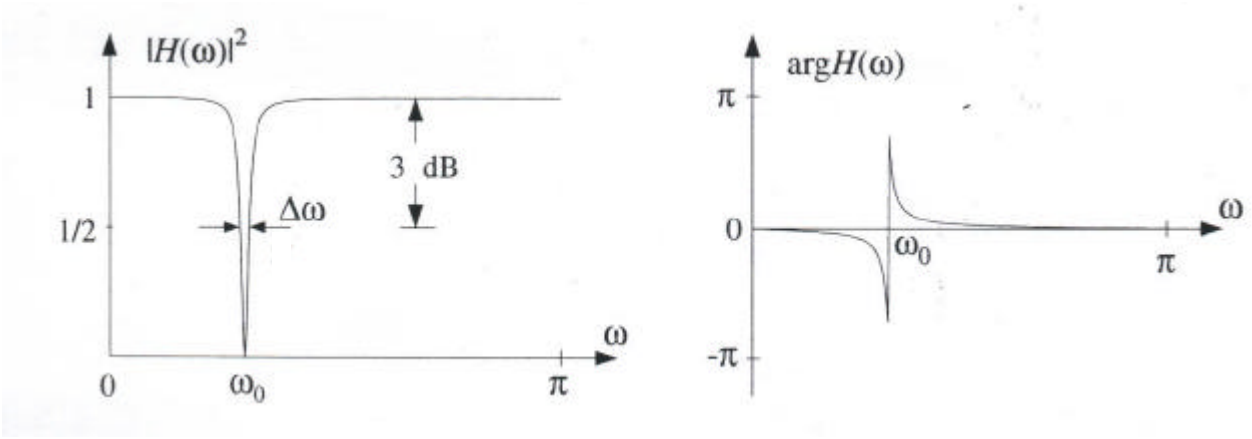


Fig. 2.5.2 Il filtro di Notch molto stretto causa forti shift di fase attorno alla frequenza di notch.

Si nota che la risposta di fase rimane essenzialmente nulla tranne nelle vicinanze della frequenza di notch dove si hanno rapide variazioni. Un possibile sistema che implementi tale effetto è rappresentato dalla funzione di trasferimento:

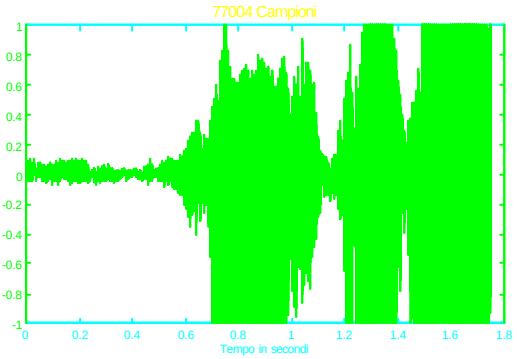
$$H(z) = b \cdot \frac{1 - 2\cos(w_0) \cdot z^{-1} + z^{-2}}{1 - 2b \cdot \cos(w_0) \cdot z^{-1} + (2b - 1) \cdot z^{-2}}$$

Con:

$$b = \frac{1}{1 + tg(\frac{dw}{2})} \quad , \quad dw = \frac{w_0}{Q} \quad , \quad \text{Fattore Q} = \left|H_{LP}\right|_{db} = -40 \cdot \text{Log}_{10}(\frac{w}{w_0})$$

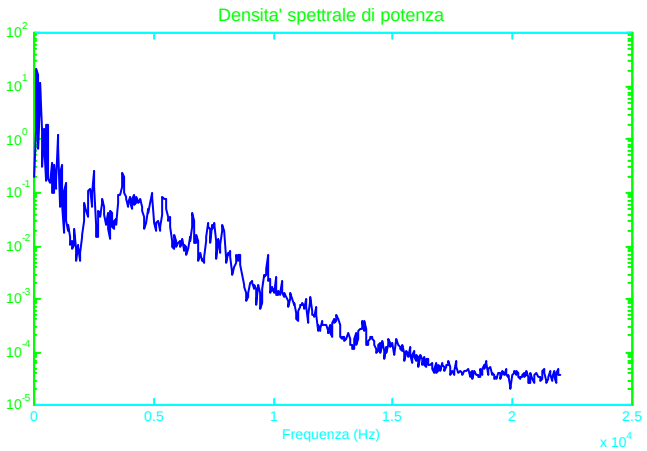
Più grande è Q, più stretto è il filtro notch. Di seguito viene mostrato un'esempio di codice Matlab per creare l'effetto Phasing:

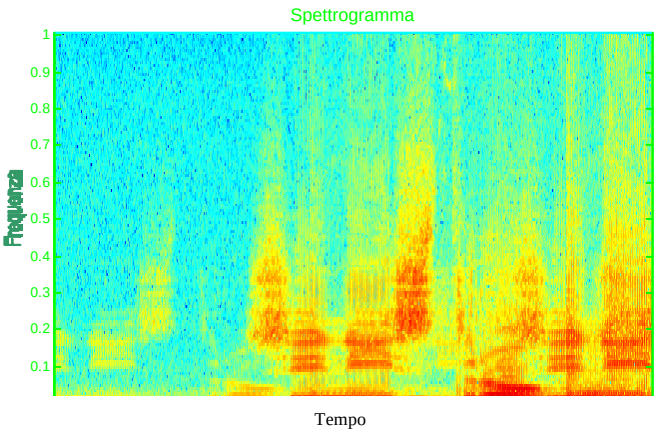
```
% Effetto Phasing - Matlab
for i=1:cmp
        w0(i)=w1+(w2*sin(wsweep*1));
        Deltaw=(w0(i)/Q);
        b=(1/(1+tan(Deltaw/2)));
         % Canale  + ritardi circolari
        indice1=i-(1*FS);
                while (indice1<=0)
                    indice1=cmp+indice1;
                end
         indice2=i-(2*FS);
                while (indice2<=0)
                    indice2=cmp+indice2;
                end
        W01(i)=b*x(i)+(2*b*(x(indice1))*cos(w0(i)))-(((2*b)-1)*(x(indice2)));
        y(i)=x(i)+(a*(W01(i)-(2*(x(indice1))*cos(w0(i)))+(x(indice2))));
          x(indice2)=x(indice1);
        x(indice1)=W01(i);
end
% Riporto il vettore nella forma canonica : 1 colonna, cmp righe.
y=y';
```

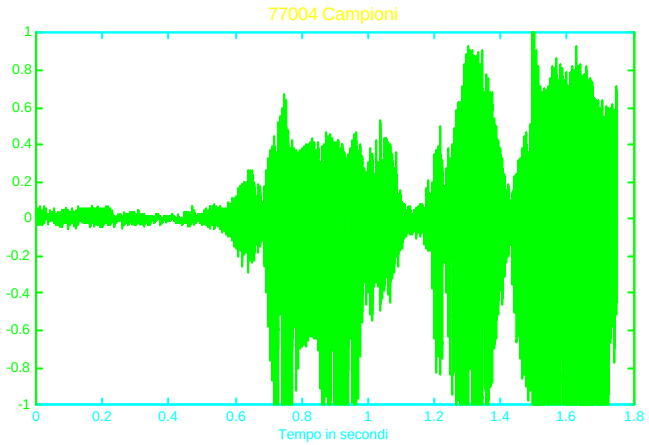


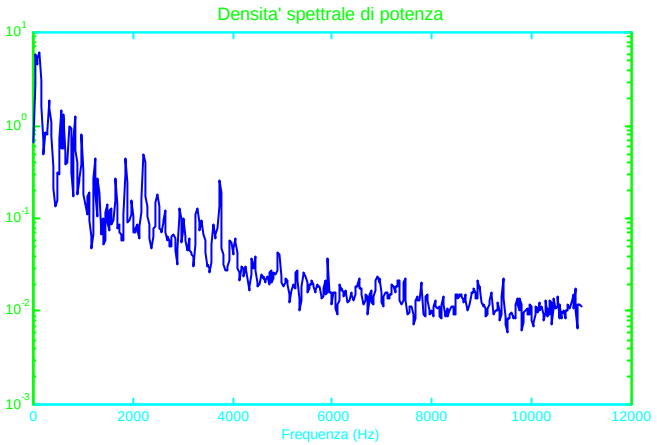
◂ **Fig. 2.5.3** Effetto Phasing su Brano03.wav con i seguenti parametri:

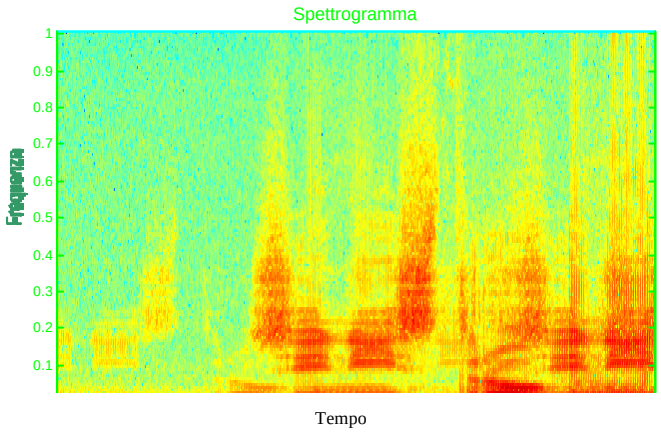
Attenuazione minima **a** = 0,8
w1 = 0,11 πrad/sec
w2 = 0,75 πrad/sec
$\mathbf{w_{sweep}}$ = 0,85 πrad/sec
Q = 3,5





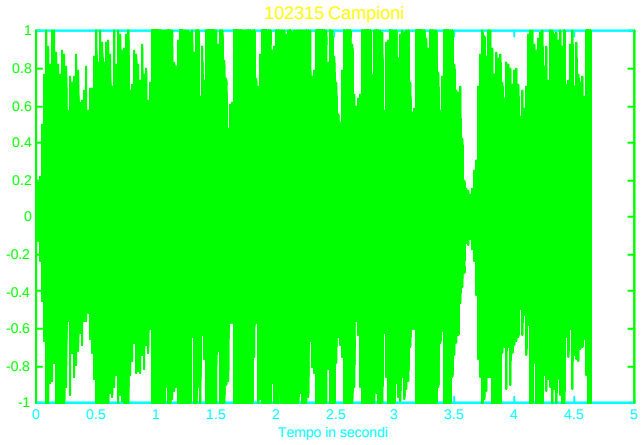


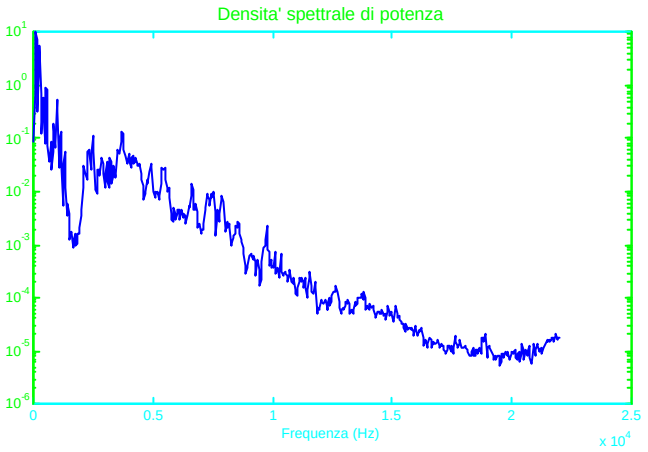


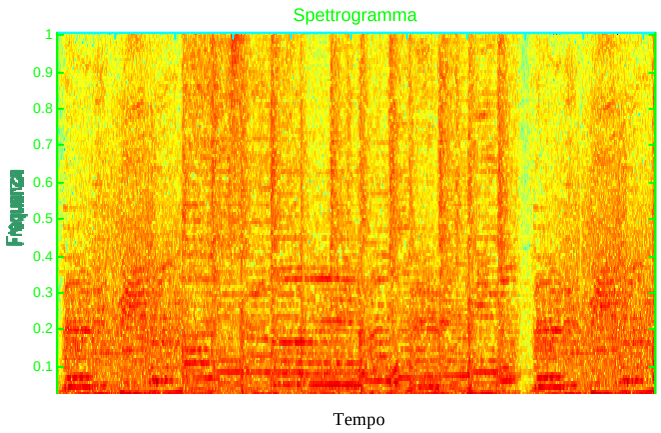


◂ **Fig. 2.5.4** Effetto Phasing su Brano03.wav con i seguenti parametri:

Attenuazione minima **a** = 0,3
w1 = 0,5 πrad/sec
w2 = 0,9 πrad/sec
$\mathbf{w_{sweep}}$ = 0,35 πrad/sec
Q = 2







◂ **Fig. 2.5.5** Effetto Phasing su BranoDan.wav con i seguenti parametri:

Attenuazione minima **a** = 0,8
w1 = 0,46 πrad/sec
w2 = 0,81 πrad/sec
$\mathbf{w_{sweep}}$ = 0,003 πrad/sec
Q = 2,5

2.6 RIVERBERI

Esistono numerose tipologie di effetti speciali che vengono raggruppate nella classe dei riverberi. Abbiamo già precedentemente affrontato l'analisi del **Riverbero Piano** o Eco Multiplo, per la cui trattazione si rimanda al paragrafo 2.2. Qui non verrà più ripreso per dedicarsi allo studio di due effetti classici:

- Il **RIVERBERO PASSATUTTO** (o di **SCHROEDER**)
- Il **RIVERBERO PASSABASSO**

Innanzitutto è bene sottolineare che il Riverbero è tipicamente caratterizzato da tre componenti distinte: il **suono diretto** (quello che arriva all'ascoltatore direttamente dalla sorgente), le **riflessioni dirette** (early) e quelle **indirette** (late), come mostrato in figura 2.6.1. La qualità di una sala da concerto dipende proprio dalla risposta all'impulso dell'effetto riverbero che essa crea.

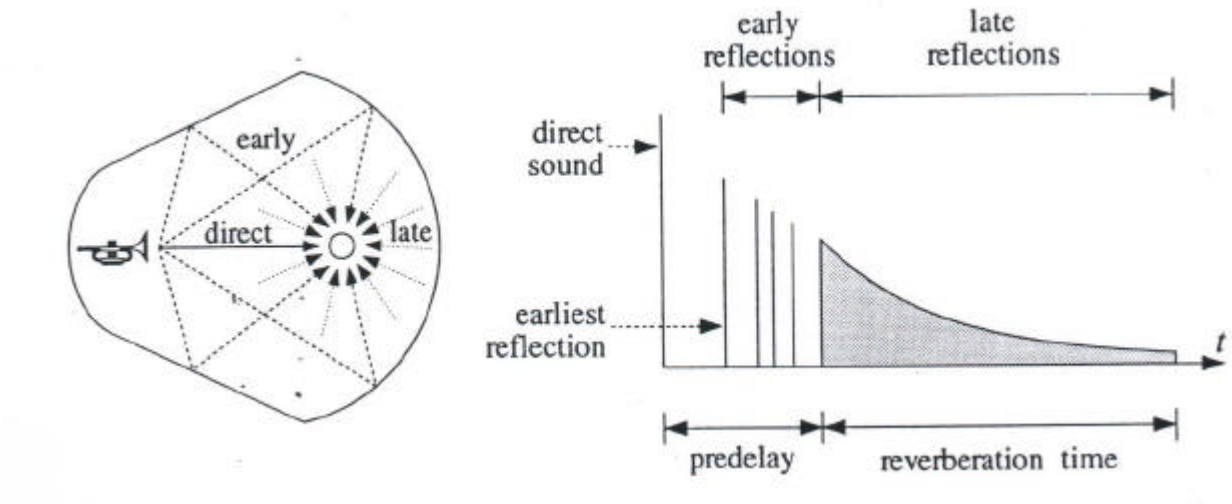


Fig. 2.6.1 Effetto delle tre componenti del riverbero in una sala da ascolto.

Il modello **PASSATUTTO di SCHROEDER** ha una risposta in ampiezza costante per tutte le frequenze. Il filtro è descritto dalla funzione di trasferimento:

$$H(z) = \frac{-a + z^{-D}}{1 + a \cdot z^{-D}}$$

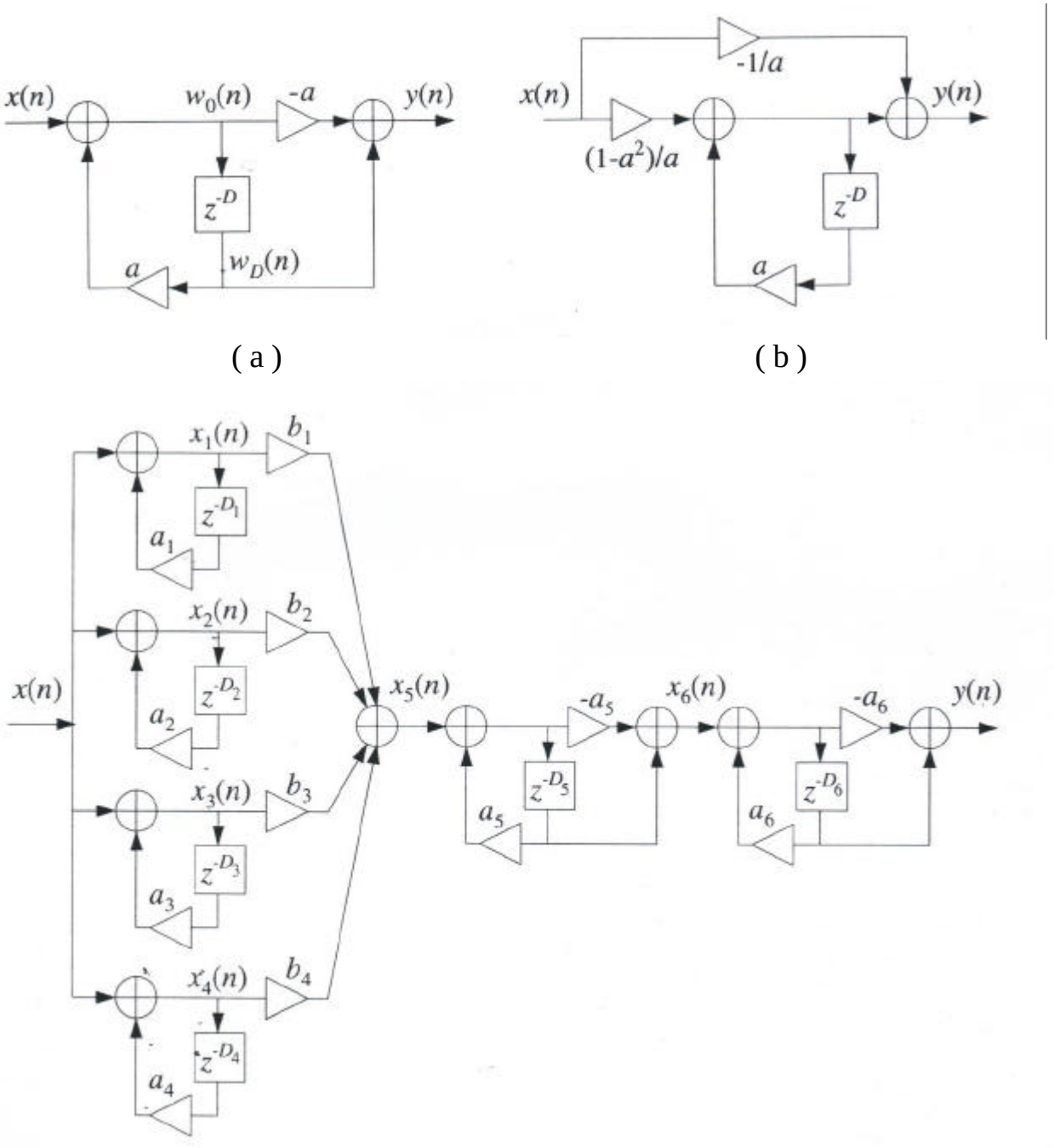
Con **D** ritardo massimo in campioni ed **a** coeff. di attenuazione minima. Da notare la tipica forma del filtro passatutto o **SFASATORE PURO** con un polo ed uno zero in posizione reciproca ad esso. Espresso in termini di equazione alle differenze:

$$y(n) = a \cdot y(n - D) - a \cdot x(n) + x(n - D)$$

Ponendo $z=e^{jw}$ ottengo la risposta in frequenza:

$$\mathbf{H(w) = \frac{-a + e^{-jwD}}{1 - a \cdot e^{-jwD}} \quad \mapsto \quad |H(w)| = 1 \ , \ per\ ogni\ w}$$

In fig. 2.6.2 vengono mostrati gli schemi a blocchi di possibili configurazioni passatutto, mentre in fig 2.6.3 e 2.6.4 i risultati dell'effetto in corrispondenza dei rispettivi parametri.



(a) (b)

(c)

Fig. 2.6.2 Esempi di Riverberatori Passatutto:
a – Forma Canonica
b – Forma Parallela
c – Configurazione Schroeder a 6 elementi

In AudioFx l'effetto Riverbero Passatutto viene realizzato attraverso il codice seguente:

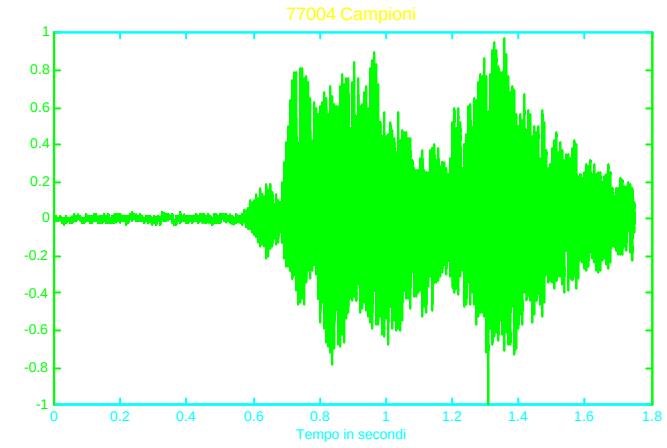
```
% Effetto Riverbero Passatutto – Matlab
% Azzero coefficienti num e den del filtro H(z)
  for j=1:(D+1)
    b(j)=0;
    c(j)=0;
  end
  b(1)=-atten;
  b(D+1)=1;
  c(1)=1;
  c(D+1)=-atten;
% Applico filtraggio
y=filter(b,c,x);
```

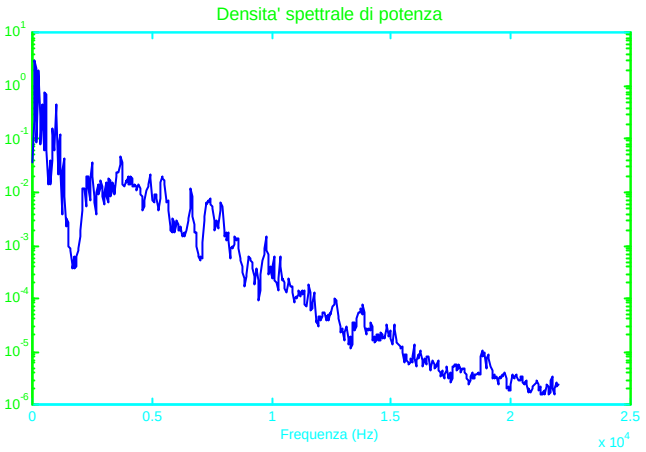
Può essere utile ricordare che la funzione Matlab **y=filter(b,a,x)** filtra i dati contenuti nel vettore **x** con il filtro descritto dai vettori **a** e **b**, creando il vettore uscita **y**. Il filtro è descritto dalla equazione alle differenze seguente:

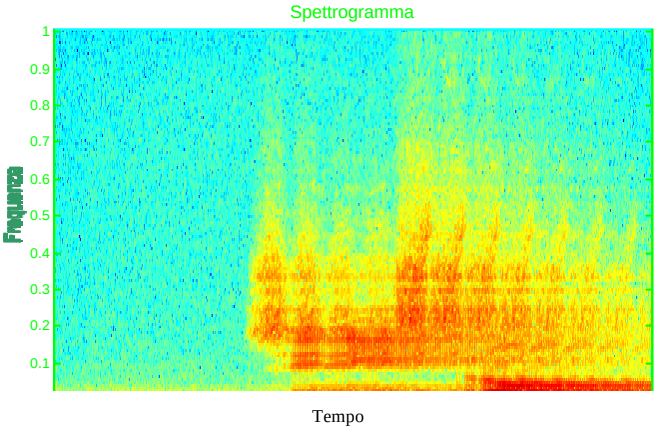
$$y(n)=b(1)x(n)+b(2)x(n-1)+\ldots+b(nb)x(n-nb+1)-a(2)y(n-1)-\ldots-a(na)y(n-na+1)$$

oppure tramite la sua trasformata Z:

$$H(z)=\frac{Y(z)}{X(z)}=\frac{b(1)+b(2)\cdot z^{-1}+\ldots+b(nb)\cdot z^{-(nb-1)}}{1+a(2)\cdot z^{-1}+\ldots+a(na)\cdot z^{-(na-1)}}$$

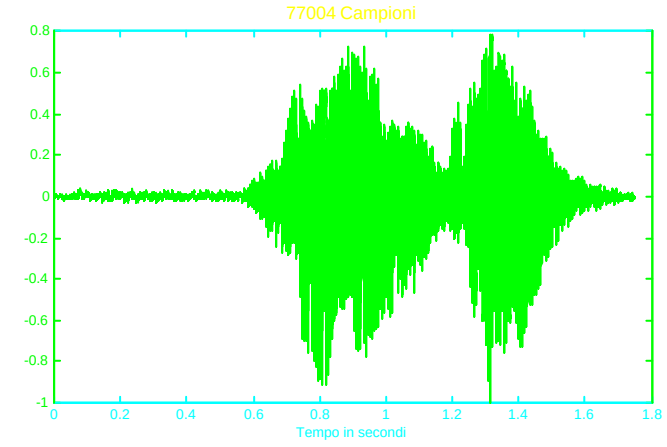


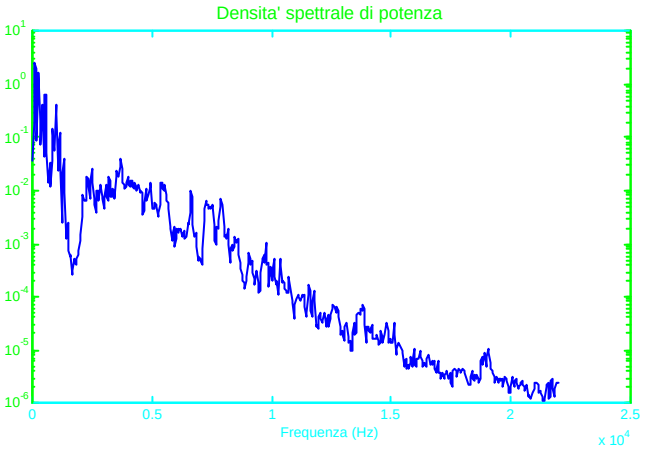


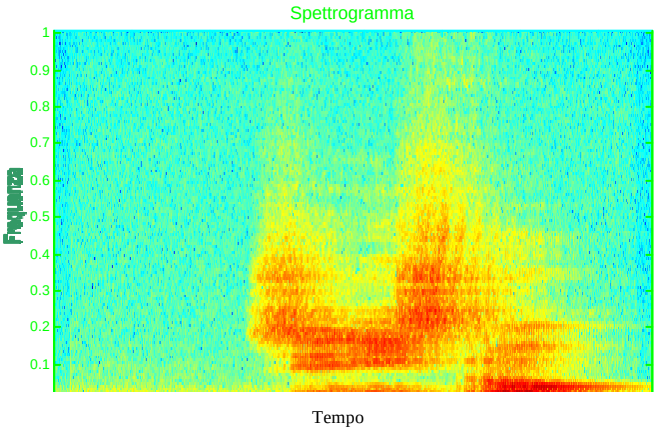


◂ **Fig. 2.6.3** Effetto Riverbero di Schroeder su Brano03.wav con i seguenti parametri:

Attenuazione minima **a** = 0,7
Ritardo Max **d** = 0,1 secondi

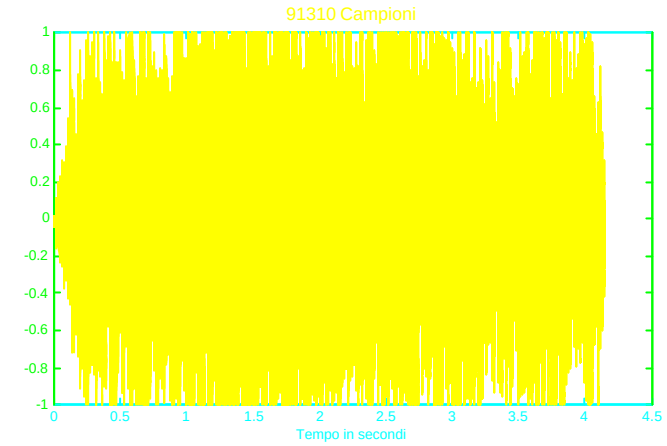


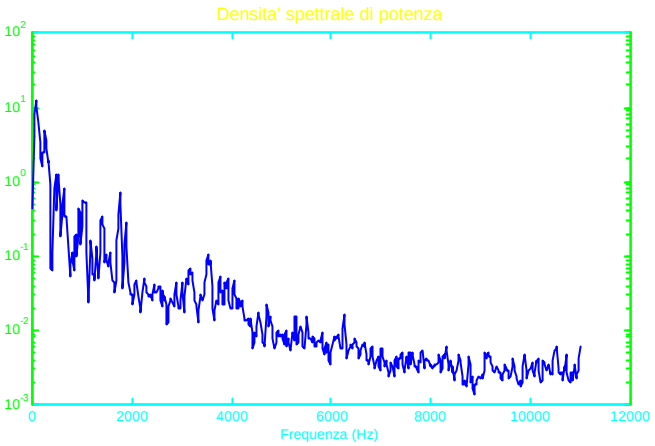


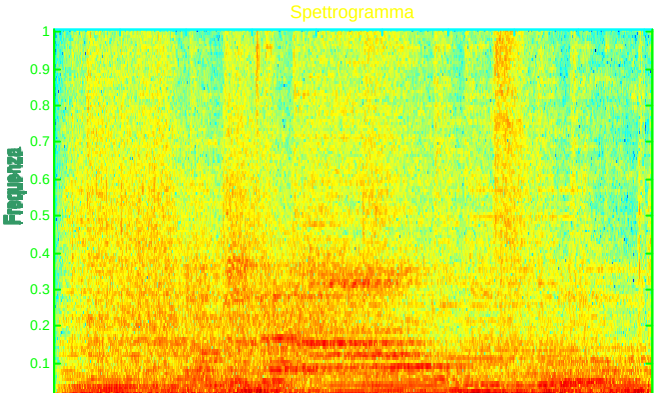


◂ **Fig. 2.6.4** Effetto Riverbero di Schroeder su Brano03.wav con i seguenti parametri:

Attenuazione minima **a** = 0,5
Ritardo Max **d** = 0,05 secondi

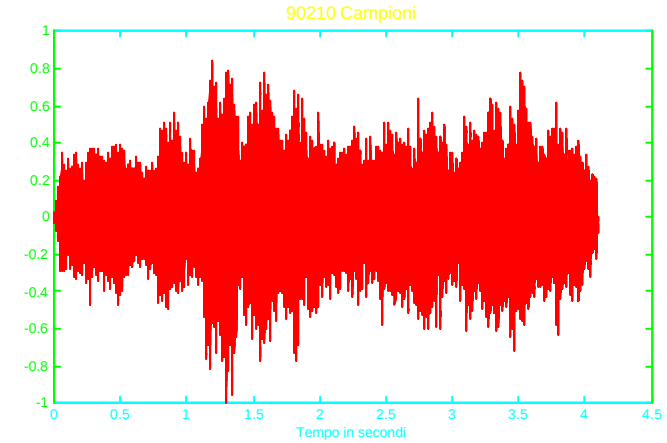


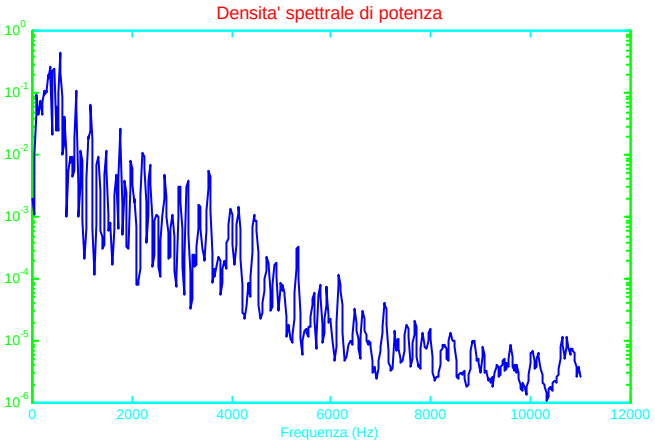


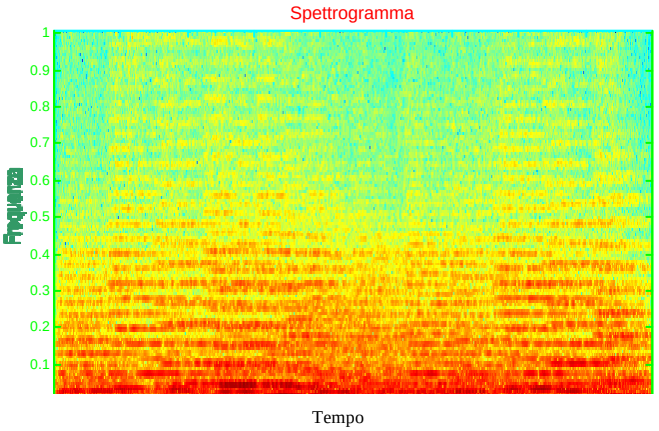


◂ **Fig. 2.6.5** Effetto Riverbero di Schroeder su BranoPop.wav con i seguenti parametri:

Attenuazione minima **a** = 0,7
Ritardo Max **d** = 0,6 secondi







◂ **Fig. 2.6.6** Effetto Riverbero di Schroeder su BranoCla.wav con i seguenti parametri:

Attenuazione minima **a** = 0,5
Ritardo Max **d** = 0,8 secondi

Una variazione più complessa è costituita da **RIVERBERO PASSABASSO**. Come mostrato in fig. 2.6.7 il coefficiente di attenuazione **a** , costante nel riverberatore piano, viene sostituito con un filtro passabasso G(z) tale che:

$$G(z) = \frac{b_0 + b_1 \cdot z^{-1} + \ldots + b_M \cdot z^{-M}}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2} + \ldots + a_M \cdot z^{-M}}$$

G(z) può essere sia di tipo FIR che di tipo IIR, a seconda del grado M e del valore dei coefficienti b_i e a_i . Con G(z) si ricava la funzione di trasferimento del filtro H(z):

$$H(z) = \frac{1}{1 - G(z) \cdot z^{-D}}$$

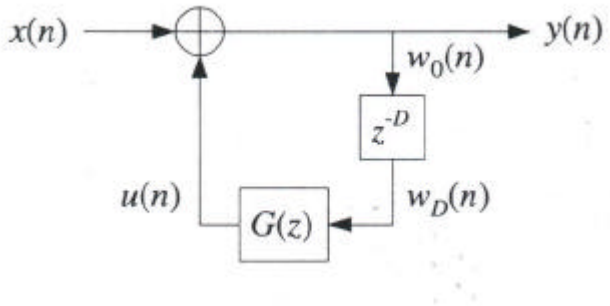


Fig 2.6.7 Schema a blocchi del Riverberatore Passabasso

Di seguito vengono riportati il codice Matlab per lo sviluppo dell'effetto riverbero passabasso e, nelle figg. 2.6.8, 2.6.9 e 2.6.11 risultati significativi elaborando i files Brano03.wav e BranoDan.wav. In fig 2.6.10 un confronto tra i tre tipi di riverbero: piano, passatutto, passabasso.

```
% Effetto Riverbero Passabasso – Matlab
% Azzero coefficienti non necessari di num e den del filtro G(z) passabasso
  for j=(M+2):(D+1)
     b(j)=0;
   end
   for p=(M+1):(D+1)
     c(p)=0;
   end
y=x;   % Pongo momentaneamente l'uscita uguale all'ingresso.
% Applico Filtro
     for i=1:cmp
        % Ritardo circolare
        indice=i-D;
          while (indice<=0)
             indice=cmp+indice;
           end
 y(i)=filter(b,c,y(indice))+x(i);
end
```

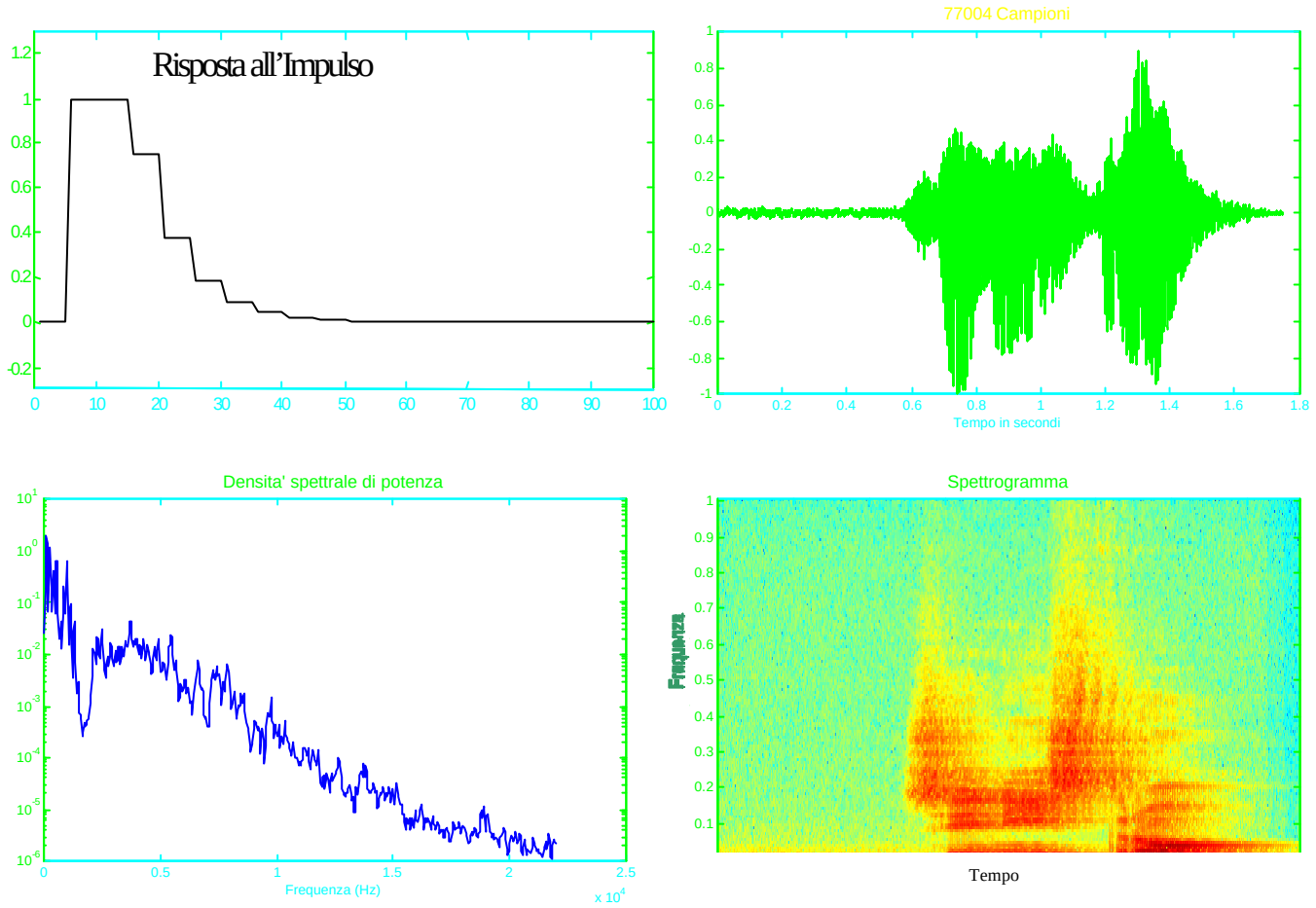
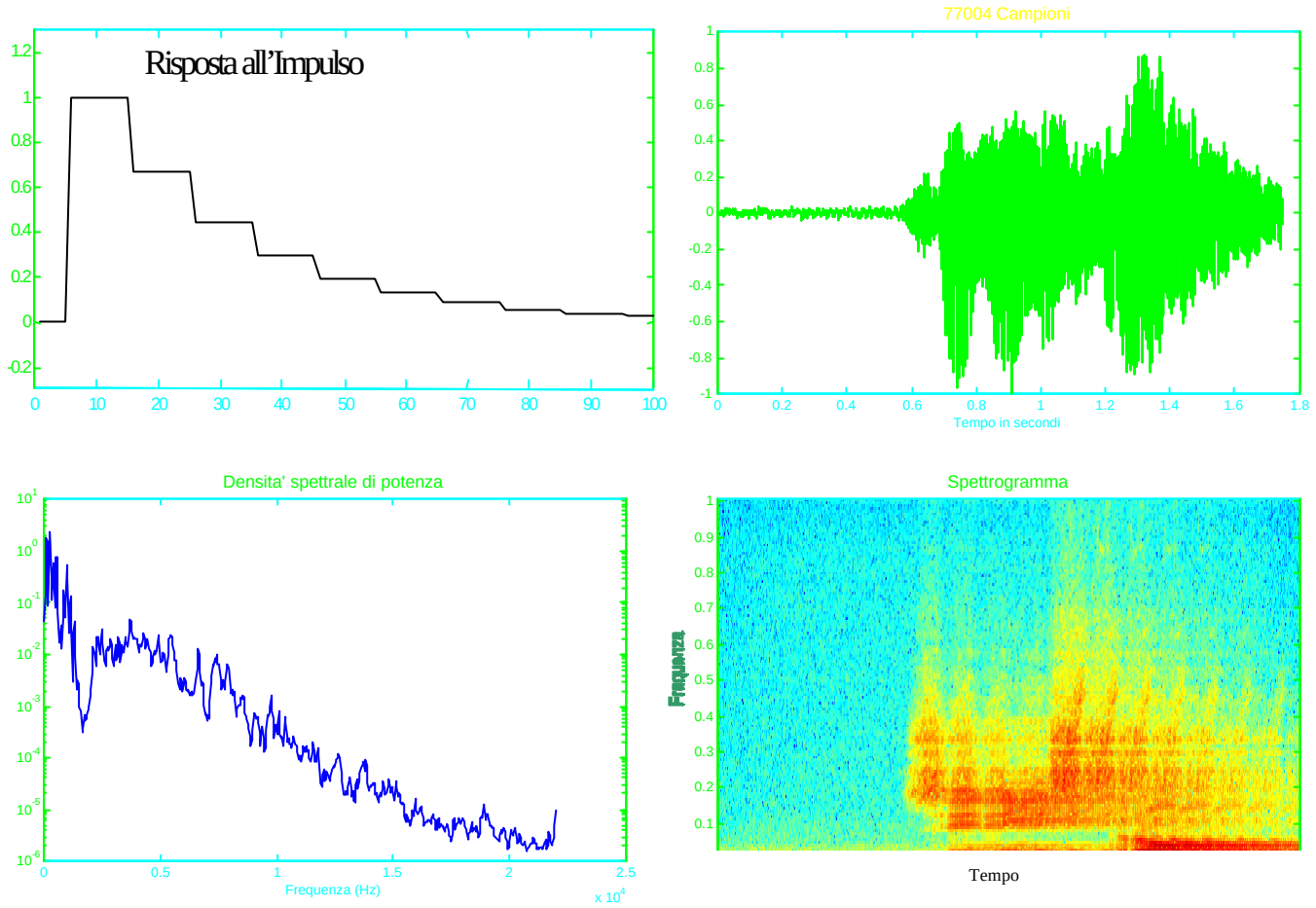


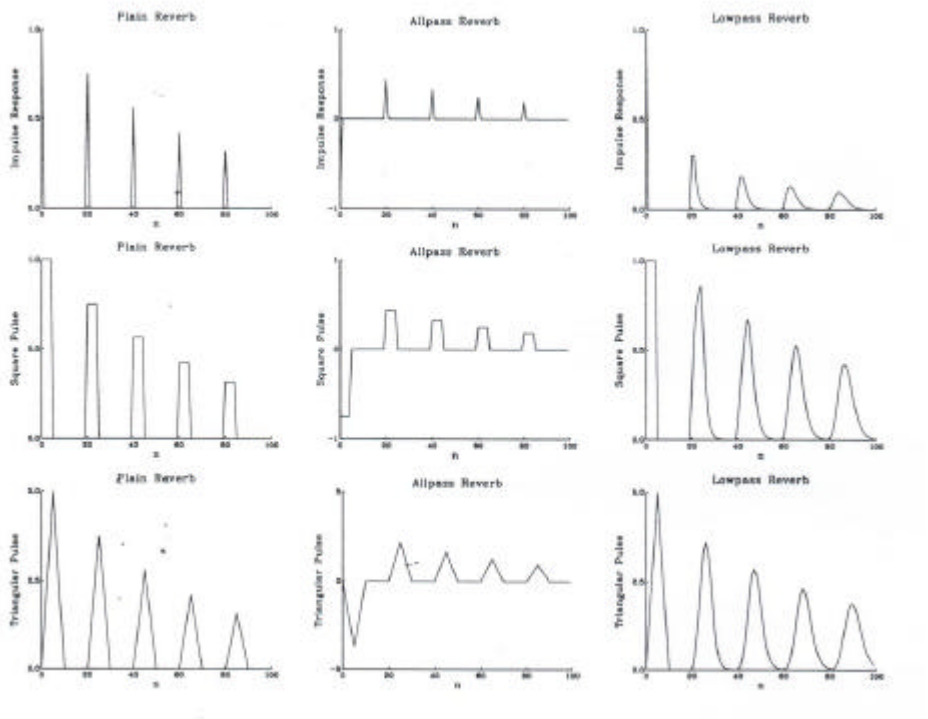
Fig. 2.6.8 Effetto Riverbero Passabasso su Brano03.wav
in corrispondenza dei seguenti parametri:

Ritardo Max **d** = 0,05 secondi
Ordine **M** del filtro Passabasso di Feedback = 2
Coefficienti numeratore filtro di feedback $\mathbf{b_i}$ = 1, 2, 1
Coefficienti denominatore filtro di feedback $\mathbf{c_i}$ = 2, 2

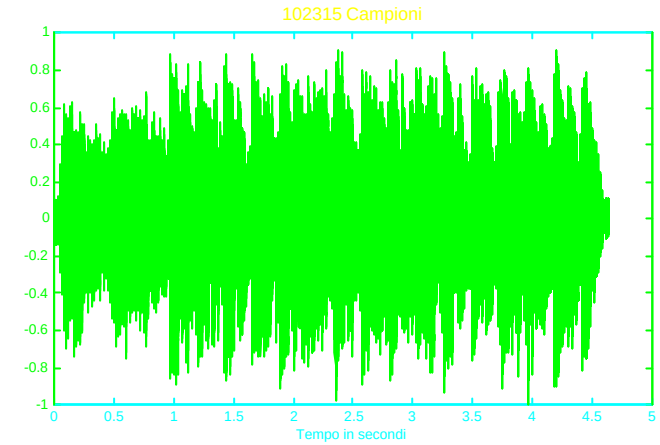
Un aspetto che va sottolineato e che non è stato ancora affrontato riguarda la scelta dei coefficienti di numeratore e denominatore del filtro passabasso G(z). Essi infatti definiscono i poli e gli zeri, rispettivamente, del filtro stesso. Valori troppo bassi per $\mathbf{c_i}$ e troppo alti per $\mathbf{b_i}$ portano ad avere poli fuori dal cerchio unitario conducendo il filtro all' instabilità. AudioFx non si occupa di rendere stabile il filtro distorcendo così il segnale originale e fornendo in uscita valori troncati, perché superiori o inferiori al range consentito (vedi Fig. 2.6.12). Da notare la differenza nella densità spettrale con la fig. 2.6.11.

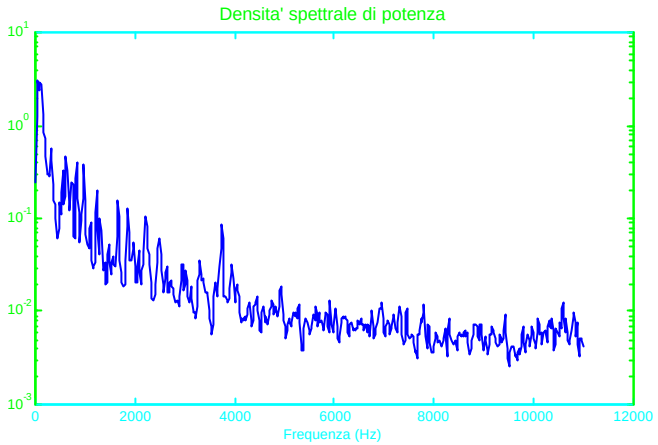


▲ **Fig 2.6.9** Effetto Riverbero Passabasso in corrispondenza dei seguenti parametri:
Ritardo Max **d** = 0,1 secondi
Ordine **M** del filtro Passabasso di Feedback = 1
Coefficienti numeratore filtro di feedback $\mathbf{b_i}$ = 2, 2
Coefficienti denominatore filtro di Feedback $\mathbf{c_i}$ = 3



◄ **Fig. 2.6.10** Confronto tra i vari tipi di riverbero: piano, passatutto e passabasso.





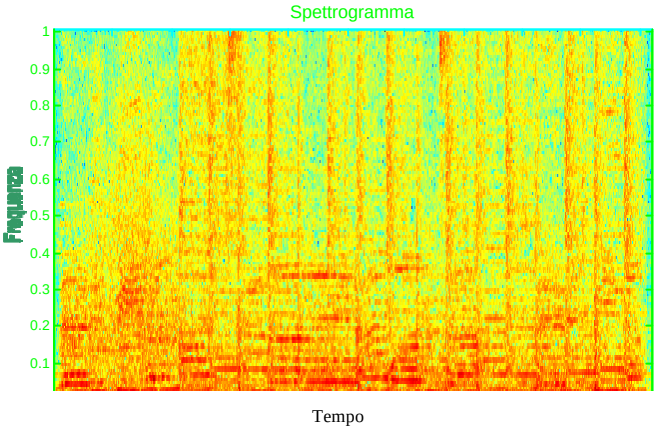
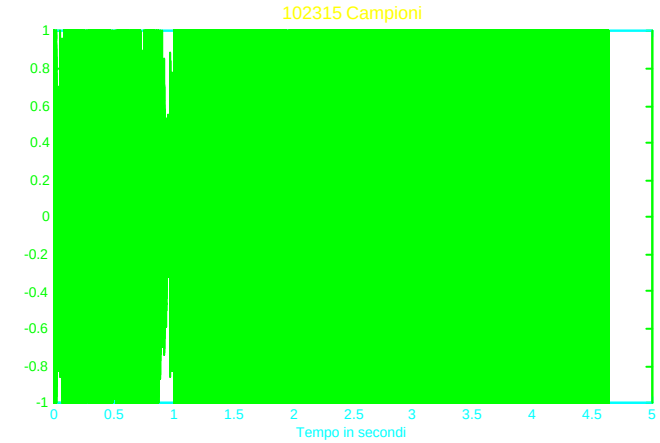
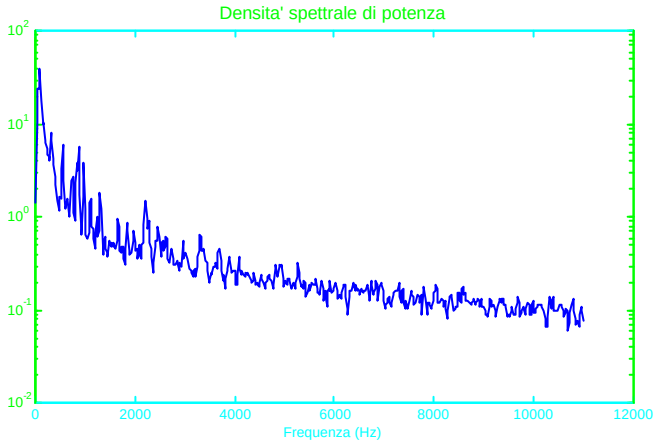


Fig. 2.6.11 Effetto Riverbero Passabasso su BranoDan.wav con i seguenti parametri:

Ritardo Max **d** = 1 secondo
Ordine **M** filtro G(z) = 2
Coeffic. Numeratore G(z), $\mathbf{b_i}$ = 1, 1, 1
Coeffic. Denominatore G(z), $\mathbf{c_i}$ = 6, 6





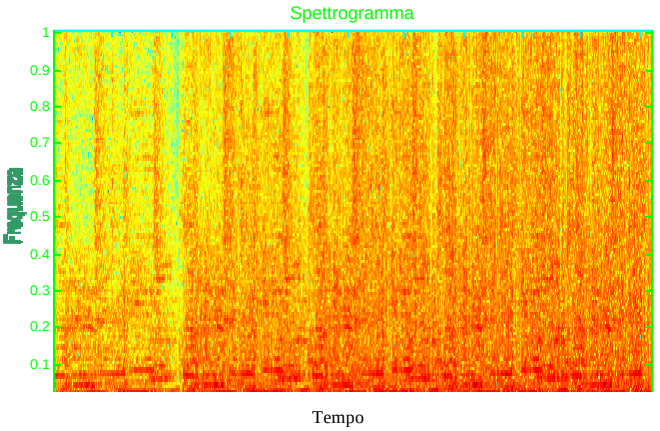


Fig. 2.6.12 Effetto Riverbero Passabasso su BranoDan.wav con i seguenti parametri:

Ritardo Max **d** = 1 secondo
Ordine **M** filtro G(z) = 1
Coeffic. Numeratore G(z), $\mathbf{b_i}$ = 5, 5
Coeffic. Denominatore G(z), $\mathbf{c_i}$ = 2

2.7 COMPRESSORI / ESTENSORI

Questa classe di effetti speciali agisce sulla dinamica del segnale audio. I compressori, per esempio, attenuano in ampiezza se questa supera una certa soglia, viceversa, per gli estensori. La relazione ingresso/uscita per un simile sistema è la seguente:

$$y = y_0 \cdot \left(\frac{x}{x_0}\right)^r$$

Dove **x** è l'ingresso, $\mathbf{x_0}$ è la soglia desiderata, **r** (indicato anche con **ρ**) è il fattore di compressione/estensione. E' importante notare che si tratta di un filtro **NON LINEARE** . Infatti se $x(n)=x_1(n)+x_2(n)$:

$$y[x(n)] = y_0 \cdot \left(\frac{x_1(n)+x_2(n)}{x_0}\right)^r \text{ che è diverso da } y_1(n)+y_2(n)$$

Vale la pena ricordare che in tutti i filtri proposti in questa trattazione si sono considerati sistemi causali, questo per la natura audio dei segnali elaborati.

Per un compressore $x \geqq x_0$, $r<1$ mentre per un estensore $x \leq x_0$, $r>1$ come illustrato in fig 2.7.1 dove il fattore **r** è indicato con **ρ**.

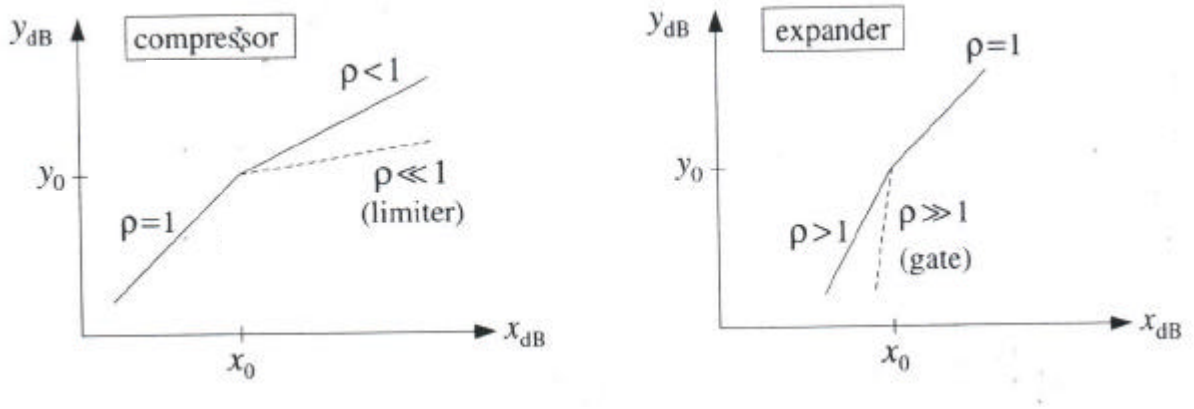


Fig. 2.7.1 Relazione ingresso/uscita di compressore/estensore.

Uno schema a blocchi per un simile filtro è mostrato in figura 2.7.2. Il **LEVEL DETECTOR** genera un segnale di controllo c(n) a partire dall'ingresso x(n). Tale segnale serve per determinare il guadagno G(n). Il Level Detector può funzionare in infiniti modi, per esempio secondo la seguente legge:

$$\mathbf{c(n) = \lambda \cdot c(n-1) + (1-\lambda) \cdot |x(n)|}$$

dove λ (con $0 \leq \lambda \leq 1$) è chiamato **FATTORE DI PERDITA**

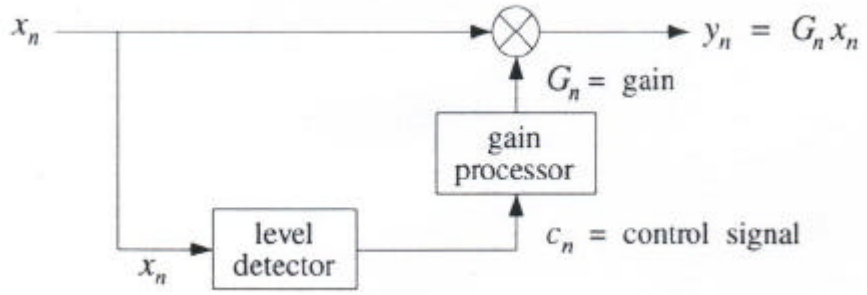


Fig. 2.7.2 Schema a blocchi del Compressore/Estensore

Così **G(n)** $\begin{cases} = (c/c_0)^{r-1} & \text{se } c \geqq c_0 \\ = 1 & \text{se } c < c_0 \end{cases}$ r<1 se il sistema è un compressore

Mentre **G(n)** $\begin{cases} = (c/c_0)^{r-1} & \text{se } c < c_0 \\ = 1 & \text{se } c \geqq c_0 \end{cases}$ r>1 se il sistema è un estensore

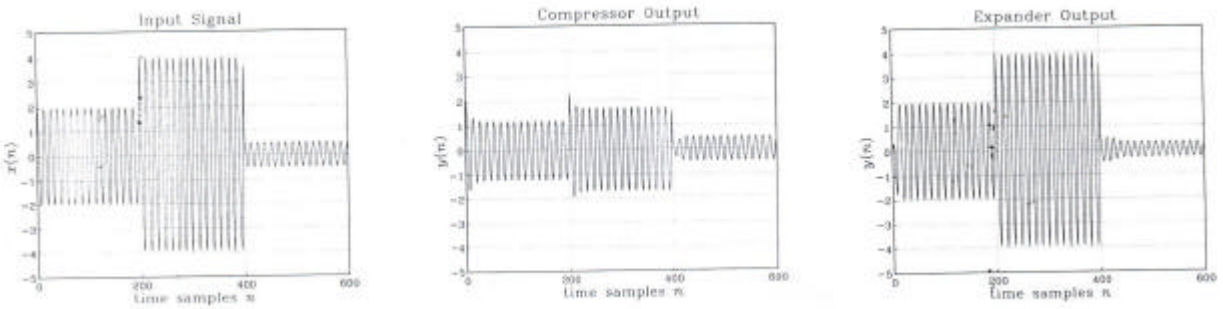


Fig. 2.7.3 Azione del Compressore/Estensore sul segnale di ingresso.

% Effetto Compressione – Matlab

```
c(1)=0;
for i=(2):cmp
      c(i)=(h*c(i-1))+((1-h)*abs(y1(i)));
      if c(i)>=c0
        G(i)=((c(i)/c0)^(r-1));
      else
        G(i)=1;
      end
      y(i)=G(i)*x(i);
end
y=y';
```

```
% Effetto Estensione – Matlab
c(1)=0;
for i=(2):cmp
        c(i)=(h*c(i-1))+((1-h)*abs(y1(i)));
        if c(i)<=c0
          G(i)=((c(i)/c0)^(r-1));
        else
          G(i)=1;
        end
        y(i)=G(i)*x(i);
end
y=y';
```

Nelle pagine successive esempi applicativi di compressione (figg. 2.7.4, 2.7.5 e 2.7.6) e di estensione (figg. 2.7.7, 2.7.8 e 2.7.9).

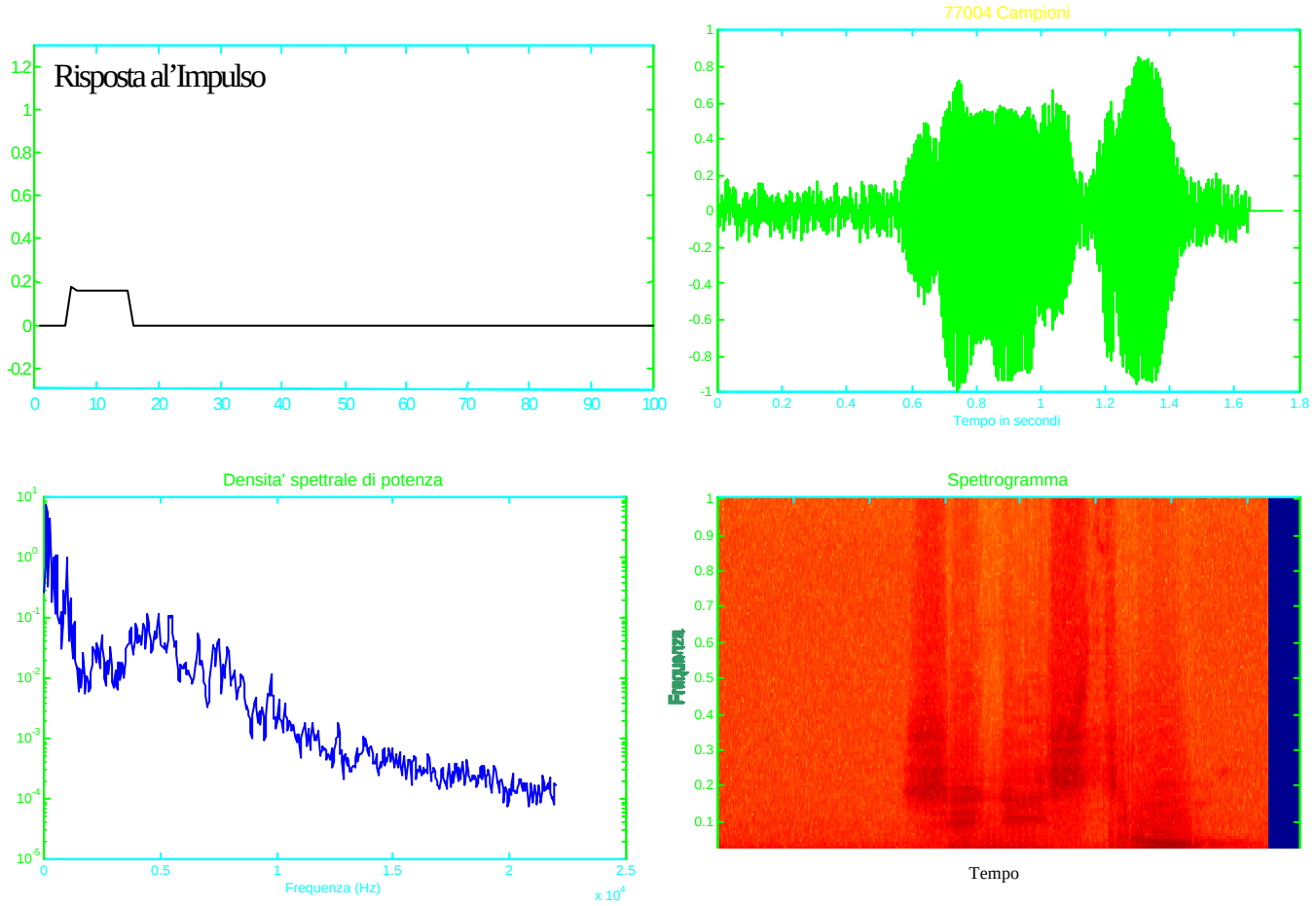


Fig. 2.7.4 Effetto Compressione su Brano03.wav con i seguenti parametri:
Parametro Iniziale $\mathbf{C_0}$ controllo guadagno = 0,025
Fattore di Perdita $\boldsymbol{\lambda}$ = 0,5
Fattore di Compressione **r** = 0,5

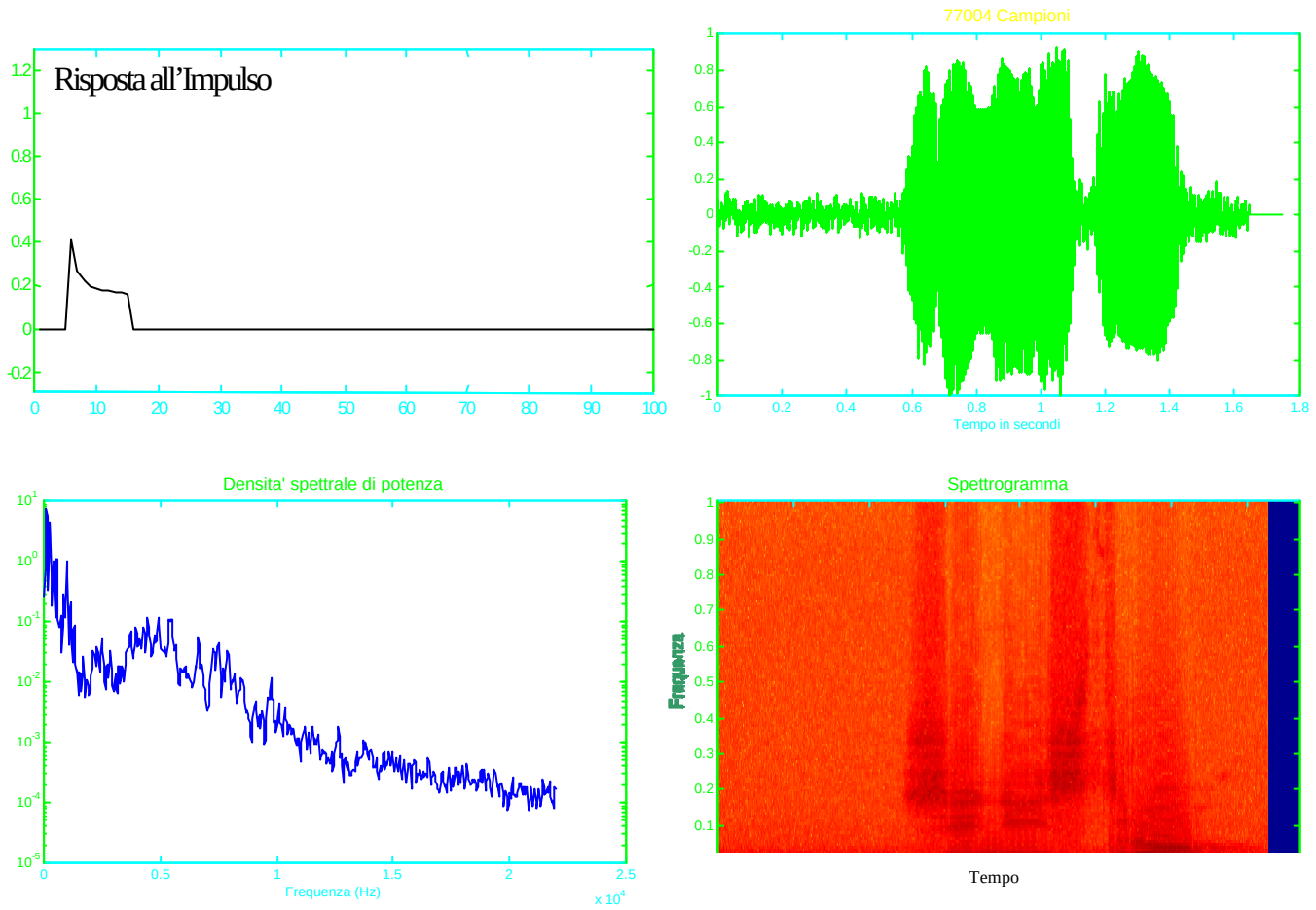
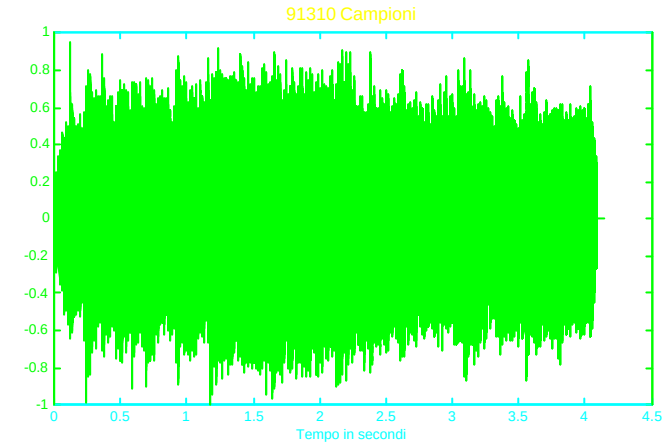
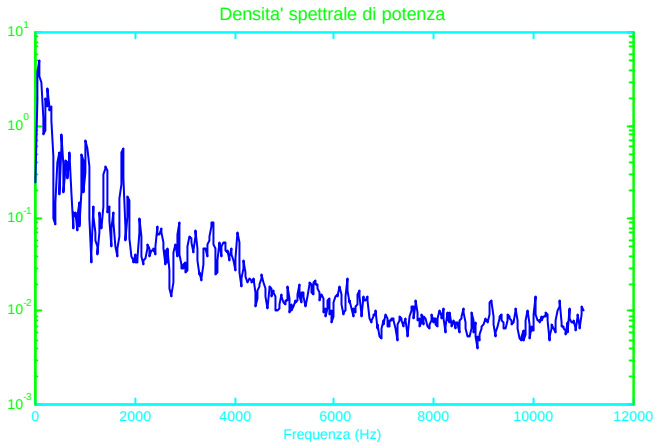


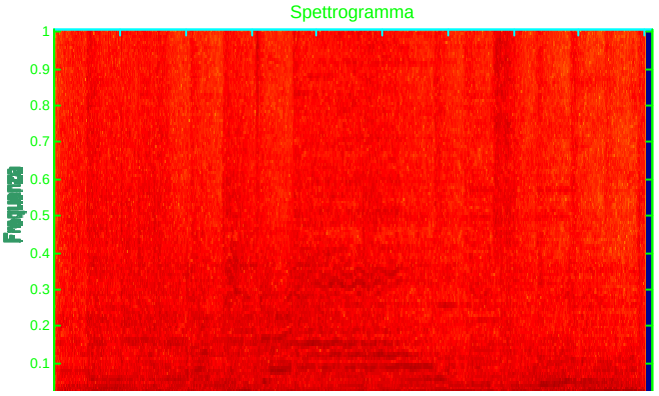
Fig. 2.7.5 Effetto Compressione su Brano03.wav con i seguenti parametri:
Parametro Iniziale $\mathbf{C_0}$ controllo guadagno = 0,1
Fattore di Perdita $\boldsymbol{\lambda}$ = 0,7
Fattore di Compressione **r** = 0,2

In fig. 2.7.8 si può osservare come l'aumento del fattore di perdita influisca profondamente sia sull'uscita che sulla densità spettrale di potenza riducendone le oscillazioni al variare della frequenza.

Un notevole incremento del fattore di estensione r porta, invece, ad un fenomeno opposto come illustrato in fig. 2.7.7, dove tali oscillazioni si sono intensificate.

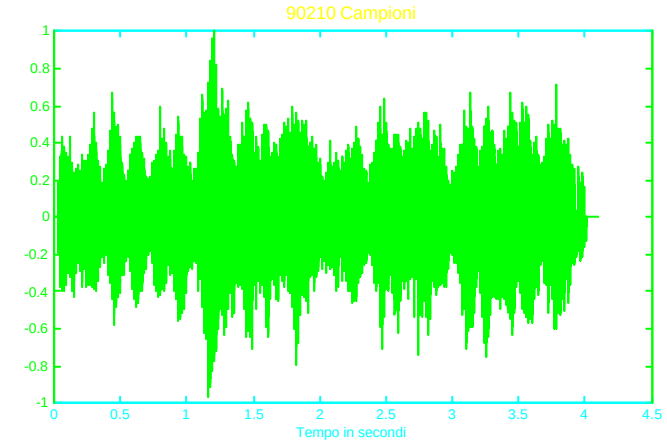


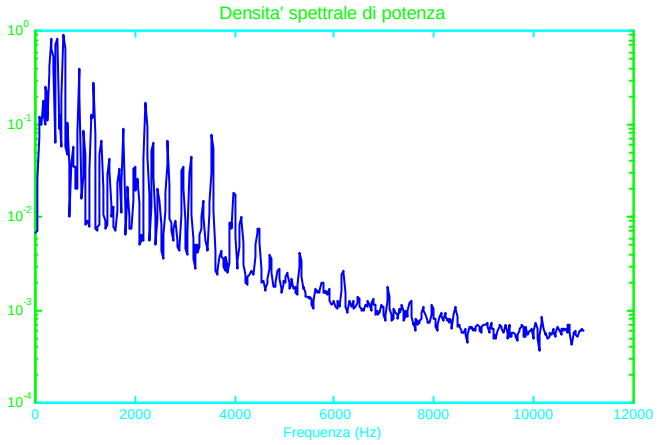


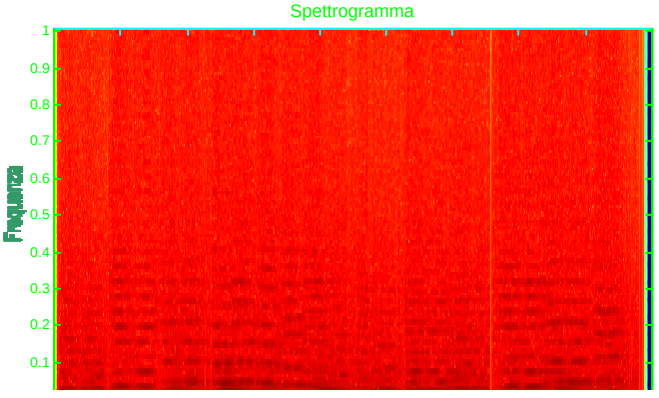


◂ **Fig. 2.7.6** Effetto Compressore su BranoPop.wav con i seguenti parametri:

Parametro Iniziale $\mathbf{C_0}$ controllo guadagno = 0,1
Fattore di Perdita $\boldsymbol{\lambda}$ = 0,6
Fattore di Compressione **r** = 0,6

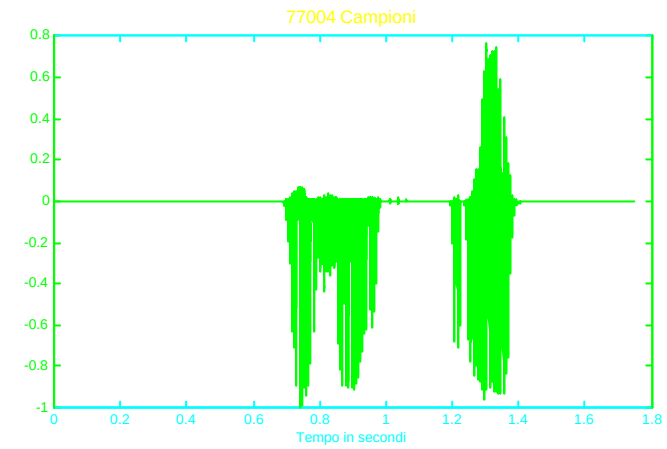


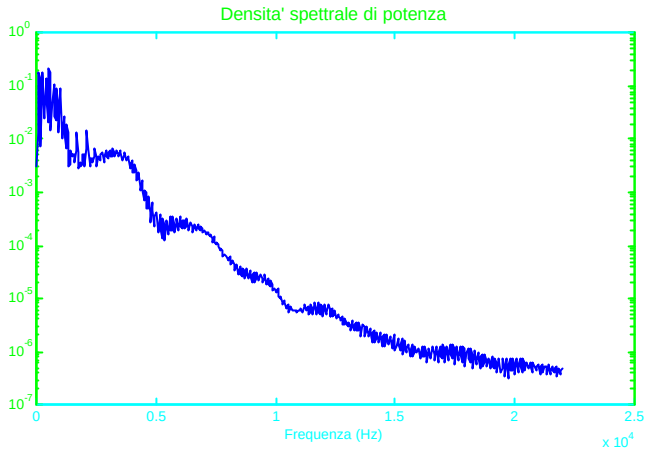


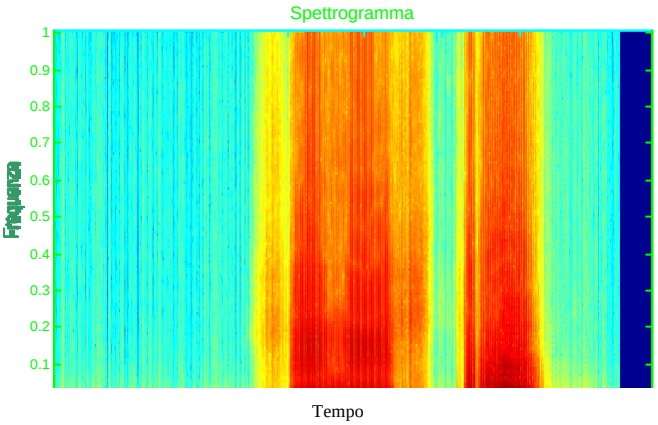


◂ **Fig. 2.7.7** Effetto Estensore su BranoCla.wav con i seguenti parametri:

Parametro Iniziale $\mathbf{C_0}$ controllo guadagno = 0,06
Fattore di Perdita $\boldsymbol{\lambda}$ = 0,3
Fattore di Estensione **r** = 20

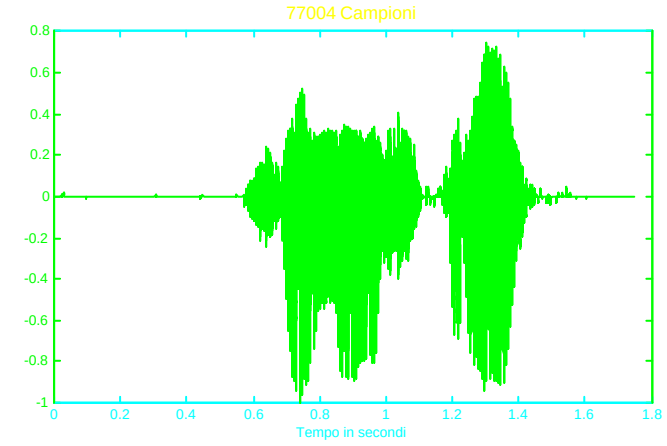


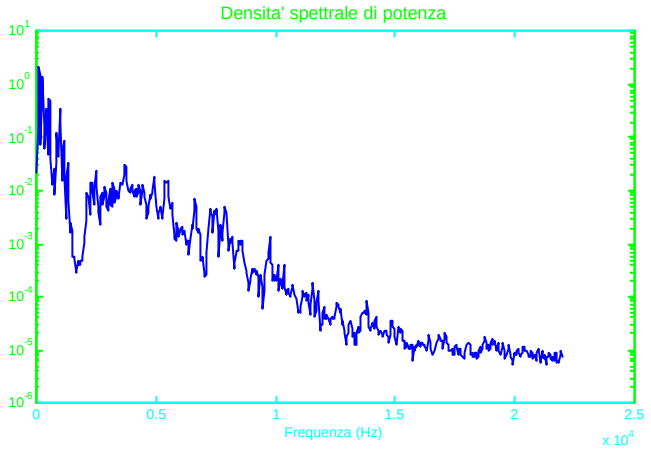


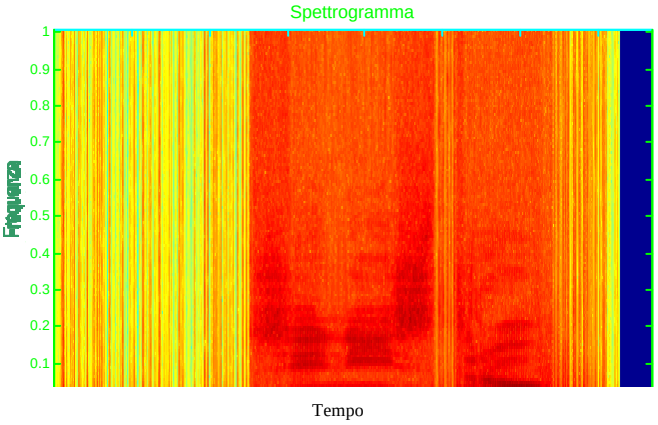


◄ **Fig. 2.7.8** Effetto Estensore su Brano03.wav con i seguenti parametri:

Parametro Iniziale $\mathbf{C_0}$ controllo guadagno = 0,5
Fattore di Perdita $\boldsymbol{\lambda}$ = 0,9
Fattore di Estensione **r** = 5







◄ **Fig. 2.7.9** Effetto Estensore su Brano03.wav con i seguenti parametri:

Parametro Iniziale $\mathbf{C_0}$ controllo guadagno = 0,03
Fattore di Perdita $\boldsymbol{\lambda}$ = 0,2
Fattore di Estensione **r** = 10

3. CONCLUSIONI

In questa trattazione sono stati descritti il progetto, la realizzazione ed alcuni esempi di risultati della elaborazione di effetti speciali sul segnale audio digitale. Il supporto dei grafici della risposta all'impulso, della densità spettrale di potenza e dello spettrogramma hanno fornito informazioni utili per capire come i filtri agiscono sulla sequenza numerica.

Per esempio è stato interessante osservare come la frequenza di variazione dei ritardi influisca profondamente sulla densità spettrale di potenza nell'effetto Phasing. Come l'aumento delle voci e della frequenza di sweep nell'effetto coro provochino una desincronizzazione via via crescente fino a rendere l'uscita indecifrabile. Oppure come il posizionamento di poli e zeri nei filtri influisca profondamente sui contenuti spettrali e frequenziali.

Infine, il sofware AudioFx 1.0 creato appositamente per tale scopo è stato sviluppato alla luce di possibili future evoluzioni. La sua struttura modulare permette, infatti, di aggiungere nuovi effetti, un help in linea, inserire nuovi parametri, perfezionare gli algoritmi ed i codici già esistenti, nonchè graficare nuove informazioni utili allo studio del segnale audio. Un'altra possibile integrazione interessante potrebbe riguardare l'introduzione di altri formati audio, oltre al WAVE.

4. BIBLIOGRAFIA

[1] Introduction to Signal Processing, S.J. Orfanidis, Prentice-Hall
[2] Digital Signal Processing, J.G.Proakis, D.G.Manolakis, Prentice-Hall
[3] Discrete-Time Signal Processing, A.V. Oppenheim, R.W. Schafer, Prentice-Hall
[4] Reverberation Algorithms, Fernando A. Beltràn, Josè R. Beltràn, Nicolas Holzem
[5] Colorless Artificial Reverberation, M.R. Schroeder, Audio Engineering Society
[6] Microsoft Win32 Multimedia Api Documentation
[7] Matlab User's Guide , MathWorks
[8] Building GUIs With Matlab – Version 5 , MathWorks
[9] Elementi di fisica II – Campi e Onde , M.Alonso, E.J.Finn, Masson

5. APPENDICE A: IL CODICE DI AUDIO FX 1.0

Di seguito viene riportato il codice completo del programma AudioFx 1.0

```
function fig = audiofx()
% Questo codice crea l'interfaccia grafica della Finestra principale

load audiofx

h0 = figure('Units','points', ...
    'Color',[0 0 0.627450980392157], ...
    'Colormap',mat0, ...
    'DoubleBuffer','on', ...
    'FileName','C:\AudioFx\audiofx.m', ...
    'KeyPressFcn','dokeypress(gcbf)', ...
    'PaperPosition',[18 180 576 432], ...
    'PaperUnits','points', ...
    'Position',[14.25 12 563.25 332.25], ...
    'ResizeFcn','doresize(gcbf)', ...
    'Tag','Fig1', ...
    'ToolBar','none', ...
    'DefaultaxesCreateFcn','plotedit(gcbf,"promoteoverlay"); ');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeFigObjStorage', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
```

```
    'HandleVisibility','off', ...
    'Tag','ScribeFigObjStorage', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeFigObjStorage', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeFigObjStorage', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeFigObjStorage', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeFigObjStorage', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeFigObjStorage', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
```

```
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeFigObjStorage', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeFigObjStorage', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeFigObjStorage', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeHGBinObject', ...
    'Visible','off');
h1 = uimenu('Parent',h0, ...
    'HandleVisibility','off', ...
    'Tag','ScribeFigObjStorage', ...
    'Visible','off');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'Callback','aperturafile                          ', ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[18.75 8.25 66 30], ...
    'String','Apri File', ...
    'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'Callback','wavplay(y,FS);', ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[91.5 25.5 96.75 12.75], ...
    'String','Play File Originale', ...
```

```
    'Tag','Pushbutton2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'Callback','wavplay(y3,FS);', ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[354.75 25.5 99.75 12.75], ...
    'String','Play File Nuovo', ...
    'Tag','Pushbutton4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'Callback',mat1, ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[461.25 9.75 81.75 13.5], ...
    'String','Salva File', ...
    'Tag','Pushbutton5', ...
    'Value',1);
h1 = uicontextmenu('Parent',h0, ...
    'Callback','domymenu update', ...
    'HandleVisibility','off', ...
    'Tag','ScribeAxisObjContextMenu');
h2 = uimenu('Parent',h1, ...
    'Callback','domymenu cut', ...
    'HandleVisibility','off', ...
    'Label','Cu&t', ...
    'Tag','ScribeAxisObjCutMenu');
h2 = uimenu('Parent',h1, ...
    'Callback','domymenu copy', ...
    'HandleVisibility','off', ...
    'Label','&Copy', ...
    'Tag','ScribeAxisObjCopyMenu');
h2 = uimenu('Parent',h1, ...
    'Callback','domymenu paste', ...
    'HandleVisibility','off', ...
    'Label','&Paste', ...
    'Tag','ScribeAxisObjPasteMenu');
h2 = uimenu('Parent',h1, ...
    'Callback','domymenu clear', ...
    'HandleVisibility','off', ...
    'Label','Clea&r', ...
    'Tag','ScribeAxisObjClearMenu');
```

```
h2 = uimenu('Parent',h1, ...
    'Callback','domymenu showlegend', ...
    'HandleVisibility','off', ...
    'Label','Show Legend', ...
    'Separator','on', ...
    'Tag','ScribeAxisObjShowLegendMenu');
h2 = uimenu('Parent',h1, ...
    'Callback','domymenu moveresize', ...
    'HandleVisibility','off', ...
    'Label','Lock Position', ...
    'Separator','on', ...
    'Tag','ScribeAxisObjMoveResizeMenu');
h2 = uimenu('Parent',h1, ...
    'Callback','domymenu more', ...
    'HandleVisibility','off', ...
    'Label','Properties...', ...
    'Separator','on', ...
    'Tag','ScribeAxischildObjMoreMenu');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'FontAngle','oblique', ...
    'FontSize',50, ...
    'ForegroundColor',[1 0 0], ...
    'ListboxTop',0, ...
    'Position',[18 261.75 525.75 57], ...
    'String','Audio FX', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'FontSize',14, ...
    'ListboxTop',0, ...
    'Position',[375 269.25 141.75 19.5], ...
    'String','by Emiliano Vezzoli', ...
    'Style','text', ...
    'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'ListboxTop',0, ...
    'Position',[206.25 9 131.25 197.25], ...
    'Style','frame', ...
    'Tag','Frame1');
```

```
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'Callback','Rimpulsodati', ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[461.25 25.5 82.5 13.5], ...
    'String','Risposta all''impulso', ...
    'Tag','Pushbutton3');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'FontSize',14, ...
    'ForegroundColor',[1 0 0], ...
    'ListboxTop',0, ...
    'Position',[210.75 186.75 123 18.75], ...
    'String','EFFETTI SPECIALI', ...
    'Style','text', ...
    'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'Callback',mat2, ...
    'ForegroundColor',[0 1 1], ...
    'ListboxTop',0, ...
    'Position',[212.25 171.75 120.75 13.5], ...
    'String','Eco', ...
    'Tag','Pushbutton6');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'Callback',mat3, ...
    'ForegroundColor',[0 1 1], ...
    'ListboxTop',0, ...
    'Position',[212.25 156 120.75 13.5], ...
    'String','Eco Multiplo', ...
    'Tag','Pushbutton6');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'Callback',mat4, ...
    'ForegroundColor',[0 1 1], ...
    'ListboxTop',0, ...
    'Position',[212.25 124.5 120.75 13.5], ...
    'String','Flanging', ...
```

```
    'Tag','Pushbutton6');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'Callback','Parametrifiltrophasing', ...
    'ForegroundColor',[0 1 1], ...
    'ListboxTop',0, ...
    'Position',[212.25 108.75 120.75 13.5], ...
    'String','Phasing', ...
    'Tag','Pushbutton6');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'Callback','Parametrifiltroriverbero', ...
    'ForegroundColor',[0 1 1], ...
    'ListboxTop',0, ...
    'Position',[212.25 93 120.75 13.5], ...
    'String','Riverbero ( Tipo I,II,III )', ...
    'Tag','Pushbutton6');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'Callback','Parametrifiltrocompressore', ...
    'ForegroundColor',[0 1 1], ...
    'ListboxTop',0, ...
    'Position',[212.25 77.25 120.75 13.5], ...
    'String','Compressore', ...
    'Tag','Pushbutton6');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'Callback','Parametrifiltroespansore', ...
    'ForegroundColor',[0 1 1], ...
    'ListboxTop',0, ...
    'Position',[212.25 61.5 120.75 13.5], ...
    'String','Estensore', ...
    'Tag','Pushbutton6');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'ForegroundColor',[0 1 1], ...
    'ListboxTop',0, ...
    'Position',[212.25 45.75 120.75 13.5], ...
    'Tag','Pushbutton6');
```

```
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 0 0], ...
    'Callback',mat5, ...
    'FontSize',10, ...
    'ForegroundColor',[1 1 0], ...
    'ListboxTop',0, ...
    'Position',[212.25 13.5 120.75 21], ...
    'String','Applica Effetto', ...
    'Tag','Pushbutton6');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'Callback',mat6, ...
    'ForegroundColor',[0 1 1], ...
    'ListboxTop',0, ...
    'Position',[212.25 140.25 120.75 13.5], ...
    'String','Coro', ...
    'Tag','Pushbutton6');
h1 = axes('Parent',h0, ...
    'Units','pixels', ...
    'AmbientLightColor',[0.658823529411765 0.827450980392157 1], ...
    'CameraUpVector',[0 1 0], ...
    'CameraUpVectorMode','manual', ...
    'Color',[0.658823529411765 0.827450980392157 1], ...
    'ColorOrder',mat7, ...
    'CreateFcn','plotedit(gcbf,''promoteoverlay''); ', ...
    'Position',[26 220 226 87], ...
    'Tag','Axes1', ...
    'XColor',[0 1 0], ...
    'YColor',[0 1 1], ...
    'ZColor',[0 0 0]);
h2 = text('Parent',h1, ...
    'Color',[0 1 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[0.4977777777777778 -0.2790697674418601 9.160254037844386],
...
    'Tag','Axes1Text4', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 1 1], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
```

```
    'Position',[-0.1377777777777778 0.4883720930232562 9.160254037844386],
...
    'Rotation',90, ...
    'Tag','Axes1Text3', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Position',mat8, ...
    'Tag','Axes1Text2', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',mat9, ...
    'Tag','Axes1Text1', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h1 = axes('Parent',h0, ...
    'Units','pixels', ...
    'AmbientLightColor',[0.658823529411765 0.827450980392157 1], ...
    'CameraUpVector',[0 1 0], ...
    'CameraUpVectorMode','manual', ...
    'Color',[0.658823529411765 0.827450980392157 1], ...
    'ColorOrder',mat10, ...
    'CreateFcn','plotedit(gcbf,''promoteoverlay''); ', ...
    'Position',[26 89 226 87], ...
    'Tag','Axes2', ...
    'XColor',[0 1 0], ...
    'YColor',[0 1 1], ...
    'ZColor',[0 0 0]);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 1 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[0.4977777777777778 -0.279069767441861 9.160254037844386], ...
    'Tag','Axes1Text4', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
```

```
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 1 1], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[-0.1377777777777778 0.4883720930232558 9.160254037844386],
...
    'Rotation',90, ...
    'Tag','Axes1Text3', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Interruptible','off', ...
    'Position',mat11, ...
    'Tag','Axes1Text2', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[0.4977777777777778 1.081395348837209 9.160254037844386], ...
    'Tag','Axes1Text1', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h1 = axes('Parent',h0, ...
    'Units','pixels', ...
    'AmbientLightColor',[0.658823529411765 0.827450980392157 1], ...
    'CameraUpVector',[0 1 0], ...
    'CameraUpVectorMode','manual', ...
    'Color',[0.658823529411765 0.827450980392157 1], ...
    'ColorOrder',mat12, ...
    'CreateFcn','plotedit(gcbf,''promoteoverlay''); ', ...
    'Position',[492 216 235 90], ...
    'Tag','Axes3', ...
    'XColor',[0 1 0], ...
    'YColor',[0 1 1], ...
    'ZColor',[0 0 0]);
```

```
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 1 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[0.4957264957264957 -0.2696629213483148 9.160254037844386],
...
    'Tag','Axes1Text4', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 1 1], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',mat13, ...
    'Rotation',90, ...
    'Tag','Axes1Text3', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Interruptible','off', ...
    'Position',[-2.102564102564103 2.539325842696629 9.160254037844386], ...
    'Tag','Axes1Text2', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',mat14, ...
    'Tag','Axes1Text1', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
```

```
h1 = axes('Parent',h0, ...
    'Units','pixels', ...
    'AmbientLightColor',[0.658823529411765 0.827450980392157 1], ...
    'CameraUpVector',[0 1 0], ...
    'CameraUpVectorMode','manual', ...
    'Color',[0.658823529411765 0.827450980392157 1], ...
    'ColorOrder',mat15, ...
    'CreateFcn','plotedit(gcbf,''promoteoverlay''); ', ...
    'Position',[492 88 234 87], ...
    'Tag','Axes4', ...
    'XColor',[0 1 0], ...
    'YColor',[0 1 1], ...
    'ZColor',[0 0 0]);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 1 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[0.4978540772532192 -0.279069767441861 9.160254037844386], ...
    'Tag','Axes1Text4', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 1 1], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[-0.133047210300429 0.4883720930232558 9.160254037844386], ...
    'Rotation',90, ...
    'Tag','Axes1Text3', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Interruptible','off', ...
    'Position',[-2.111587982832618 4.116279069767442 9.160254037844386], ...
    'Tag','Axes1Text2', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
```

```
h2 = text('Parent',h1, ...
    'ButtonDownFcn','ctlpanel SelectMoveResize', ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Interruptible','off', ...
    'Position',[0.4978540772532192 1.081395348837209 9.160254037844386], ...
    'Tag','Axes1Text1', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'Callback','wavplay(y1o,FS);', ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[355.5 9 46.5 12.75], ...
    'String','Play C.S.', ...
    'Tag','Pushbutton4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'Callback','wavplay(y2o,FS);', ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[407.25 9.75 46.5 12.75], ...
    'String','Play C.D.', ...
    'Tag','Pushbutton4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'ForegroundColor',[1 0 0], ...
    'ListboxTop',0, ...
    'Position',[18.75 133.5 74.25 10.5], ...
    'String','CANALE DESTRO', ...
    'Style','text', ...
    'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'ForegroundColor',[1 0 1], ...
    'ListboxTop',0, ...
    'Position',[373.5 135 87 8.25], ...
    'String','CANALE DESTRO FX', ...
    'Style','text', ...
```

```
    'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'ForegroundColor',[1 1 0], ...
    'ListboxTop',0, ...
    'Position',[18.75 231 78.75 10.5], ...
    'String','CANALE SINISTRO', ...
    'Style','text', ...
    'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'ForegroundColor',[0 1 0], ...
    'ListboxTop',0, ...
    'Position',[372 231.75 93.75 9.75], ...
    'String','CANALE SINISTRO FX', ...
    'Style','text', ...
    'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'Callback','wavplay(y(:,1),FS);', ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[90.75 9 46.5 12.75], ...
    'String','Play C.S.', ...
    'Tag','Pushbutton4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'Callback','wavplay(y(:,2),FS);', ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[141.75 9 46.5 12.75], ...
    'String','Play C.D.', ...
    'Tag','Pushbutton4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 0 0], ...
    'Callback',mat16, ...
    'ForegroundColor',[1 1 0], ...
    'ListboxTop',0, ...
    'Position',[96 231 45 12], ...
    'String','T.S.S.', ...
    'Tag','Pushbutton7');
```

```
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'FontSize',6, ...
    'ForegroundColor',[0 1 0], ...
    'ListboxTop',0, ...
    'Position',[64.5 39.75 78.75 12], ...
    'String','campioni', ...
    'Style','text', ...
    'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'FontSize',6, ...
    'ForegroundColor',[0 1 0], ...
    'ListboxTop',0, ...
    'Position',[428.25 39.75 52.5 12], ...
    'String','campioni', ...
    'Style','text', ...
    'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[149.25 38.25 45 12], ...
    'Style','frame', ...
    'Tag','Frame2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[504.75 39.75 45 9.75], ...
    'Style','frame', ...
    'Tag','Frame2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 0 0], ...
    'Callback',mat17, ...
    'ForegroundColor',[1 1 0], ...
    'ListboxTop',0, ...
    'Position',[95.25 133.5 45 12], ...
    'String','T.S.S.', ...
    'Tag','Pushbutton7');
```

```
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 0 0], ...
    'Callback',mat18, ...
    'ForegroundColor',[1 1 0], ...
    'ListboxTop',0, ...
    'Position',[467.25 231.75 45 12], ...
    'String','T.S.S.', ...
    'Tag','Pushbutton7');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 0 0], ...
    'Callback',mat19, ...
    'ForegroundColor',[1 1 0], ...
    'ListboxTop',0, ...
    'Position',[466.5 133.5 45 12], ...
    'String','T.S.S.', ...
    'Tag','Pushbutton7');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'FontAngle','italic', ...
    'FontSize',25, ...
    'ForegroundColor',[1 0 0], ...
    'ListboxTop',0, ...
    'Position',[390 286.5 36.75 28.5], ...
    'String','1.0', ...
    'Style','text', ...
    'Tag','StaticText5');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 0], ...
    'Callback','Info', ...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'ListboxTop',0, ...
    'Position',[206.25 216.75 130.5 13.5], ...
    'String','Informazioni su AudioFx', ...
    'Tag','Pushbutton3');
h1 = axes('Parent',h0, ...
    'CameraUpVector',[0 1 0], ...
    'CameraUpVectorMode','manual', …
    'Color','none', ...
    'ColorOrder',mat20, ...
    'CreateFcn','', ...
    'HandleVisibility','off', ...
```

```
    'HitTest','off', ...
    'Position',[0 0 1 1], ...
    'Tag','ScribeOverlayAxesActive', ...
    'Visible','off', ...
    'XColor',[0.8 0.8 0.8], ...
    'XLimMode','manual', ...
    'XTickMode','manual', ...
    'YColor',[0.8 0.8 0.8], ...
    'YLimMode','manual', ...
    'YTickMode','manual', ...
    'ZColor',[0 0 0]);
h2 = text('Parent',h1, ...
    'Color',[0.8 0.8 0.8], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[0.4986666666666667 -0.01809954751131238 9.160254037844386],
...
    'VerticalAlignment','cap', ...
    'Visible','off');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0.8 0.8 0.8], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[-0.009333333333333332 0.4977375565610859
9.160254037844386], ...
    'Rotation',90, ...
    'VerticalAlignment','baseline', ...
    'Visible','off');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Position',[-0.001333333333333333 0.997737556561086 9.160254037844386],
...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[0.4986666666666667 1.015837104072398 9.160254037844386], ...
    'VerticalAlignment','bottom', ...
    'Visible','off');
```

```
set(get(h2,'Parent'),'Title',h2);
if nargout > 0, fig = h0; end


% Operazioni apertura e plot della traccia audio
clear;
cla;

branooriginale=uigetfile('*.wav','Apri File Wav');
[y,FS,Nbits]=wavread(branooriginale);

subplot('Position',[ 0.0330250990752972 0.494974874371859
0.299867899603699 0.198492462311558 ])
plot(y(:,1),'y')
set(gca,'XColor','green','YColor','cyan','Color','black','FontSize',10);

subplot('Position',[ 0.0330250990752972 0.190954773869347
0.299867899603699 0.198492462311558 ])
plot(y(:,2),'r')
set(gca,'XColor','green','YColor','cyan','Color','black','FontSize',10);

% Acquisizione Parametri Filtro Eco Singolo  - Emiliano Vezzoli - vezzoli@libero.it
effetto=1;
prompt={'Canale a cui applicare ECO: 1=sinistro, 2=Destro, 0=Entrambi','Ritardo
(secondi)','Attenuazione( |a|<=1)' };
defans={'0','0.3','0.5'};
filelds={'canale','d','atten'}
info=inputdlg(prompt,'Parametri ECO',1,defans)
if isempty(info)
   msgbox('Verranno applicati i parametri predefiniti!','ATTENZIONE!');
   canale=0;
   d=0.3;
   atten=0.5;
else
   canale=info(1);
   d=info(2);
   atten=info(3);
end

% Acquisizione Parametri Filtro Coro - Emiliano Vezzoli - vezzoli@libero.it

effetto=3;
prompt={'Canale a cui applicare effetto CORO: 1=Sinistro, 2=Destro,
0=Entrambi','Ritardo MAX (secondi)','Attenuazione Minima ( |a|<=1)','Numero di
voci nel coro','Frequenza variazione ritardi random (cicli/simbolo):'};
```

```
defans={'0','0.3','0.6','3','0.0001'};
fields={'canale','d','atten','voci','freq'}
info=inputdlg(prompt,'Parametri EFFETTO CORO',1,defans)
if isempty(info)
   msgbox('Verranno applicati i parametri predefiniti!','ATTENZIONE!');
   canale=0;
   d=0.3;
   atten=0.6;
   voci=3;
   freq=0.0001;
else
   info=str2double(info);
   canale=info(1);
   d=info(2);
   atten=info(3);
   voci=info(4);
   freq=info(5);
end

% Acquisizione Parametri Filtro Flanger - Emiliano Vezzoli - vezzoli@libero.it
effetto=4;
prompt={'Canale a cui applicare effetto FLANGER: 1=Sinistro, 2=Destro,
0=Entrambi','Ritardo MAX (secondi)','Attenuazione Minima (|a|<=1)','Frequenza
variazione ritardi (rad/sec)'};
defans={'0','0.4','0.4','0.1'};
fields={'canale','d','atten','freq'}
info=inputdlg(prompt,'Parametri EFFETTO FLANGER',1,defans)
if isempty(info)
   msgbox('Verranno applicati i parametri predefiniti!','ATTENZIONE!');
   canale=0;
   d=0.4;
   atten=0.4;
   freq=0.1;
else
   info=str2double(info);
   canale=info(1)
   d=info(2)
   atten=info(3)
   freq=info(4)
end

% Acquisizione Parametri Filtro Phasing  - Emiliano Vezzoli - vezzoli@libero.it
effetto=5;
d=0.3;
prompt={'Canale a cui applicare effetto PHASING: 1=Sinistro, 2=Destro,
```

```
0=Entrambi','Attenuazione Minima (|a|<=1)','Estremi intervallo variazione
frequenza di Notch (w1<w0<w2): w1 (pi*rad/sec)','w2 (pi*rad/sec)','Frequenza di
Sweep (pi*rad/sec)','Q del filtro Notch'};
defans={'0','0.8','0.11','0.75','0.85','3.5'};
fields={'canale','atten','w1','w2','wsweep','Q'}
info=inputdlg(prompt,'Parametri EFFETTO PHASING',1,defans)
if isempty(info)
   msgbox('Verranno applicati i parametri predefiniti!','ATTENZIONE!');
   canale=0;
   atten=0.8;
   w1=0.11;
   w2=0.75;
   wsweep=0.85;
   Q=3.5;
else
   info=str2double(info);
   canale=info(1)
   atten=info(2)
   w1=info(3)
   w2=info(4)
   wsweep=info(5)
   Q=info(6)
end

% Acquisizione Parametri Filtro Riverbero - Emiliano Vezzoli - vezzoli@libero.it
effetto=6;
prompt1={'Tipo di riverbero da applicare: 1=Piano, 2=Passatutto(Schroeder),
3=Passabasso'};
defans1={'1'};
fields={'riv'}
info1=inputdlg(prompt1,'TIPO DI RIVERBERO',1,defans1)
if isempty(info1)
   msgbox('Verranno applicati i parametri predefiniti!','ATTENZIONE!');
   riv=1;
else
   info1=str2double(info1);
   riv=info1(1);
end
if riv==1
prompt={'Canale a cui applicare effetto RIVERBERO PIANO: 1=Sinistro,
2=Destro, 0=Entrambi','Attenuazione Minima (|a|<=1)','Ritardo MAX (secondi)'};
defans={'0','0.7','0.1'};
fields={'canale','atten','d'}
```

```
info=inputdlg(prompt,'Parametri EFFETTO RIVERBERO PIANO',1,defans)
if isempty(info)
   msgbox('Verranno applicati i parametri predefiniti!','ATTENZIONE!');
   canale=0;
   atten=0.7;
   d=0.1;
else
   info=str2double(info);
   canale=info(1)
   atten=info(2)
   d=info(3)
end
end
if riv==2
prompt={'Canale a cui applicare effetto RIVERBERO DI SCHROEDER:
1=Sinistro, 2=Destro, 0=Entrambi','Attenuazione Minima (|a|<=1)','Ritardo MAX
(secondi)'};
defans={'0','0.7','0.1'};
fields={'canale','atten','d'}
info=inputdlg(prompt,'Parametri EFFETTO RIVERBERO  DI
SCHROEDER',1,defans)
if isempty(info)
   msgbox('Verranno applicati i parametri predefiniti!','ATTENZIONE!');
   canale=0;
   atten=0.7;
   d=0.1;
else
   info=str2double(info);
   canale=info(1)
   atten=info(2)
   d=info(3)
end
end
if riv==3
atten=0
prompt={'Canale a cui applicare effetto RIVERBERO PASSABASSO: 1=Sinistro,
2=Destro, 0=Entrambi','Ritardo MAX (secondi)','Ordine M del filtro di Feedback
G(z)','M+1 Coefficienti b(i) del numeratore di G(z): Es: 1 2 1 3...','M Coefficienti c(i)
del denominatore di G(z): Es: 1 2 0 ...'};
defans={'0','0.05','2','1 2 1','2 2'};
fields={'canale','d','M','b','c'}
info=inputdlg(prompt,'Parametri EFFETTO RIVERBERO
PASSABASSO',1,defans)
if isempty(info)
   msgbox('Verranno applicati i parametri predefiniti!','ATTENZIONE!');
```

```
   canale=0;
   d=0.05;
   M=2;
   b=[1 2 1];
   c=[2 2];
else
   canale=str2double(info(1));
   d=str2double(info(2));
   M=str2double(info(3));
   b=char(info(4));
   b=str2num(b);
   c=char(info(5));
   c=str2num(c);
end
end

% Acquisizione Parametri Filtro Compressore
effetto=7;
d=0;
atten=0;
prompt={'Canale a cui applicare effetto COMPRESSIONE: 1=Sinistro, 2=Destro,
0=Entrambi','Parametro Iniziale "Co" Controllo Guadagno:','Fattore di Perdita (
h<=1 ):','Fattore di Compressione (r<1):'};
defans={'0','0.025','0.2','0.5'};
fields={'canale','c0','h','r'}
info=inputdlg(prompt,'Parametri EFFETTO COMPRESSIONE',1,defans)
if isempty(info)
   msgbox('Verranno applicati i parametri predefiniti!','ATTENZIONE!');
   canale=0;
   c0=0.025;
   h=0.2;
   r=0.5;
else
   info=str2double(info);
   canale=info(1)
   c0=info(2)
   h=info(3)
   r=info(4)
end

% Acquisizione Parametri Filtro Estensore  - Emiliano Vezzoli - vezzoli@libero.it
effetto=8;
d=0;
atten=0;
```

```
prompt={'Canale a cui applicare effetto ESPANSIONE: 1=Sinistro, 2=Destro,
0=Entrambi','Parametro Iniziale "Co" Controllo Guadagno:' ,'Fattore di Perdita (
h<=1 ):','Fattore di Espansione (r>1):'};
defans={'0','0.03','0.2','1.5'};
fields={'canale','c0','h','r'}
info=inputdlg(prompt,'Parametri EFFETTO ESPANSIONE',1,defans)
if isempty(info)
   msgbox('Verranno applicati i parametri predefiniti!','ATTENZIONE!');
   canale=0;
   c0=0.03;
   h=0.2;
   r=1.5;
else
   info=str2double(info);
   canale=info(1)
   c0=info(2)
   h=info(3)
   r=info(4)
end

function fig = Rimpulso()
% Questa funzione crea la finestra con la risposta all'impulso

load Rimpulso
h0 = figure('Units','points', ...
    'Color',[0 0 0.627450980392157], ...
    'Colormap',mat0, ...
    'FileName','C:\Impulso\Rimpulso.m', ...
    'PaperPosition',[18 180 576 432], ...
    'PaperUnits','points', ...
    'Position',[31.5 -63.75 524.25 282.75], ...
    'Tag','Fig1', ...
    'ToolBar','none');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0 0 0.627450980392157], ...
    'FontSize',14, ...
    'FontWeight','bold', ...
    'ForegroundColor',[1 1 0], ...
    'ListboxTop',0, ...
    'Position',[129.75 244.5 244.5 22.5], ...
    'String','RISPOSTA ALL"IMPULSO', ...
    'Style','text', ...
    'Tag','StaticText1');
h1 = axes('Parent',h0, ...
    'Units','pixels', ...
```

```
'AmbientLightColor',[0.658823529411765 0.827450980392157 1], ...
    'CameraUpVector',[0 1 0], ...
    'Color',[0.658823529411765 0.827450980392157 1], ...
    'ColorOrder',mat1, ...
    'Position',[41 40 634 279], ...
    'Tag','Axes1', ...
    'XColor',[0 1 1], ...
    'YColor',[0 1 0], ...
    'ZColor',[0 0 0]);
h2 = text('Parent',h1, ...
    'Color',[0 1 1], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[0.4992101105845182 -0.08633093525179869 9.160254037844386],
...
    'Tag','Axes1Text4', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 1 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[-0.04897314375987361 0.4964028776978416 9.160254037844386],
...
    'Rotation',90, ...
    'Tag','Axes1Text3', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Position',[-0.06477093206951026 1.20863309352518 9.160254037844386], ...
    'Tag','Axes1Text2', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[0.4992101105845182 1.025179856115108 9.160254037844386], ...
    'Tag','Axes1Text1', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
if nargout > 0, fig = h0; end
```

```
% Simulazione della risposta all'impulso considerando un ingresso di soli 100
campioni.
% Per vedere in modo qualitativo il comportamento dei filtri al variare dei vari
parametri,
% all'impulso e' stata preferita una sequenza di 10 campioni unitari dopo i primi
cinque
% nulli, sequita da tutti zeri. - Emiliano Vezzoli - vezzoli@libero.it

ritardo=(d*FS);
a=atten;
campioni=size(y);
cmp=campioni(1);
y1=y(:,1);
y2=y(:,2);
Rimpulso
x=[0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0];

% EFFETTO ECO
if effetto==1
  for i=1:100
    indice=i-ritardo;
    while (indice<=0)
      indice=indice+100;
    end
      uscita(i)=x(i)+a*x(indice);
   end
    uscita=uscita';
end

% EFFETTO ECO MULTIPLO
if effetto==2
  rit=10*d;
  uscita=x;
  for i=(rit+1):100
      uscita(i)=(a*(uscita(i-rit)))+x(i);
    end
    uscita=uscita';
 end

% EFFETTO CORO
if effetto==3
  rit=d*10;
  periodo=1/freq;
```

```
   nperiodi=round(100000/periodo);
   for i=1:100
     for voc=1:(voci-1)
       d(i,voc)=0;
      a1(i,voc)=0;
   end
   end
   % Creo un vettore di ritardi variabili 'd' e amplific variabili 'a1'
   % uno per ogni voce nel coro ( esclusa l'originale )
for voc=1:(voci-1)
j=1;
   for n=1:nperiodi
   r(n)=randn(1,1);
   v(n)=r(n)/2;
   % Arrotondamento per creare ritardi interi.
   d2(n)=round(rit*(0.5+v(n)));
   % Creo vettore di guadagni variabili ( sempre 0<ai(k)<1 )
   a2(n)=abs(a*(sin(2*pi*(abs(r(n))))));
   for i=1:periodo
      d3(n,i)=d2(n);
      a3(n,i)=a2(n);
      d(j,voc)=d3(n,i);
      a1(j,voc)=a3(n,i);
      j=j+1;
   end
end
end
      for i=1:100
      uscita(i)=x(i);
       % aggiunta delle voci
      for voc=1:(voci-1)
            % Creo ritardi circolari
            indice(i,voc)=i-d(i,voc);
               while (indice(i,voc)<=0)
                    indice(i,voc)=indice(i,voc)+100;
              end
              while (indice(i,voc)>100)
                    indice(i,voc)=indice(i,voc)-100;
              end
              uscita(i)=uscita(i)+(a1(i,voc)*x(indice(i,voc)));
       end
     end
  end

  % EFFETTO FLANGER
  if effetto==4
```

```
    rit=d*10
   % Creo un vettore di ritardi variabili nel tempo
    for k=1:100
     d2(k)=(rit/2)*(1-cos(2*pi*freq*k));
     % Arrotondamento per creare ritardi interi.
     d(k)=round(d2(k));
     end
   for i=1:100
      % Ritardo circolare
      indice(i)=i-d(i);
      while (indice(i)<=0)
      indice(i)=100+indice(i);
      end
      uscita(i)=x(i)+(a*x(indice(i)));
   end
   uscita=uscita';
 end

 % EFFETTO PHASING
if effetto==5
      % Filtro Notch variabile + Filtro Phasing
      for i=1:100
        w0(i)=w1+(w2*sin(wsweep*1));
          Deltaw=(w0(i)/Q);
        b=(1/(1+tan(Deltaw/2)));
        % Canale Sinistro + ritardi circolari
        indice1=i-(1*FS);
              while (indice1<=0)
                    indice1=100+indice1;
                end
                while (indice1>100)
                  indice1=indice1-100;
                  end
         indice2=i-(2*FS);
              while (indice2<=0)
                    indice2=100+indice2;
                end
                while (indice2>100)
                  indice2=indice2-100;
                  end
        W01(i)=b*x(i)+(2*b*(x(indice1))*cos(w0(i)))-(((2*b)-1)*(x(indice2)));
        uscita(i)=x(i)+(a*(W01(i)-(2*(x(indice1))*cos(w0(i)))+(x(indice2))));
          x(indice2)=x(indice1);
        x(indice1)=W01(i);
```

```
end
    uscita=uscita';
 end

% EFFETTO RIVERBERO PIANO
if ((effetto==6)&(riv==1))
  % Azzero coefficienti num e den del filtro H(z)
  rit=10*d;
  for j=1:(rit+1)
    b(j)=0;
    c(j)=0;
  end
  b(1)=1;
  c(1)=1;
  c(rit+1)=-atten;
  uscita=filter(b,c,x);
 end

% EFFETTO RIVERBERO PASSATUTTO (SCHROEDER)
if ((effetto==6)&(riv==2))
  rit=10*d;
  % Azzero coefficienti num e den del filtro H(z)
  for j=1:(rit+1)
    b(j)=0;
    c(j)=0;
  end
  b(1)=-atten;
  b(rit+1)=1;
  c(1)=1;
  c(rit+1)=-atten;
  uscita=filter(b,c,x);
 end

% EFFETTO RIVERBERO PASSABASSO
if ((effetto==6)&(riv==3))
  % Azzero coefficienti non necessari di num e den del filtro G(z) passabasso
  rit=100*d;
  for j=(M+2):(rit+1)
    b(j)=0;
  end
  for p=(M+1):(rit+1)
    c(p)=0;
  end
  uscita=x;
    for i=1:100
      % Ritardo circolare
```

```
 indice=i-rit;
        while (indice<=0)
        indice=100+indice;
        end
        uscita(i)=filter(b,c,uscita(indice))+x(i);
      end

end

% EFFETTO COMPRESSORE
if effetto==7
   c(1)=0;
      for i=(2):100
         c(i)=(h*c(i-1))+((1-h)*abs(x(i)));
         if c(i)>=c0
            G(i)=((c(i)/c0)^(r-1));
         else
            G(i)=1;
         end
         uscita(i)=G(i)*x(i);
      end
      uscita=uscita';
end

% EFFETTO ESTENSORE
if effetto==8
   c(1)=0;
      for i=(2):100
         c(i)=(h*c(i-1))+((1-h)*abs(x(i)));
         if c(i)<=c0
            G(i)=((c(i)/c0)^(r-1));
         else
            G(i)=1;
         end
         uscita(i)=G(i)*x(i);
      end
      uscita=uscita';
end
% Controllo e troncamento dei valori che non rientrano nel range -1,1
        for i=1:100
                if (uscita(i)<-1)
                        uscita(i)=-1;
                elseif (uscita(i)>1)
                        uscita(i)=1;
                end
```

```
        end
plot(uscita,'w')
set(gca,'XColor','cyan','YColor','green','Color','black','YLim',[-0.3 1.3]);
 [beep,f]=wavread('Beep.wav');
wavplay(beep,f)

% Filtri con Effetti Speciali - Emiliano Vezzoli - vezzoli@libero.it
msgbox('ELABORAZIONE DATI IN CORSO...');
pause(2);
close
ritardo=(d*FS);
a=atten;
campioni=size(y);
cmp=campioni(1);
y1=y(:,1);
y2=y(:,2);
%  Azzero le eventuali uscite, se gia' presenti.
y1o(:,:)=0;
y2o(:,:)=0;
y3(:,:)=0;

% EFFETTO ECO
if effetto==1
   y1o=y1;
   y2o=y2;
   if canale==0
      for i=(ritardo+1):cmp
         y1o(i)=y1(i)+a*y1(i-ritardo);
         y2o(i)=y2(i)+a*y2(i-ritardo);
       end
   elseif canale==1
      for i=(ritardo+1):cmp
         y1o(i)=y1(i)+a*y1(i-ritardo);
      end
      y2o=y2;
   elseif canale==2
      for i=(ritardo+1):cmp
         y2o(i)=y2(i)+a*y2(i-ritardo);
      end
    y1o=y1;
    end
end
```

```
% EFFETTO ECO MULTIPLO

if effetto==2
   y1o=y1;
   y2o=y2;
   if canale==0
      for i=(ritardo+1):cmp
         y1o(i)=(a*(y1o(i-ritardo)))+y1(i);
         y2o(i)=(a*(y2o(i-ritardo)))+y2(i);
      end
   elseif canale==1
      for i=(ritardo+1):cmp
         y1o(i)=(a*(y1o(i-ritardo)))+y1(i);
      end
      y2o=y2;
   elseif canale==2
      for i=(ritardo+1):cmp
         y2o(i)=(a*(y2o(i-ritardo)))+y2(i);
      end
      y1o=y1;
      end
  end

% EFFETTO CORO
if effetto==3
   periodo=1/freq;
   nperiodi=round(cmp/periodo);
   for i=1:cmp
     for voc=1:(voci-1)
       d(i,voc)=0;
      a1(i,voc)=0;
   end
   end
   % Creo un vettore di ritardi variabili 'd' e amplific variabili 'a1'
   % uno per ogni voce nel coro ( esclusa l'originale )
for voc=1:(voci-1)
j=1;
   for n=1:nperiodi
   r(n)=randn(1,1);
   v(n)=r(n)/2;
   % Arrotondamento per creare ritardi interi.
   d2(n)=round(ritardo*(0.5+v(n)));
   % Creo vettore di guadagni variabili ( sempre 0<ai(k)<1 )
   a2(n)=abs(a*(sin(2*pi*(abs(r(n))))));
   for i=1:periodo
      d3(n,i)=d2(n);
```

```
      a3(n,i)=a2(n);
      d(j,voc)=d3(n,i);
      a1(j,voc)=a3(n,i);
      j=j+1;
    end
end
end
if canale==0
     for i=1:cmp
      y1o(i)=y1(i);
      y2o(i)=y2(i);
       % aggiunta delle voci
      for voc=1:(voci-1)
            % Creo ritardi circolari
            indice(i,voc)=i-d(i,voc);
              while (indice(i,voc)<=0)
                    indice(i,voc)=indice(i,voc)+cmp;
              end
              while (indice(i,voc)>cmp)
                    indice(i,voc)=indice(i,voc)-cmp;
              end
            y1o(i)=y1o(i)+(a1(i,voc)*y1(indice(i,voc)));
            y2o(i)=y2o(i)+(a1(i,voc)*y2(indice(i,voc)));
         end
     end
   y1o=y1o';
   y2o=y2o';
   elseif canale==1
      for i=1:cmp
      y1o(i)=y1(i);
       % aggiunta delle voci
      for voc=1:(voci-1)
            % Creo ritardi circolari
            indice(i,voc)=i-d(i,voc);
              while (indice(i,voc)<=0)
                    indice(i,voc)=indice(i,voc)+cmp;
              end
              while (indice(i,voc)>cmp)
                    indice(i,voc)=indice(i,voc)-cmp;
              end
              y1o(i)=y1o(i)+(a1(i,voc)*y1(indice(i,voc)));
       end
     end
    y2o=y2;
    y1o=y1o';
```

```
 elseif canale==2
   for i=1:cmp
   y2o(i)=y2(i);
    % aggiunta delle voci
   for voc=1:(voci-1)
        % Creo ritardi circolari
        indice(i,voc)=i-d(i,voc);
          while (indice(i,voc)<=0)
               indice(i,voc)=indice(i,voc)+cmp;
          end
          while (indice(i,voc)>cmp)
               indice(i,voc)=indice(i,voc)-cmp;
          end
        y2o(i)=y2o(i)+(a1(i,voc)*y2(indice(i,voc)));
    end
  end
 y1o=y1;
 y2o=y2o';
 end
end

% EFFETTO FLANGER
if effetto==4
  % Creo un vettore di ritardi variabili nel tempo
    for k=1:cmp
     d2(k)=(ritardo/2)*(1-cos(2*pi*freq*k));
     % Arrotondamento per creare ritardi interi.
     d(k)=round(d2(k));
     end
 if canale==0
   for i=1:cmp
     % Ritardo circolare
     indice(i)=i-d(i);
     while (indice(i)<=0)
     indice(i)=cmp+indice(i);
     end
     y1o(i)=y1(i)+(a*y1(indice(i)));
     y2o(i)=y2(i)+(a*y2(indice(i)));
   end
 y1o=y1o';
 y2o=y2o';
 elseif canale==1
   for i=1:cmp
     % Ritardo circolare
     indice(i)=i-d(i);
     while (indice(i)<=0)
```

```
        indice(i)=cmp+indice(i);
        end
        y1o(i)=y1(i)+(a*y1(indice(i)));
      end
      y2o=y2;
      y1o=y1o';
   elseif canale==2
      for i=1:cmp
        % Ritardo circolare
        indice(i)=i-d(i);
        while (indice(i)<=0)
        indice(i)=cmp+indice(i);
        end
        y2o(i)=y2(i)+(a*y2(indice(i)));
      end
      y1o=y1;
      y2o=y2o';
    end
  end

 % EFFETTO PHASING
if effetto==5
   if canale==0
        % Filtro Notch variabile + Filtro Phasing
      for i=1:cmp
                w0(i)=w1+(w2*sin(wsweep*1));
             Deltaw=(w0(i)/Q);
           b=(1/(1+tan(Deltaw/2)));
           % Calcolo ritardi circolari
           indice1=i-(1*FS);
                  while (indice1<=0)
                        indice1=cmp+indice1;
                      end
            indice2=i-(2*FS);
                  while (indice2<=0)
                        indice2=cmp+indice2;
                    end
           % Canale Sinistro
           W01(i)=b*y1(i)+(2*b*(y1(indice1))*cos(w0(i)))-(((2*b)-1)*(y1(indice2)));
           y1o(i)=y1(i)+(a*(W01(i)-(2*(y1(indice1))*cos(w0(i)))+(y1(indice2))));
             y1(indice2)=y1(indice1);
           y1(indice1)=W01(i);
           % Canale Destro
           W02(i)=b*y2(i)+(2*b*(y2(indice1))*cos(w0(i)))-(((2*b)-1)*(y2(indice2)));
           y2o(i)=y2(i)+(a*(W02(i)-(2*(y2(indice1))*cos(w0(i)))+(y2(indice2))));
             y2(indice2)=y2(indice1);
```

```
        y2(indice1)=W02(i);
        y1o=y1o';
        y2o=y2o';
    end
elseif canale==1
  % Filtro Notch variabile + Filtro Phasing
  for i=1:cmp
            w0(i)=w1+(w2*sin(wsweep*1));
      Deltaw=(w0(i)/Q);
      b=(1/(1+tan(Deltaw/2)));
      % Canale Sinistro + ritardi circolari
      indice1=i-(1*FS);
            while (indice1<=0)
                  indice1=cmp+indice1;
                end
       indice2=i-(2*FS);
            while (indice2<=0)
                  indice2=cmp+indice2;
                end
      W01(i)=b*y1(i)+(2*b*(y1(indice1))*cos(w0(i)))-(((2*b)-1)*(y1(indice2)));
      y1o(i)=y1(i)+(a*(W01(i)-(2*(y1(indice1))*cos(w0(i)))+(y1(indice2))));
        y1(indice2)=y1(indice1);
      y1(indice1)=W01(i);
    end
   y1o=y1o';
   y2o=y2;
elseif canale==2
  % Filtro Notch variabile + Filtro Phasing
  for i=1:cmp
            w0(i)=w1+(w2*sin(wsweep*1));
      Deltaw=(w0(i)/Q);
      b=(1/(1+tan(Deltaw/2)));
      % Canale Destro + ritardi circolari
      indice1=i-(1*FS);
            while (indice1<=0)
                  indice1=cmp+indice1;
                end
      indice2=i-(2*FS);
            while (indice2<=0)
                  indice2=cmp+indice2;
                end
      W02(i)=b*y2(i)+(2*b*(y2(indice1))*cos(w0(i)))-(((2*b)-1)*(y2(indice2)));
      y2o(i)=y2(i)+(a*(W02(i)-(2*(y2(indice1))*cos(w0(i)))+(y2(indice2))));
        y2(indice2)=y2(indice1);
      y2(indice1)=W02(i);
```

```
      end
      y2o=y2o';
      y1o=y1;
    end
  end

% EFFETTO RIVERBERO PIANO

if ((effetto==6)&(riv==1))
   % Azzero coefficienti num e den del filtro H(z)
   for j=1:(ritardo+1)
      b(j)=0;
      c(j)=0;
   end
   b(1)=1;
   c(1)=1;
   c(ritardo+1)=-atten;
   if canale==0
         y1o=filter(b,c,y1);
         y2o=filter(b,c,y2);
   elseif canale==1
         y1o=filter(b,c,y1);
         y2o=y2;
   elseif canale==2
         y2o=filter(b,c,y2);
         y1o=y1;
    end
  end

% EFFETTO RIVERBERO PASSATUTTO (SCHROEDER)

if ((effetto==6)&(riv==2))
   % Azzero coefficienti num e den del filtro H(z)
   for j=1:(ritardo+1)
      b(j)=0;
      c(j)=0;
   end
   b(1)=-atten;
   b(ritardo+1)=1;
   c(1)=1;
   c(ritardo+1)=-atten;
   if canale==0
         y1o=filter(b,c,y1);
```

```
        y2o=filter(b,c,y2);
    elseif canale==1
      y1o=filter(b,c,y1);
      y2o=y2;
    elseif canale==2
     y2o=filter(b,c,y2);
     y1o=y1;
     end
  end

% EFFETTO RIVERBERO PASSABASSO

if ((effetto==6)&(riv==3))
        % Azzero coefficienti non necessari di num e den del filtro G(z) passabasso
    for j=(M+2):(ritardo+1)
      b(j)=0;
    end
    for p=(M+1):(ritardo+1)
      c(p)=0;
    end
    if canale==0
      y1o=y1;
      y2o=y2;
      for i=1:cmp
        % Ritardo circolare
        indice=i-ritardo;
        while (indice<=0)
        indice=cmp+indice;
        end
        y1o(i)=filter(b,c,y1o(indice))+y1(i);
        y2o(i)=filter(b,c,y2o(indice))+y2(i);
      end
      %y1o=y1o';
      %y2o=y2o';
      elseif canale==1
        y1o=y1;
      for i=1:cmp
        % Ritardo circolare
        indice=i-ritardo;
        while (indice<=0)
        indice=cmp+indice;
        end
        y1o(i)=filter(b,c,y1o(indice))+y1(i);
      end
      %y1o=y1o';
```

```
    y2o=y2;
  elseif canale==2
    y20=y2;
    for i=1:cmp
      % Ritardo circolare
      indice=i-ritardo;
      while (indice<=0)
      indice=cmp+indice;
      end
      y2o(i)=filter(b,c,y2o(indice))+y2(i);
    end
   %y2o=y2o';
   y1o=y1;
   end
  end

% EFFETTO COMPRESSORE

if effetto==7
  c1(1)=0;
  c2(1)=0;
  c(1)=0;
  if canale==0
    for i=(2):cmp
      c1(i)=(h*c1(i-1))+((1-h)*abs(y1(i)));
      c2(i)=(h*c2(i-1))+((1-h)*abs(y2(i)));
       if c1(i)>=c0
         G1(i)=((c1(i)/c0)^(r-1));
       else
         G1(i)=1;
       end
       if c2(i)>=c0
         G2(i)=((c2(i)/c0)^(r-1));
       else
         G2(i)=1;
       end
  y1o(i)=G1(i)*y1(i);
  y2o(i)=G2(i)*y2(i);
    end
      y1o=y1o';
      y2o=y2o';
  elseif canale==1
    for i=(2):cmp
      c(i)=(h*c(i-1))+((1-h)*abs(y1(i)));
      if c(i)>=c0
```

```
        G(i)=((c(i)/c0)^(r-1));
        else
          G(i)=1;
        end
        y1o(i)=G(i)*y1(i);
      end
      y1o=y1o';
      y2o=y2;
    elseif canale==2
      for i=(2):cmp
        c(i)=(h*c(i-1))+((1-h)*abs(y1(i)));
        if c(i)>=c0
          G(i)=((c(i)/c0)^(r-1));
        else
          G(i)=1;
        end
        y2o(i)=G(i)*y2(i);
      end
     y2o=y2o';
     y1o=y1;
     end
end

% EFFETTO ESTENSORE
if effetto==8
   c(1)=0;
   c1(1)=0;
   c2(1)=0;
   if canale==0
     for i=2:cmp
       c1(i)=(h*c1(i-1))+((1-h)*abs(y1(i)));
      c2(i)=(h*c2(i-1))+((1-h)*abs(y2(i)));
       if c1(i)<=c0
         G1(i)=((c1(i)/c0)^(r-1));
       else
         G1(i)=1;
       end
       if c2(i)<=c0
         G2(i)=((c2(i)/c0)^(r-1));
       else
         G2(i)=1;
       end
   y1o(i)=G1(i)*y1(i);
   y2o(i)=G2(i)*y2(i);
```

```
    end
        y1o=y1o';
        y2o=y2o';
   elseif canale==1
     for i=(2):cmp
        c(i)=(h*c(i-1))+((1-h)*abs(y1(i)));
        if c(i)<=c0
          G(i)=((c(i)/c0)^(r-1));
        else
          G(i)=1;
        end
        y1o(i)=G(i)*y1(i);
     end
     y1o=y1o';
     y2o=y2;
   elseif canale==2
     for i=(2):cmp
        c(i)=(h*c(i-1))+((1-h)*abs(y1(i)));
        if c(i)<=c0
          G(i)=((c(i)/c0)^(r-1));
        else
          G(i)=1;
        end
        y2o(i)=G(i)*y2(i);
     end
    y2o=y2o';
    y1o=y1;
    end
end
% Controllo e troncamento dei valori che non rientrano nel range -1,1
        for i=1:cmp
                if (y1o(i)<-1)
                        y1o(i)=-1;
                elseif (y1o(i)>1)
                        y1o(i)=1;
                end
                if (y2o(i)<-1)
                        y2o(i)=-1;
                elseif (y2o(i)>1)
                        y2o(i)=1;
                end
         end
% Creo file d'uscita
y3(:,1)=y1o;
y3(:,2)=y2o;
```

```
% Plotting dei nuovi canali audio con gli effetti
subplot('Position',[ 0.660025090752972 0.494974874371859
0.299867899603699 0.198492462311558 ])
plot(y1o,'g')
set(gca,'XColor','green','YColor','cyan','Color','black','FontSize',10);
subplot('Position',[ 0.660025090752972 0.190954773869347
0.299867899603699 0.198492462311558 ])
plot(y2o,'m')
set(gca,'XColor','green','YColor','cyan','Color','black','FontSize',10);
msgbox('ELABORAZIONE DATI COMPLETATA...');
[beep,f]=wavread('Beep.wav');
wavplay(beep,f)

% Disegna il grafico Densita' Spettrale di Potenza del canale audio corrispondente
- Emiliano Vezzoli - vezzoli@libero.it
y1=y(:,1);    % canale sinistro originale
y2=y(:,2);    % canale destro originale
if channel==1 % canale sinistro originale
  SD_PLOTCMD = str2mat('plot(t,y1,"y");xlabel("Tempo in secondi");',...
    ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
      'delete(h(1:2));xlabel("Frequenza in hertz");'],...
    ['specgram(y);set(gca,"XTickLabel",[]);', ...
      'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
      'ylabel("Frequenza");']);
SD_DISP = 1;
specgram(y1);
title('Spettrogramma','Color','y','FontSize',14);
set(gca,'XColor','cyan','YColor','green','Color','black');
xlabel('Tempo (Secondi)');
ylabel('Frequenza (Hz)');
elseif channel==2 % canale destro originale
  SD_PLOTCMD = str2mat('plot(t,y2,"r");xlabel("Tempo in secondi");',...
    ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
      'delete(h(1:2));xlabel("Frequenza in hertz");'],...
    ['specgram(y);set(gca,"XTickLabel",[]);', ...
      'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
      'ylabel("Frequenza");']);
SD_DISP = 1;
specgram(y2);
title('Spettrogramma','Color','r','FontSize',14);
set(gca,'XColor','cyan','YColor','green','Color','black');
xlabel('Tempo (Secondi)');
ylabel('Frequenza (Hz)');
elseif channel==3 % canale sinistro con effetto
  SD_PLOTCMD = str2mat('plot(t,y1o,"g");xlabel("Tempo in secondi");',...
    ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
```

```
        'delete(h(1:2));xlabel("Frequenza in hertz");'],...
      ['specgram(y);set(gca,"XTickLabel",[]);', ...
        'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
        'ylabel("Frequenza");']);
SD_DISP = 1;
specgram(y1o);
title('Spettrogramma','Color','g','FontSize',14);
set(gca,'XColor','cyan','YColor','green','Color','black');
xlabel('Tempo (Secondi)');
ylabel('Frequenza (Hz)');
elseif channel==4 % canale destro con effetto
   SD_PLOTCMD = str2mat('plot(t,y2o,"m");xlabel("Tempo in secondi");',...
      ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
        'delete(h(1:2));xlabel("Frequenza in hertz");'],...
      ['specgram(y);set(gca,"XTickLabel",[]);', ...
        'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
        'ylabel("Frequenza");']);
SD_DISP = 1;
specgram(y2o);
title('Spettrogramma','Color','m','FontSize',14);
set(gca,'XColor','cyan','YColor','green','Color','black');
xlabel('Tempo (Secondi)');
ylabel('Frequenza (Hz)');
end
% Disegna il grafico temporale del canale audio corrispondente - Emiliano Vezzoli
- vezzoli@libero.it
y1=y(:,1);    % canale sinistro originale
y2=y(:,2);    % canale destro originale

if channel==1 % canale sinistro originale
   SD_PLOTCMD = str2mat('plot(t,y1,"y");xlabel("Tempo in secondi");',...
      ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
        'delete(h(1:2));xlabel("Frequenza in hertz");'],...
      ['specgram(y);set(gca,"XTickLabel",[]);', ...
        'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
        'ylabel("Frequenza");']);
SD_DISP = 1;
y1=y1/max(abs(y1));t=(0:length(y1)-1)/FS;
   set(gcf,'UserData',[t(:) y1(:)]);
   n = SD_DISP;
   eval([SD_PLOTCMD(n,:)]);
   title([int2str(length(y1)) ' Campioni'],'Color','y','FontSize',14);
   drawnow;
   set(gca,'XColor','cyan','YColor','green','Color','black');
elseif channel==2 % canale destro originale
   SD_PLOTCMD = str2mat('plot(t,y2,"r");xlabel("Tempo in secondi");',...
```

```
    ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
      'delete(h(1:2));xlabel("Frequenza in hertz");'],...
    ['specgram(y);set(gca,"XTickLabel",[]);', ...
      'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
      'ylabel("Frequenza");']);
SD_DISP = 1;
y2=y2/max(abs(y2));t=(0:length(y2)-1)/FS;
  set(gcf,'UserData',[t(:) y2(:)]);
  n = SD_DISP;
  eval([SD_PLOTCMD(n,:)]);
  title([int2str(length(y2)) ' Campioni'],'Color','y','FontSize',14);
  drawnow;
  set(gca,'XColor','cyan','YColor','green','Color','black');
elseif channel==3 % canale sinistro con effetto
  SD_PLOTCMD = str2mat('plot(t,y1o,"g");xlabel("Tempo in secondi");',...
    ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
      'delete(h(1:2));xlabel("Frequenza in hertz");'],...
    ['specgram(y);set(gca,"XTickLabel",[]);', ...
      'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
      'ylabel("Frequenza");']);
SD_DISP = 1;
y1o=y1o/max(abs(y1o));t=(0:length(y1o)-1)/FS;
  set(gcf,'UserData',[t(:) y1(:)]);
  n = SD_DISP;
  eval([SD_PLOTCMD(n,:)]);
  title([int2str(length(y1o)) ' Campioni'],'Color','y','FontSize',14);
  drawnow;
  set(gca,'XColor','cyan','YColor','green','Color','black');
elseif channel==4 % canale destro con effetto
  SD_PLOTCMD = str2mat('plot(t,y2o,"m");xlabel("Tempo in secondi");',...
    ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
      'delete(h(1:2));xlabel("Frequenza in hertz");'],...
    ['specgram(y);set(gca,"XTickLabel",[]);', ...
      'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
      'ylabel("Frequenza");']);
SD_DISP = 1;
y2o=y2o/max(abs(y2o));t=(0:length(y2o)-1)/FS;
  set(gcf,'UserData',[t(:) y2(:)]);
  n = SD_DISP;
  eval([SD_PLOTCMD(n,:)]);
  title([int2str(length(y2o)) ' Campioni'],'Color','y','FontSize',14);
  drawnow;
  set(gca,'XColor','cyan','YColor','green','Color','black');
end
```

```
% Disegna il grafico Densita' Spettrale di Potenza del canale audio corrispondente
- Emiliano Vezzoli - vezzoli@libero.it
y1=y(:,1);    % canale sinistro originale
y2=y(:,2);    % canale destro originale
if channel==1 % canale sinistro originale
   SD_PLOTCMD = str2mat('plot(t,y1,"y");xlabel("Tempo in secondi");',...
     ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
       'delete(h(1:2));xlabel("Frequenza in hertz");'],...
     ['specgram(y);set(gca,"XTickLabel",[]);', ...
       'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
       'ylabel("Frequenza");']);
SD_DISP = 1;
p1=spectrum(y1,1024);
   specplot(p1,FS);
   title('Densita" spettrale di potenza','Color','y','FontSize',14);
set(gca,'XColor','cyan','YColor','green','Color','black','Xlabel',text('String','Frequenz
a (Hz)'));
   h=get(gca,'children');
   delete(h(1:2));
elseif channel==2 % canale destro originale
   SD_PLOTCMD = str2mat('plot(t,y2,"r");xlabel("Tempo in secondi");',...
     ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
       'delete(h(1:2));xlabel("Frequenza in hertz");'],...
     ['specgram(y);set(gca,"XTickLabel",[]);', ...
       'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
       'ylabel("Frequenza");']);
SD_DISP = 1;
p2=spectrum(y2,1024);
   specplot(p2,FS);
   title('Densita" spettrale di potenza','Color','r','FontSize',14);
set(gca,'XColor','cyan','YColor','green','Color','black','Xlabel',text('String','Frequenz
a (Hz)'));
   h=get(gca,'children');
   delete(h(1:2));
elseif channel==3 % canale sinistro con effetto
   SD_PLOTCMD = str2mat('plot(t,y1o,"g");xlabel("Tempo in secondi");',...
     ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
       'delete(h(1:2));xlabel("Frequenza in hertz");'],...
     ['specgram(y);set(gca,"XTickLabel",[]);', ...
       'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
       'ylabel("Frequenza");']);
SD_DISP = 1;
p1o=spectrum(y1o,1024);
   specplot(p1o,FS);
   title('Densita" spettrale di potenza','Color','g','FontSize',14);
```

```
set(gca,'XColor','cyan','YColor','green','Color','black','Xlabel',text('String','Frequenza (Hz)'));
  h=get(gca,'children');
  delete(h(1:2));
elseif channel==4 % canale destro con effetto
  SD_PLOTCMD = str2mat('plot(t,y2o,"m");xlabel("Tempo in secondi");',...
    ['p=spectrum(y,1024);specplot(p,Fs);h=get(gca,"children");',...
      'delete(h(1:2));xlabel("Frequenza in hertz");'],...
    ['specgram(y);set(gca,"XTickLabel",[]);', ...
      'set(gca,"YTickLabel",[]);xlabel("Tempo");',...
      'ylabel("Frequenza");']);
SD_DISP = 1;
p2o=spectrum(y2o,1024);
  specplot(p2o,FS);
  title('Densita" spettrale di potenza','Color','m','FontSize',14);
set(gca,'XColor','cyan','YColor','green','Color','black','Xlabel',text('String','Frequenza (Hz)'));
  h=get(gca,'children');
  delete(h(1:2));
end
```