UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

CORSO DI LAUREA TRIENNALE IN INFORMATICA MUSICALE

IMPLEMENTAZIONE DI ALGORITMI DI WAVESHAPING NEL DOMINIO AUDIO

Relatore: Prof. Luca Andrea Ludovico
Correlatore: Prof. Giorgio Presti

Tesi di Laurea di:
Campanelli Federico
Matr. Nr. 810407

anno accademico 2016-2017

INDICE

1.1 Introduzione

Prima di iniziare bisogna definire il fenomeno di distorsione del suono:

Per distorsione del suono si intende l'alterazione della forma originale dell'onda.

Questo fenomeno può essere indesiderato in alcuni ambiti, poiché introduce un disturbo (come ad esempio quello delle telecomunicazioni), oppure voluto, perché si cerca un determinato tipo di suono (si pensi ad un distorsore per chitarra).

Bisogna però fare un'ulteriore precisazione, i ronzii e le interferenze, nonostante modifichino la forma d'onda, non sono considerati dei tipi di distorsione, poiché sono segnali che si sommano al segnale originale.

Il tipo di distorsione che si andrà ad analizzare è una distorsione non lineare che, a differenza di quella lineare, modifica l'onda in entrata, per creare un output diverso.

Il risultato finale sarà l'analisi di una distorsione esistente e la sua ricreazione, tutto questo verrà fatto grazie a due software:

- Matlab: software che che permette di analizzare e creare grafici partendo da funzioni matematiche inoltre permette di riprodurre i file audio che sono stati creati
- Juce: un software finalizzato alla creazione di diversi tipi di plugin digitali

Con Matlab si analizzeranno tutte le caratteristiche dell'onda distorta con un plugin della Waves, da ricreare. Inoltre verranno anche create delle distorsioni semplici in modo da comprendere il funzionamento di vari tipi di distorsori.

Con Juce si ricostruirà il plugin originale partendo dai risultati delle analisi fatte con Matlab.

La creazione di varie distorsioni è stata possibile grazie alla tecnica del waveshaping, che consiste nella distorsione dell'ampiezza di un suono allo scopo di alterarne la forma d'onda.

Utilizzando questa tecnica con cura è possibile ottenere timbri interessanti e gradevoli.

La decisione di affrontare questo studio per realizzare un plugin che permetta di distorcere il suono di una chitarra senza l'ausilio di un amplificatore è stata presa perché oggi è sempre in crescita il numero di musicisti che si affidano a plugin digitali, sia per una questione di comodità che per una questione di ricerca del suono, infatti, con una distorsione digitale, è possibile ricreare un suono molto fedele a quello di una distorsione reale, ma è anche possibile creare nuove sonorità, che non potrebbero essere riprodotte tramite l'uso di amplificatori reali.

Nel primo capitolo vedremo le varie definizioni dei concetti che stanno alla base della distorsione del suono e alcuni dei principali tipi di distorsione che sono stati

analizzati, nel secondo capitolo vedremo i software che servono per realizzare un progetto di questo tipo (dall'analisi all'implementazione).

Nel terzo capitolo vedremo nel dettaglio l'implementazione, le analisi ed i test che sono stati fatti per arrivare al risultato finale, mentre nel quarto capitolo mostreremo il funzionamento del software e delle idee per un possibile aggiornamento del software, al fine di migliorarne o ampliarne il funzionamento.

2.1 Storia della distorsione

La distorsione di un segnale è sempre stato considerato un problema, sopratutto nell'ambito degli studi di registrazione.

Molti fonici cercavano di evitare ogni tipo di interferenza o di guasto della strumentazione per non rovinare l'incisione delle tracce sui vinili, ma tutto questo cambiò nel 1961.

Durante una registrazione del singolo “Don't worry” di Marty Robbins al Quonset Hut Studio, un trasformatore nel mixer si danneggia e aggiunge alla traccia di basso una distorsione che, successivamente, verrà chiamata “Fuzz”.

Al posto che incidere nuovamente la traccia, i produttori e lo stesso Marty decisero di tenerla e cercarono anche di riprodurla nuovamente, poiché dava alla canzone un effetto che nessuno aveva mai sentito.

Dopo che venne messa in commercio Don't Worry, molti musicisti e produttori iniziarono a non vedere più un difetto nella distorsione, bensì un potenziale strumento innovativo per ottenere nuove sonorità e nuovi effetti, infatti l' ingegnere Glen Snoddy ed il produttore Don Law crearono un circuito a 3 transistor in grado di riprodurre il suono presente nella registrazione della canzone di Marty.

Il circuito creato da Snoddy e Law venne successivamente messo in commercio dalla Gibson, sotto il nome di “Maestro Fuzz Tone” e questo fu il primo pedale analogico di distorsione della storia.

1) Il Gibson Maestro Fuzz Tone

Precedentemente alcuni chitarristi cercavano di spingere al limite gli amplificatori, che al tempo erano costruiti per riprodurre un suono “pulito” di chitarra elettrica, per ottenere una leggera distorsione, chiamata overdrive, ma non si era mai sentito un

effetto di distorsione aggressivo come il fuzz, inoltre mandare spingere le valvole di un amplificatore fino alla saturazione rischiava di danneggiarlo e renderlo inutilizzabile.

La vera svolta che portò ad un grande inserimento della distorsione nel mondo della musica fu l'uscita del singolo “Satisfaction” dei Rolling Stones nel 1965, dopo questa canzone l'uso delle distorsori crebbe esponenzialmente e nacquero anche nuovi generi musicali, come il rock e, successivamente, il metal.

Al giorno d'oggi il distorsore è un accessorio che non può mancare tra l'equipaggiamento di un chitarrista.

Col passare degli anni varie case produttrici hanno creato pedali analogici e digitali di svariati tipi, rendendo possibile la creazione di diversi tipi di distorsioni, oramai largamente utilizzati in ogni genere musicale.

I distorsori analogici sono molto più costosi di quelli digitali, ma hanno una resa migliore

2) Il blacksmith di Paul Gagon

I distorsori digitali, pur essendo più a buon mercato, possono non avere la stessa resa di quelli analogici.

3) Un overdrive della Boss

Oggi esistono anche software in grado di simulare delle distorsioni reali e sono largamente utilizzati sia in studio di registrazione che nei live, ma nonostante questo

molti musicisti che ricercano ancora determinate sonorità, tendono a preferire dei pedali analogici, valorizzando più il suono rispetto alla comodità del digitale.

2.2 Funzione di trasferimento

Come è stato affermato in precedenza, una distorsione non lineare modifica la forma dell'onda. Questo significa che se in entrata si ha un'onda sinusoidale, in uscita non si avrà una sinusoide.

Nella teoria dei segnali (che analizza le proprietà matematiche e statistiche dei segnali, definiti come funzioni matematiche del tempo) un “sistema” esente da rumori può essere caratterizzato da una funzione di trasferimento, tale che l'uscita

y(t) possa essere scritta in funzione dell'ingresso x come:

$$Y(t) = F(x(t))$$

La funzione di trasferimento è una funzione che caratterizza il comportamento di un sistema dinamico tempo-invariante nel dominio della frequenza, mettendo in relazione l'ingresso e l'uscita.

Se la funzione di trasferimento non è più complessa della funzione originale non si avrà una distorsione e il segnale in uscita verrà definito come “fedele all'originale”.

Il segnale è fedele all'originale anche se nella funzione di trasferimento includiamo un guadagno “A” costante ed un ritardo “T”

$$Y(t) = A \cdot x(t-T)$$

2.3 Tipi di distorsione

Possono esserci vari tipi di distorsione:

- Distorsione Armonica:

La distorsione armonica si ha quando L'onda è affetta da componenti non lineari. Questo fa si che venga alterato il contenuto armonico del segnale

- Distorsione di ampiezza:

Una distorsione di ampiezza è una distorsione che si verifica nei sistemi, sottosistemi o dispositivi quando l'ampiezza d'uscita è una funzione non lineare dell'ampiezza d'ingresso

- Distorsione della risposta in frequenza:

Questo tipo di distorsione avviene per mezzo di filtri, quando frequenze diverse sono amplificate con coefficienti diversi. Nel caso particolare dell'audio, questa distorsione è principalmente causata da vari fattori, come l'acustica dell'ambiente, l'utilizzo di microfoni o altoparlanti scadenti e può anche verificarsi se si utilizzano cavi troppo lunghi per gli altoparlanti.

- Distorsione di fase:

In questa forma di distorsione tutti i componenti di un segnale d'ingresso non sono amplificati con la stessa fase, portando il segnale di uscita ad essere fuori fase con il

resto dell'uscita.

La distorsione di fase non è percepita dall'orecchio umano.

2.4 Effetti e prevenzione della distorsione

Gli effetti della distorsione di un suono possono essere molteplici, si può verificare l'introduzione di rumori o interferenze, una risposta in frequenza non piatta del canale trasmissivo, delle alterazioni delle informazioni associate al segnale, che portano ad un degrado della qualità di quest'ultimo.

Nella musica la distorsione viene utilizzata di proposito per ottenere degli effetti e dei suoni ben specifici, mentre nel campo delle telecomunicazioni è solo un elemento di disturbo. Per questo vengono utilizzati dei metodi di correzione della distorsione, come ad esempio l'uso di filtri lineari che, riproducendo un effetto inverso alla distorsione originale, lascia il sistema intatto.

La correzione non è sempre possibile, infatti se la funzione inversa non esiste, per esempio se la funzione di trasferimento ha delle zone piatte (la funzione inversa avrà molti punti d'ingresso per ottenere un unico punto d'uscita), il risultato rappresenta una perdita d''informazione, quindi incorreggibile.

Questo si verifica, ad esempio, quando un amplificatore è in stato di clipping, o quando si verifica una distorsione dovuta allo slow rate (velocità di risposta), quando l'uscita dipende dall'amplificatore e non dal segnale d'ingresso.

Per annullare una distorsione di un segnale è anche possibile usare a valle del sistema distorcente un equalizzatore che modifichi i toni in frequenza dell'onda ricevuta, oppure utilizzare a monte del sistema distorcente e conoscendo le caratteristiche della distorsione, un predistorsore con distorsione reciproca al sistema distorcente in cascata, in modo tale che in uscita alla doppia distorsione il segnale rimanga fedele a quello di ingresso.

2.5 Waveshaping

Il waveshaping consiste nel prendere i valori di un segnale in ingresso X in un determinato istante di tempo e applicare una funzione del tipo:

$$y = f(x)$$

Questa funzione che prende l'input X e crea l'output Y è detta "funzione di trasferimento".

La funzione f(x) può essere una funzione qualsiasi, infatti le possibilità sono potenzialmente infinite, poiché la forma d'onda può essere alterata in qualsiasi modo.

Il waveshaping è una modalità di approccio alla sintesi del suono mediante distorsione che consente di ottenere spettri le cui componenti si evolvono

dinamicamente nel tempo. Dato che la forma d'onda viene modificata, il waveshaping è considerato un processo non lineare.

La tecnica del waveshaping consente il controllo continuo del segnale tramite un indice, rendendo possibile la creazione di spettri dinamici mediante la variazione nel tempo dell'indice stesso.

L'output del waveshaper cambia al variare dell'ampiezza dell'onda in ingresso, questo perché cambieranno le ampiezze delle componenti dello spettro risultante.

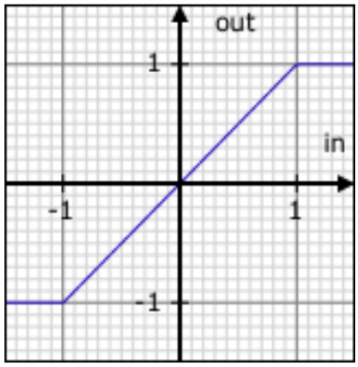
Il waveshaper è caratterizzato dalla sua funzione di trasferimento, che può essere rappresentata graficamente, con l'ampiezza del segnale d'ingresso sull'asse delle ascisse e quella del segnale d'uscita sull'asse delle ordinate.

Con il waveshaping siamo in grado di aggiungere frequenze (armoniche) non presenti nel segnale originale, questo perché l'output sarà una somma di sinusoidi con frequenze intere multiple della funzione originale.

Questo discorso si rifà al teorema di Fourier in cui si rappresenta una funzione periodica mediante una combinazione lineare di funzioni sinusoidali:

$$y(t) = a_0 + a_1 \cos(\omega_1 t) + a_2 \cos(\omega_2 t) + \dots a_n \cos(\omega_n t)$$

Ogni termine della sommatoria rappresenta un'armonica, mentre il primo termine è la frequenza fondamentale.



(a) hard clipper

4) output dell'hard clipper

Se prendiamo come esempio l'hard clipping (che vedremo successivamente nel dettaglio) se inseriamo come input una forma d'onda sinusoidale avremo in output un'onda quadra che, secondo l'analisi di Fourier, aggiunge solo armoniche di ordine dispari all'onda.

Dalla forma della funzione possiamo già dedurre alcune caratteristiche del suono prodotto dal waveshaper ancora prima di ascoltarlo, ad esempio se è una retta, non avremo distorsione, mentre tutte le forme che sono diverse dalla retta producono una distorsione del suono, ad esempio se prendiamo in considerazione la funzione

$y = x^2$ noteremo che le frequenze con ampiezza più elevata saranno più accentuate

rispetto a quelle di ampiezza bassa.

Altre funzioni di trasferimento sono le funzioni polinomiali.

Una funzione polinomiale di ordine *n* è definita come la sommatoria di tutti i *k* elementi $a_n x^n$

con *k* che va da 0 a *n*:

$$f(x) = anx^n + a(n-1)x^{(x-1)} + \ldots + a_2x^2 + a_1x + a_0$$

Con questa funzione di trasferimento si riescono a manipolare le onde in maniera più precisa, inoltre utilizzando un polinomio di ordine *n* creeremo solamente armoniche di quell'ordine.

L'unico difetto di questo waveshaping è che il polinomio tenderà a $+\infty$ o a $-\infty$, quindi, in un caso reale, bisognerà limitare l'ampiezza dell'onda in ingresso.

Usando quindi, una funzione di trasferimento di questo tipo avremo come risultato un'onda che contiene tutti i multipli interi della frequenza di input, fino ad arrivare al massimo (*n*) che è il grado del polinomio della funzione.

2.6 Funzioni pari e dispari

Esistono due tipi principali di funzioni; le funzioni pari e quelle dispari.

Le funzioni pari sono funzioni che, su un grafico cartesiano, sono simmetriche rispetto all'asse delle ordinate, mentre le funzioni dispari sono simmetriche rispetto all'origine.

Se utilizziamo una funzione di trasferimento pari avremo come risultato un'onda che conterrà solo armoniche pari, al contrario se utilizziamo una funzione dispari le armoniche contenute nell'onda in uscita saranno dispari.

2.7 Intermodulzioni e powerchords

Se proviamo a inserire una somma di sinusoidi in entrata al posto che una sola, otterremo un risultato particolare, infatti, l'output non solo conterrà le armoniche di tutte le sinusoidi che formano l'onda, ma anche la loro differenza.

Prendiamo l'esempio più semplice, in cui abbiamo una somma di due sinusoidi in ingresso e una funzione di trasferimento del tipo $y = x^2$ inoltre le due sinusoidi avranno frequenze ω_1 e ω_2 e ampiezze A_1 e A_2 :

$$x(t) = A1 \cdot sin(\omega_1 t) + A2 \cdot sin(\omega_2 t)$$

Quindi, applicando la funzione di trasferimento avremo:

$$y(t) = (x(t))^2 = A_1^2 \cdot \sin2(\omega 1t) + A_2^2 \cdot \sin2(\omega 2t) + 2A_1A_2 \sin(\omega 1t) \cdot \sin(\omega 2t)$$

Notare che questa è una comunissima risoluzione di un prodotto notevole del tipo $(x+y)^2$.

I primi due elementi della funzione in uscita sono i normali risultati della potenza

applicata sulle due onde in ingresso, ma il terzo elemento (il doppio prodotto dei primi due termini) è un nuovo fenomeno.

Per capire quali sono le frequenze che vengono create da questo nuovo elemento bisogna applicare un'identità trigonometrica:

$$sin(\alpha) \cdot sin(\beta) = ½ (cos(\alpha-\beta) - cos(\alpha+\beta))$$

Quindi, applicandola al risultato precedente avremo:

$$y(t)= A^2_1/2-A^2_1/2\cos(2\omega_1 t) + A^2_2/2-A^2_2/2\cos(2\omega_2 t) + A_1A_2\cos(\omega_1-\omega_2)t-A_1A_2\cos(\omega_1+\omega_2)t$$

$(\omega_1-\omega_2)$ e $(\omega_1-\omega_2)$ sono le somme e le differenze delle frequenze delle due onde che vengono sommate.

Questa è una situazione che si verifica per la somma di sinusoidi, ma con la somma di più sinusoidi si verificheranno fenomeni più complessi e si creeranno sempre più elementi nell'equazione in uscita.

Se dovessimo inserire in un waveshaper un suono con la prima, la terza e la quinta armonica (come ad esempio un accordo) graficamente potremmo vedere uno spettro dell'onda praticamente incomprensibile, causato dalle differenze e dalle somme dei vari toni.

Dal punto di vista uditivo, se in input abbiamo solo la fondamentale e la quinta armonica il risultato è ancora piacevole, ed è per questo che molti chitarristi preferiscono non suonare accordi completi mentre la loro chitarra un è collegata ad un distorsore, ma tendono a suonare i powerchords, ovvero bicordi (accordo composto da due note suonate simultaneamente) in cui suonano la nota fondamentale e la quinta (alcuni ci aggiungono l'ottava, che però essendo al doppio della frequenza della nota fondamentale non crea dissonanze).

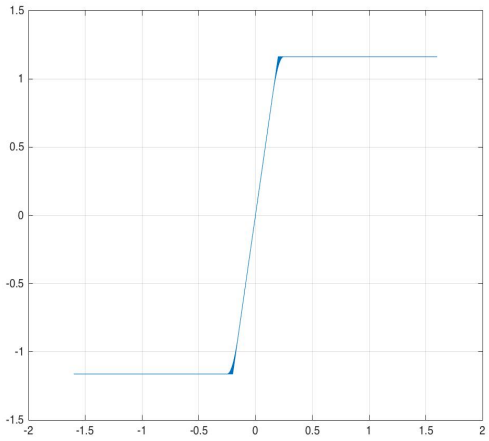
2.8 Tipi di distorsione

Esistono vari tipi di distorsione, infatti un waveshaper può modellare l'onda in ingresso in infiniti modi, ognuno caratterizzato da una formula di trasferimento. Visto che non è possibile rappresentarli tutti quanti, qui verranno descritti alcuni tra quelli principali.

- Hard clipping

L'hard clipping è un tipo di distorsione che si verifica quando un amplificatore tenta di erogare una tensione maggiore della sua capacità massima, ovvero va in saturazione.

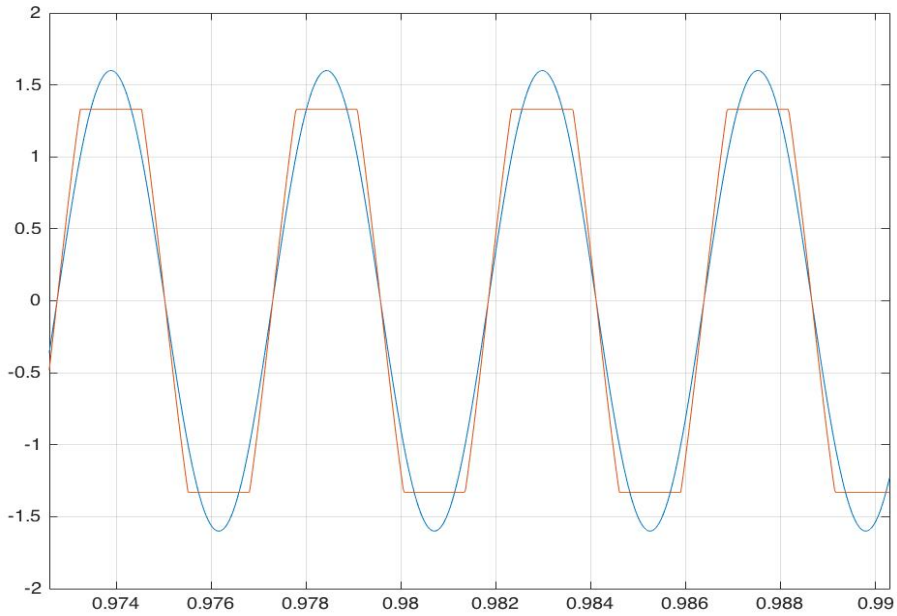
L'hard clipper è anche noto col nome di “overdrive” tra i chitarristi.



5) Funzione di trasferimento del clipper

Come si può vedere dall'immagine, il segnale verrà amplificato fino ad una soglia, dopodiché non potrà più essere amplificato ulteriormente.

Al “taglio” il segnale viene definito in clipping e il segnale extra che va oltre la soglia viene semplicemente tagliato e la forma d'onda risulta simile ad un'onda quadra.



6) Confronto tra sinusoide pura e clipper

Questo tipo di distorsione viene utilizzata intenzionalmente dai chitarristi per ottenere un suono specifico di chitarra.

Nel dominio delle frequenze, il clipping produce armoniche a frequenze più elevate rispetto ai segnali non tagliati, inoltre produce anche una compressione in ampiezza, dove le note più alte in volume sono tagliate a livelli di quelle con volume più basso.

In amplificatori reali il clipping potrebbe, a causa del surriscaldamento dei suoi componenti, danneggiare l'amplificatore stesso.

Il clipping digitale, invece, si riproduce definendo una soglia oltre la quale il segnale verrà tagliato.

Questa soglia simula la capacità massima di erogazione di corrente di un amplificatore.

- Fuzz

Il fuzz è un tipo di distorsione che sfrutta la saturazione del segnale, ma in maniera molto maggiore rispetto all'overdrive. Vengono quindi aggiunte numerose armoniche allo spettro e si verificano fenomeni di intermodulazione.

In un fuzz sono presenti due componenti principali: un amplificatore ed un circuito di clipping.

Il principio di funzionamento degli effetti di distorsione è sempre lo stesso: si amplifica il segnale di ingresso, e si esegue su di esso il “clipping”: da un certo livello in poi, il segnale viene “tagliato” (lo stesso concetto di un hard clipper).

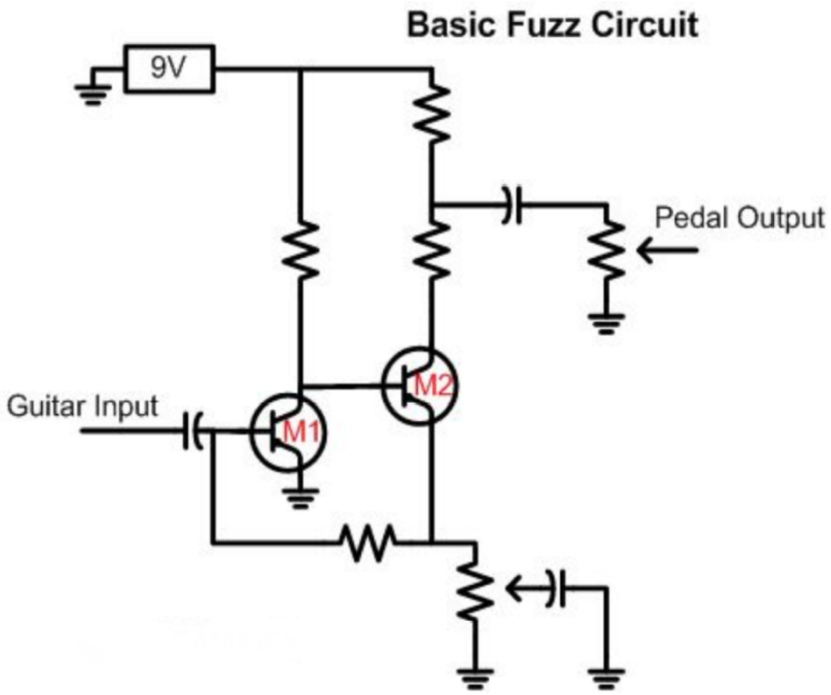
Il *fuzzbox* (nome del pedale del fuzz) amplifica l'onda e smorza il segnale portandolo dall'onda sinusoidale originaria a un'onda in uscita molto simile all'onda quadra, inoltre taglia anche molte frequenze medie rispetto ad altri distorsori.

Nel fuzz, lo scopo è ottenuto con l’impiego di poche parti, di cui 2 o 3, a seconda del tipo, sono transistor.

Proprio i transistor sono generalmente riconosciuti come gli elementi “critici” del fuzz, ovvero come componenti che influiscono pesantemente sul suono e sulle sue caratteristiche.

Negli anni 60 venivano utilizzati transistor al germanio, oggi invece, per la loro maggiore affidabilità ed il loro costo inferiore, vengono usati transistor al silicio.

Oggi vengono usati fuzz con componenti in germano solo se si vuole ricercare un particolare suono “vintage”, l'unico difetto è l'affidabilità dei componenti in germano, molto meno resistenti di quelli in silicio.



7) circuito elettronico di un fuzz

Qui viene riportato il circuito di un fuzz (un circuito molto semplice).

M1 e M2 sono i due transistor del circuito e, nonostante possano essere simili, presentano caratteristiche differenti che danno un “carattere” diverso al suono di ogni fuzz, sia esso composto da transistor in germanio o in silicio.

L'input passa in M1 e successivamente in M2 (i transistor sono in “cascata”), questo è essenzialmente un amplificatore con molto gain, ma col segnale in input più alto rispetto al suo output.

L'output di M2 viene quindi rimandato a M1 (feedback negativo), ma in questo caso il segnale avrà un livello maggiore grazie al gain e, nel momento in cui il volume della chitarra raggiunge un certo livello, il segnale verrà distorto.

La distorsione è causata da tutti e due i transistor ed è una distorsione non lineare (forma d'onda in input diversa da quella in output) e più gain c'è più il segnale sarà distorto.

Le proprietà dei singoli transistor incidono molto sulle caratteristiche del segnale in uscita, perché il fuzz è composto solo da due transistor, quindi il suono potrebbe cambiare, anche drasticamente, cambiando le proprietà di uno solo dei transistor del circuito.

- Bitcrusher

Il bitcrusher è un effetto digitale lo-fi (low fidelty) che produce una distorsione riducendo la lunghezza di banda del segnale.

Il prodotto di questa distorsione è il rumore di quantizzazione.

La quantizzazione è un'operazione che si fa dopo il campionamento.

Il segnale, una volta campionato, è costituito da una successione di impulsi che si presentano a determinati intervalli di tempo e la loro codifica si ottiene mediante la quantizzazione nel discreto dei valori reali degli impulsi-campione (Pulse Code Modulation).

Prendiamo il caso della quantizzazione lineare, per ottenerla dobbiamo:

1) Dividere la banda d'ampiezza del segnale f(t) in intervalli aventi larghezza fissa ∂
2) Associare un'unica parola di codifica in corrispondenza con tutti i valori (reali) di un particolare intervallo
3) Ripetere queste due operazioni per ogni intervallo della banda d'ampiezza

Ora, associamo una codifica binaria di parole di 3 bit per una quantizzazione ad 8 intervalli della banda di ampiezza:

$q = c_1 c_2 c_3$

è una parola che rappresenta il numero

$c_1 \times 2^2 + c_2 \times 2^1 + c_3 \times 2^0$

Questo numero è compreso tra 0 e 7 (8 intervalli - 1), è associato ad uno specifico intervallo e c_1, c_2, e c_3 sono cifre binarie.

Esistono due tipi di codifiche PCM binaria:

- Unipolare: un impulso per la cifra 1 e nessuno per la cifra 0
- Polare: un impulso positivo per la cifra 1 e un impulso negativo per la cifra 0

nel caso della codifica PCM binaria si hanno 2^k livelli di quantizzazione con una parola di codifica costituita da K cifre binarie, mentre se avessimo una codifica N-aria avremmo N^k livelli di quantizzazione , con parole costituite da K cifre espresse nel sistema di numerazione N-aria.

La codifica più diffusa è la PCM binaria, poiché introduce una minor quantità di rumore, ma nel caso della distorsione del suono potrebbe essere interessante utilizzare codifiche N-arie, perché la creazione del rumore di quantizzazione è fondamentale in questo caso.

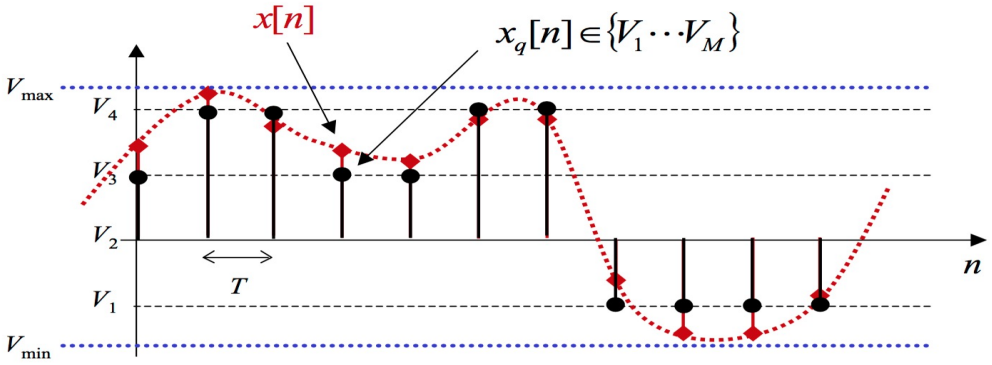
Definiamo ora un campione generico *x[n]*, questo è un numero reale che può assumere un qualsiasi valore compreso in un certo intervallo di ampiezze (Vmin, Vmax).

Durante la quantizzazione, il quantizzatore riceve un insieme di valori reali *x[n]* ed assegna, ad ognuno di essi, il valore più vicino a *x*[*n*] fra gli *M* possibili livelli di quantizzazione $V_1 ... V_M$.

Il valore quantizzato lo definiremo come $x_q[n]$.

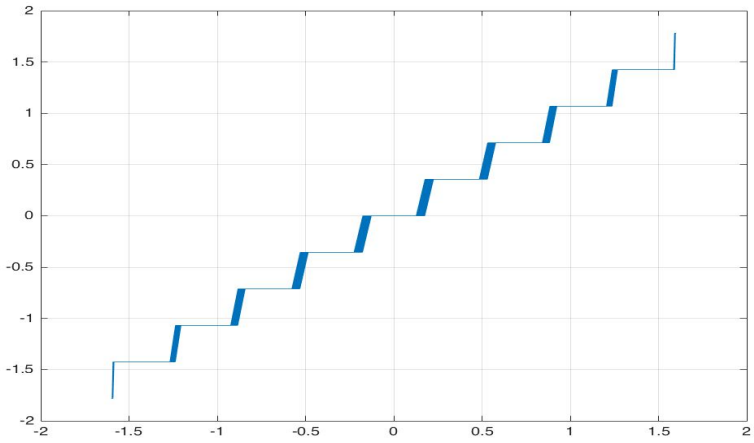
Più sono gli *M* livelli di quantizzazione, più ci sarà una riproduzione precisa del

segnale, infatti se si usano molti livelli (ad esempio 200) il suono in uscita sarà molto simile al suono in ingresso, mentre usando pochi livelli (ad esempio 2) il suono in uscita risulterà molto più distorto e per niente fedele all'originale.



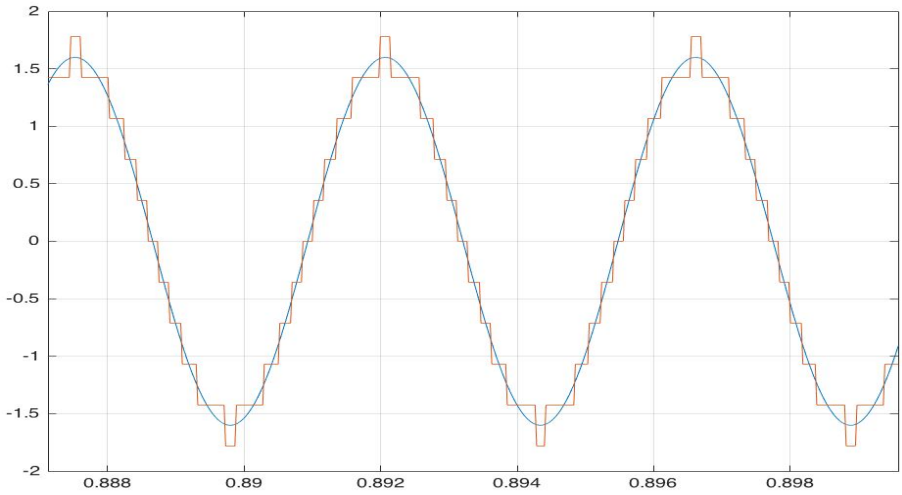
8) schema del funzionamento di un quantizzatore

Si può notare che la funzione di trasferimento di un bitcrusher è simile a degli "scalini".



9) Funzione di trasferimento del bitcrusher

Vediamo graficamente qual'è la differenza tra il segnale originale ed il segnale quantizzato:



10) confronto tra la sinusoide in ingresso e l'uscita del bitcrusher

Il segnale colorato in blu rappresenta la sinusoide in ingresso, mentre l'onda quadra è l'uscita del bitcrusher.

Il rumore di quantizzazione $e_q[n]$ è dato dalla differenza tra il campione $x[n]$ ed il suo valore assegnatoli dal quantizzatore $x_q[n]$:

$$e_q[n] = x[n] - x_q[n]$$

Con questo tipo di distorsione, più si riduce la qualità del segnale, più si produce rumore di quantizzazione, quindi si diminuiscono le informazioni processate e si introduce distorsione.

- X^n

La distorsione X^n è una distorsione particolare, pur essendo semplice da realizzare, infatti basta elevare ogni valore della sinusoide all'ennesima potenza

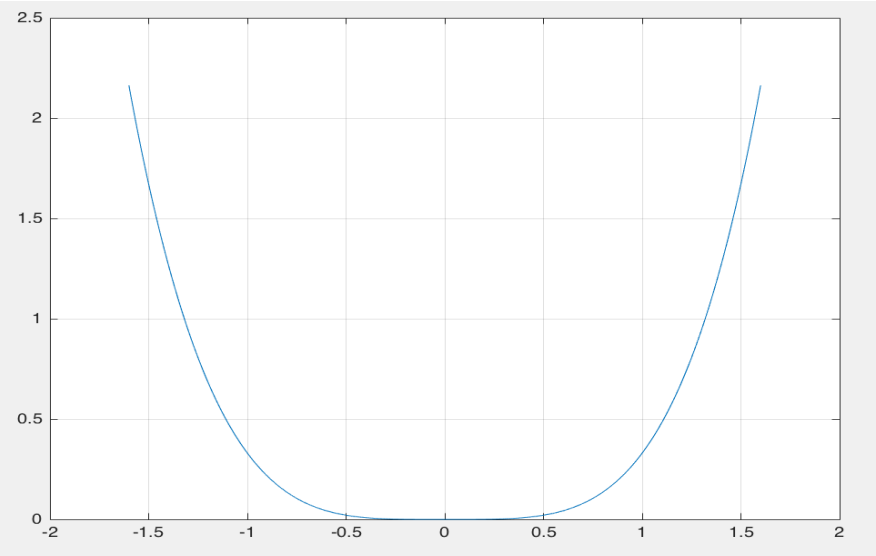
La sua formula è:

$$y = x^k$$

La particolarità di questa distorsione è che, a seconda dell'esponente, la funzione può essere pari o dispari, quindi cambiano le armoniche che vengono aggiunte e, di conseguenza, cambia anche il tipo di suono.

Infatti, se il nostro esponente sarà un numero pari, avremo una funzione pari, simmetrica rispetto all'asse delle ordinate.

Se il nostro esponente sarà un numero dispari avremo una funzione dispari



11) esempio di funzione $Y = X^4$

2.8 Distorsori digitali esistenti

Oggi ci sono diverse aziende che creano software digitali che permettono di modificare il suono di un qualsiasi strumento, partendo da pedali digitali che mirano ad aggiungere effetti (come riverbero, chorus, delay, phaser etc...) o a distorcere il suono in ingresso, in maniera più o meno significativa (overdrive, bitcrusher, fuzz etc...).

La Fractal ha creato una testata digitale molto diffusa tra i chitarristi ai giorni d'oggi, l'axe fx, un simulatore di testate/casse digitale molto pratico che permette di simulare molto fedelmente varie coppie testata-cassa in modo da permettere ai chitarristi di avere a disposizione migliaia di suoni diversi, in modo da adattarsi ad ogni tipo di genere.

12) Un axe fx della Fractal

La Positivegrid ha creato BIAS, un software che ha le stesse funzioni dell'Axe fx e che necessita solo un pc per poter funzionare.

Altre compagnie hanno creato software altrettanto potenti, ma più leggeri e meno conosciuti nell'ambito commerciale, basti pensare alle librerie della Waves, o al più conosciuto guitar rig della native instrument.

13) l'interfaccia di guitar rig

L'uso di questi software è sempre più diffuso (live ed in studio), sia per una questione di praticità, sia per una questione economica, infatti queste testate digitali e questi programmi offrono agli utilizzatori centinaia di possibilità per un costo che non supera mai i 2500 euro.

Grazie alle infinite possibilità che offre il waveshaping è anche possibile creare digitalmente degli effetti che, con l'ausilio di pedali e testate analogiche, sono impossibili da riprodurre (ci sono addirittura effetti digitali che permettono di cambiare radicalmente il suono di una chitarra, trasformandolo in un suono di un sintetizzatore).

In questo capitolo vedremo verranno mostrati i software utilizzati per arrivare allo sviluppo dell'applicativo partendo dall'analisi di segnali audio. Inoltre verrà descritto brevemente il ruolo di ognuno di essi.

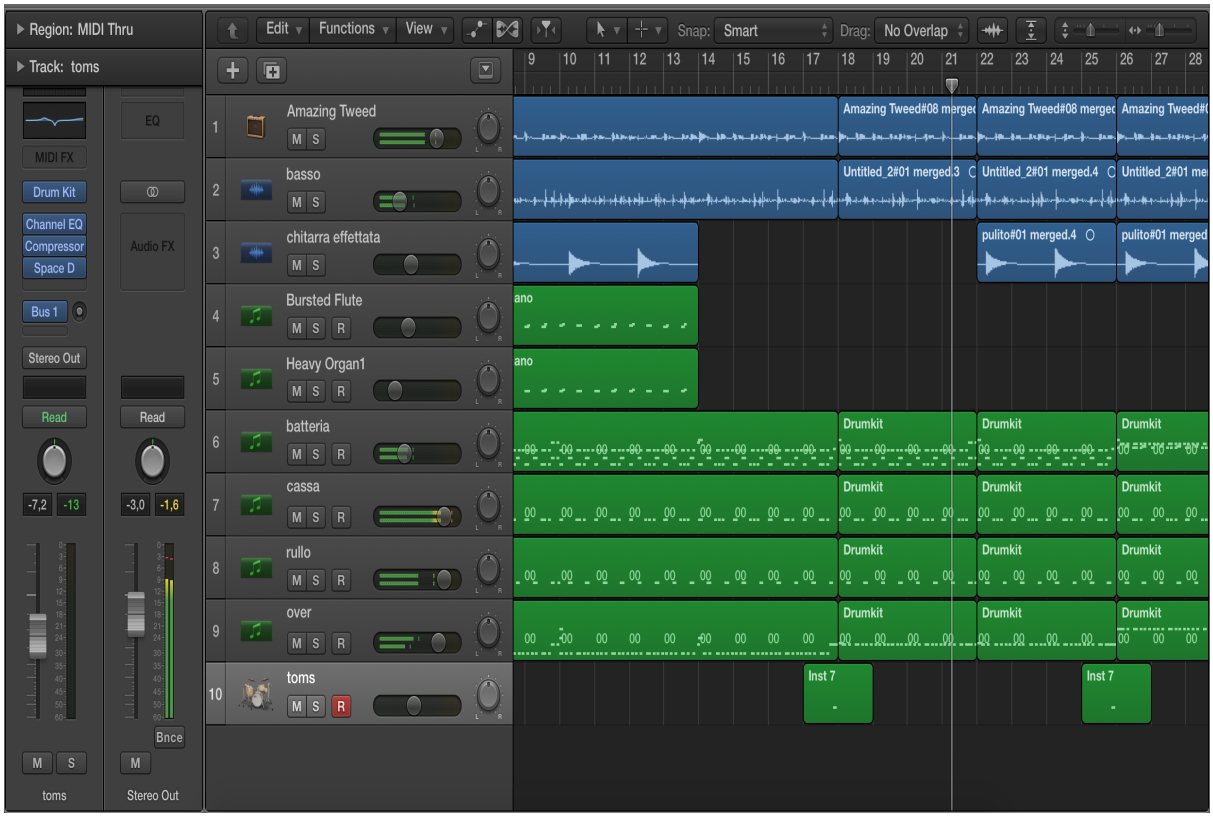
3.1 Logic Pro

Logic pro è un software utilizzato per registrare tracce audio, in modo da permettere l'incisione di un brano in multitraccia.

È stato prodotto dalla casa tedesca Emagic fino alla versione 5 ed in seguito acquisito da Apple Computer che ne ha portato avanti lo sviluppo solo per i computer Mac.

Si tratta di uno dei software maggiormente utilizzati negli studi di registrazione, insieme a Cubase e Pro tools (anch'essi software finalizzati alla registrazione di tracce audio) e proprio per questo motivo su Logic si può trovare molta documentazione online.

Con questo software è stato possibile applicare una distorsione digitale alla sinusoide creata in matlab, in modo da poter avere un suono da studiare per ricreare la distorsione. Oltre a questo è stata registrata una nota di chitarra a cui verrà applicata la distorsione ricreata.



14) l'interfaccia di logic pro

3.2 Matlab

Matlab è un ambiente creato dalla Mathworks per il calcolo numerico e l'analisi statistica.

È stato scritto in C e comprende anche l'omonimo linguaggio di programmazione creato dalla sua casa produttrice.

È utilizzato per l'apprendimento automatico, l'elaborazione di segnali ed immagini, la visione artificiale, le comunicazioni, la finanza computazionale, la progettazione di controllo, la robotica e molto altro.

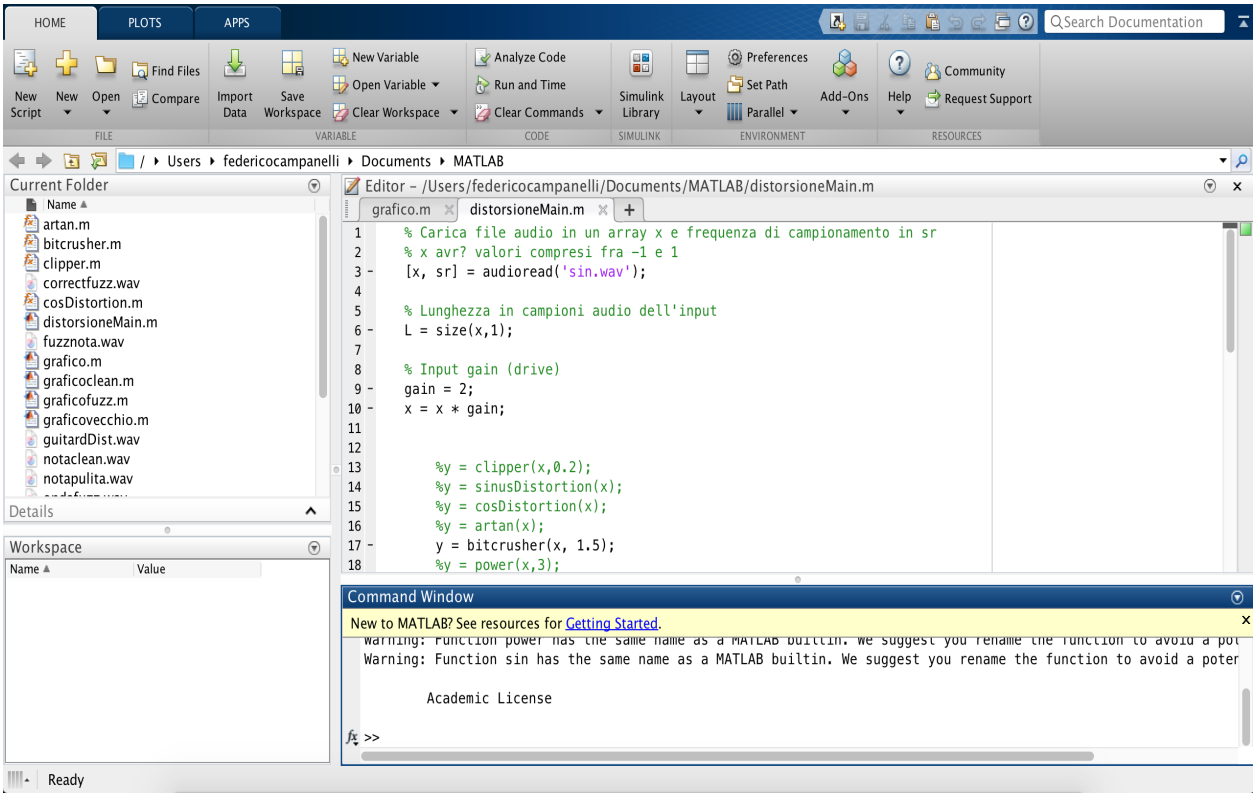
Molti scienziati usano questo software per progettare vari sistemi, come i sistemi di sicurezza delle auto o i dispositivi di monitoraggio dello stato di salute.

È un linguaggio che funziona su una base matematica, permette infatti di manipolare in maniera semplice formule e grafici, sia tramite codice, sia tramite un'interfaccia grafica.

Con questo software è stato possibile creare e studiare vari tipi di distorsione, analizzando spettri e forme d'onda.

Un altro vantaggio di Matlab, al contrario di software come Logic, è che funziona su diversi sistemi operativi.

Oggi è usato da milioni di persone nell'industria e nelle università per via dei suoi numerosi strumenti a supporto dei più disparati campi di studio e, proprio grazie a questa sua espansione, online si trova molta documentazione, il che rende comodo l'utilizzo di questo software per utenti poco esperti.



15) l'interfaccia di Matlab

3.3 Juce

Juce è un software multipiattaforma finalizzato alla creazione di plugin digitali (sia a 32 che a 64 bit), creato da un gruppo indipendente di sviluppatori.

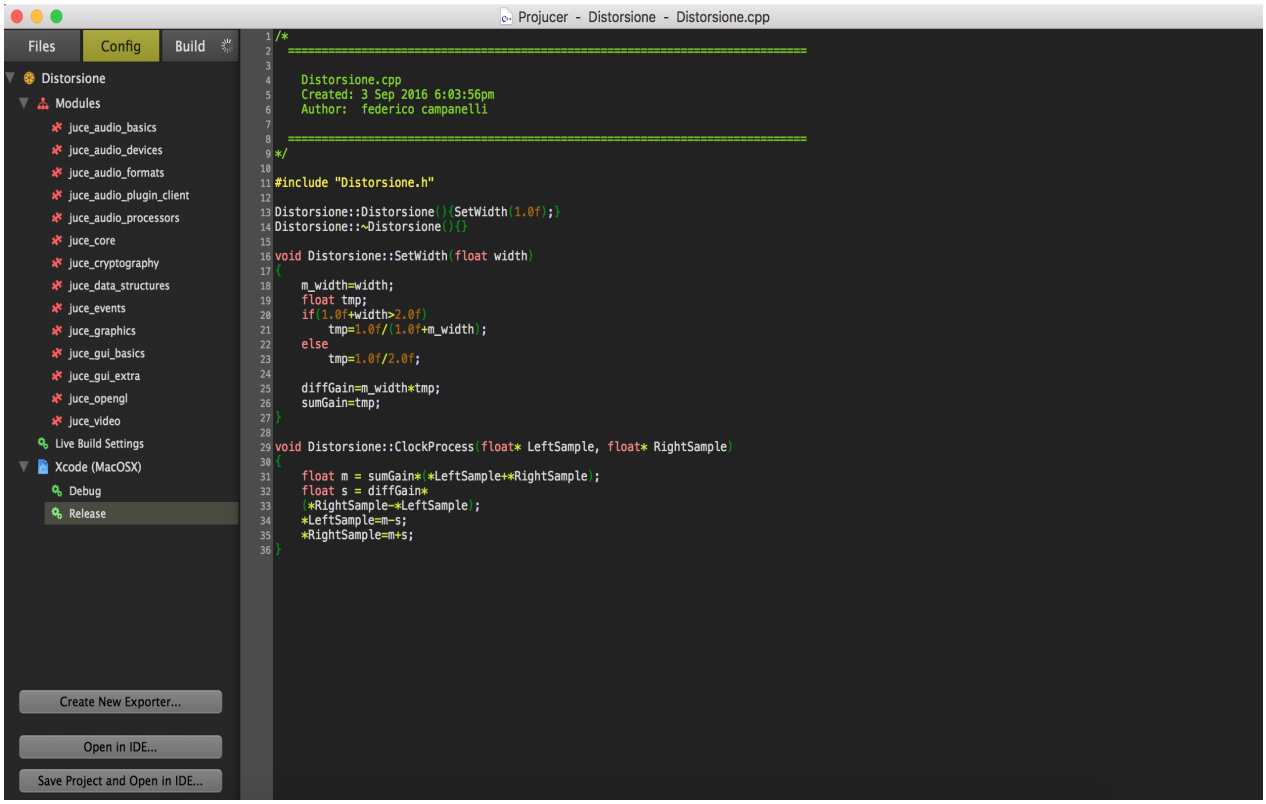
Juce nasce come libreria per C++, successivamente, dopo che Roli ha acquistato Juce, è stato introdotto un IDE (Projucer)

Per quanto riguarda la parte di codice, Juce ha bisogno di un ambiente di lavoro per poter compilare, infatti per poterlo utilizzare bisogna anche installare anche Visual studio (se si usa un sistema windows) o Xcode (se si usa un sistema Apple).

Al contrario dei due programmi visti in precedenza, Juce è free, quindi è accessibile a tutti (per Logic e Matlab si possono avere costi d'acquisto che superano i 200 euro), ma nonostante questo non è un software molto diffuso, poiché è complesso da usare e richiede competenze avanzate in ambito di programmazione e di conoscenza dei segnali audio.

Proprio per questo motivo non c'è molta documentazione online, si possono trovare solo un paio di tutorial sul sito della Redwood audio e qualche indicazione sul sito ufficiale di Juce.

Nonostante non sia molto diffuso tra i singoli sviluppatori, l'utenza di Juce comprende alcune tra le maggiori case sviluppatrici di software musicali, come la Korg e la M-Audio.



16) Interfaccia di Juce

4.1 Acquisizione dei suoni

In questo capitolo verrà trattata la parte di modellazione, ovvero l'analisi del segnale distorto in ogni sua caratteristica e la sua riproduzione, partendo da un segnale non distorto.
In questa fase sono stati utilizzati Logic e Matlab.
Per prima cosa è stata registrata una nota di chitarra (un la a 220 Hz), da cui sono stati esportate due tracce audio:

- Il la in dry: Questa traccia audio è il la prodotto dalla chitarra senza l'aggiunta di effetti, registrata direttamente con la chitarra inserita nella scheda audio, senza passare per pedali o amplificatori
- il la distorto: Questa traccia audio, invece, è il la in dry con l'aggiunta dei una distorsione, più precisamente un fuzz digitale della Waves

Entrambe le tracce audio sono state esportate in mono.
È stato fondamentale avere due tracce audio diverse che di base però, avevano la stessa nota (il la in dry), altrimenti un confronto dettagliato non sarebbe stato possibile.

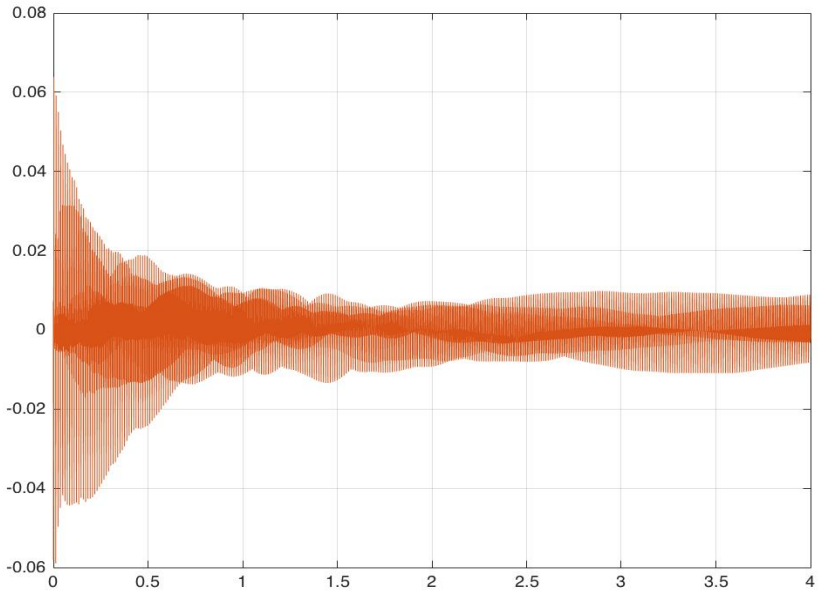
17) Il fuzz utilizzato

Lo studio per arrivare a riprodurre la distorsione è stato fatto su una sinusoide pura a cui è stato applicato il fuzz digitale.
Anche in questo caso bisogna avere due suoni:

- Una sinusoide pura: un segnale creato in Matlab, più precisamente un la a 440 Hz, della durata di tre secondi
- Una sinusoide distorta: questo segnale non è altro che la sinusoide pura a cui è stato applicato il fuzz

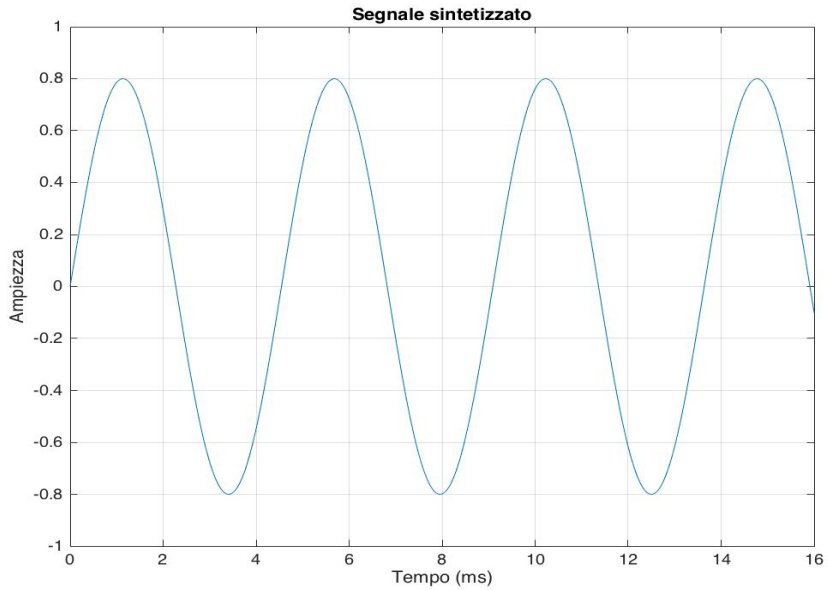
L'analisi per la riproduzione della distorsione originale è stata fatta sulle sinusoidi per poi essere applicata alla nota in dry registrata con la chitarra, questo perché le

forme d'onda delle note di chitarra sono molto complesse e questo portava a delle difficoltà nelle analisi dello spettro e delle funzioni di trasferimento.
Qui si può vedere la forma d'onda del la a 220 Hz registrato con la chitarra



18) Forma d'onda della nota di chitarra in dry

Qui, invece, è rappresentata la sinusoide pura creata con Matlab



19) Forma d'onda della sinusoide a 220 Hz

4.2 Preparazione dei suoni

Come prima cosa bisogna caricare nello script di Matlab le due onde, la sinusoide pura e quella distorta:

```
[clean, Fs] = audioread('sin.wav');
[fuzz, ~] = audioread('ondafuzz.wav');
```

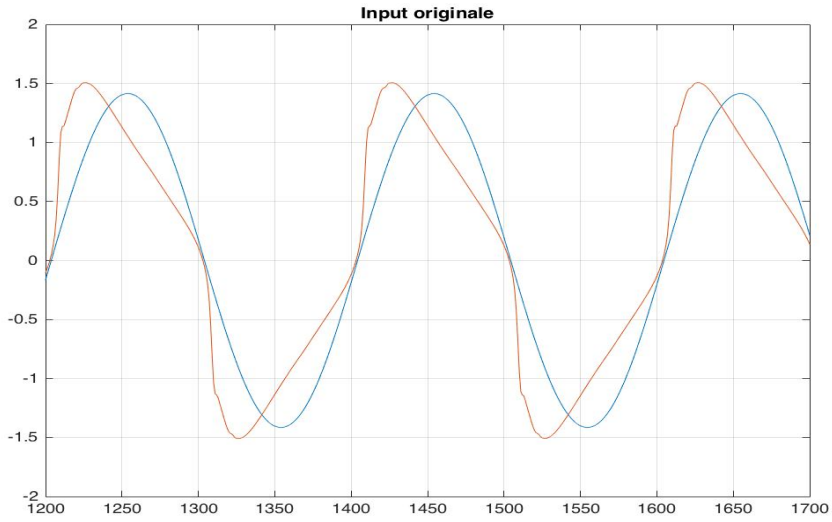
Dopodiché si portano entrambe le onde allo stesso livello, per far si che vengano confrontate nella stessa scala:

```
clean = clean./rms(clean);
fuzz = fuzz./rms(fuzz);
```

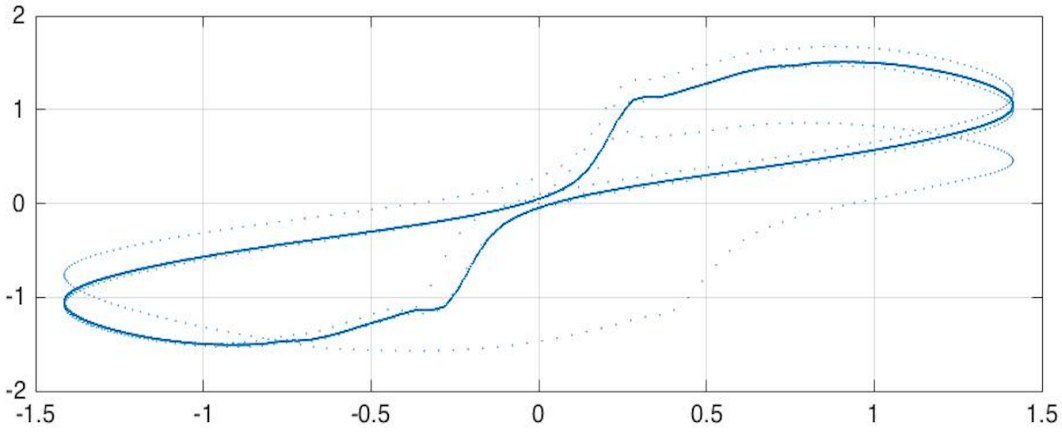
Questo viene fatto grazie all' RMS(root mean square), ovvero la radice quadrata della media dei valori del segnale elevati alla seconda potenza, che serve a calcolare la potenza del segnale.
Ad esempio se abbiamo n valori (x_1, x_2...x_n) avremo:

$$x_{\text{rms}} = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)}.$$

Dopodiché si può confrontare la sinusoide pura con quella distorta e si nota che è presente una distorsione di fase.
Questo tipo di distorsione può essere vista solo tramite una visualizzazione grafica, poiché la distorsione di fase non è udibile.



20) Grafico del confronto tra la sinusoide pura (blu) e quella distorta (rosso)



21) Funzione di trasferimento della sinusoide distorta sfasata

Si nota lo sfasamento dalla posizione dei picchi dell'onda distorta i quali sono visivamente non in linea con quelli della sinusoide pura.
Arrivati a questo punto è necessario eliminare la fase, per poi reimpostarla manualmente, in modo da sincronizzare l'input (sinusoide pura) con l'output (sinusoide distorta):

```
ffuzz = fft(fuzz);
```

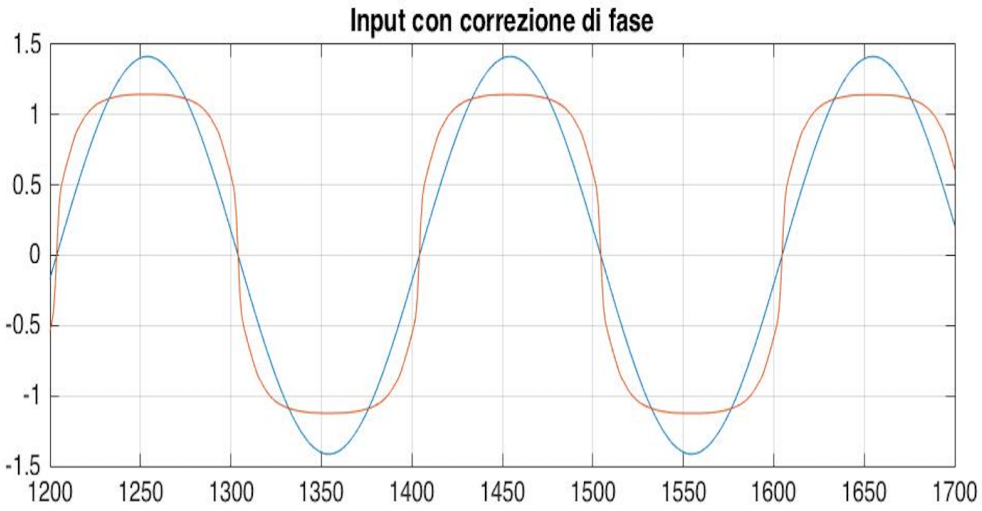
Con questo pezzo di codice passiamo dal domino del tempo al dominio della frequenza, eseguendo una FFT(Fast Fourier Transform) sull'onda distorta.

```
ffuzz = abs(ffuzz);
```

Con “abs” viene restituito il valore assoluto del valore tra parentesi, così facendo vengono scartate le informazioni sulla fase.
Ora va impostata manualmente la fase dell'onda distorta

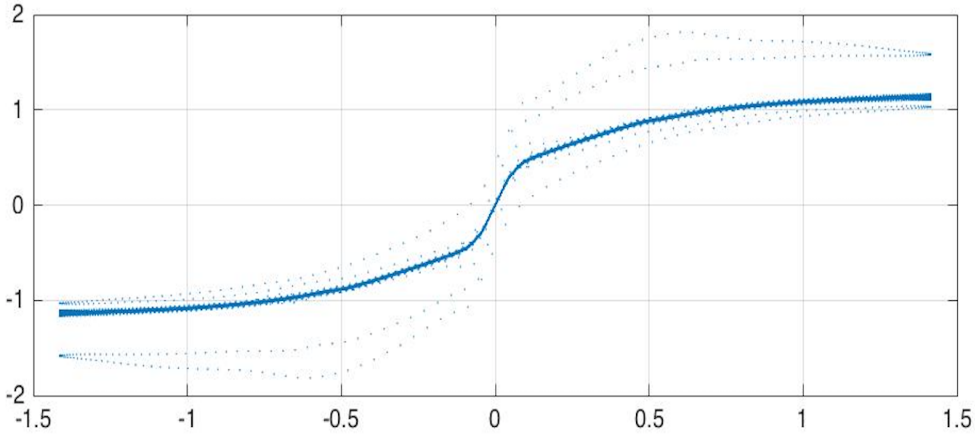
```
ffuzz = -ffuzz .* exp(1i*pi/2);
```

Dopo aver sistemato manualmente la fase, possiamo notare che le due onde sono sincronizzate



22) Confronto tra la sinusoide pura e quella distorta senza la distorsione di fase

Inoltre si può anche notare che la sua funzione di trasferimento riesce ad unire in media tutti i valori.



23) Funzione di trasferimento della sinusoide distorta senza sfasamento

Una volta che è stata sistemata la fase della sinusoide distorta, è possibile salvare il nuovo file audio per poterlo utilizzare allo scopo di ricreare la distorsione inizialmente applicata (assicurandoci che l'audio non clippi):

```
fuzz = fuzz/max(abs(fuzz));
audiowrite('correctfuzz.wav',fuzz,Fs);
```

4.3 Riproduzione della distorsione

Ora che è stata tolta la distorsione di fase all'onda distorta è possibile utilizzarla per ricreare il polinomio applicato su di essa.
La formula della distorsione sarà applicata sulla nota di chitarra registrata in dry e sarà successivamente confrontata con la nota di chitarra distorta per vedere se la

distorsione è stata riprodotta fedelmente.
Inizialmente vanno caricati i tre file audio necessari:

- la sinusoide pura
- la sinusoide distorta (senza distorsione di fase)
- la nota di chitarra in dry

```
[y, ~] = audioread('correctfuzz.wav');
[x, sr] = audioread('sin.wav');
[guitar,gfs] = audioread('notapulita.wav');
```

Dopodiché si eguagliano i valori di potenza della sinusoide distorta e di quella pura con l'RMS

```
y = 0.7071 * y/rms(y);
x = 0.7071 * x/rms(x);
```

Questo viene fatto per poterle confrontare in modo preciso.
Ora è necessario ricostruire il polinomio che crea la distorsione, ma per farlo bisogna decidere il grado del polinomio.
Maggiore sarà il grado, migliore sarà la precisione con cui si andrà a riprodurre il segnale, ma ovviamente non si potrà scegliere un numero troppo grande (ad esempio 1000), perché, nonostante la precisione, il calcolo sarà talmente complesso che non tutti i computer potrebbero riuscire a svolgerlo in tempi brevi. Inoltre, con un grado del polinomio eccessivamente grande, molti risultati sarebbero prossimi allo zero, quindi potrebbero anche non essere considerati (dato che senza quei dati il risultato varierebbe in maniera insignificante).
Il grado scelto per ricreare il polinomio della distorsione scelta è 17 (numero assegnato alla variabile “ordine”).
Per poter estrarre i coefficienti, bisogna applicare due funzioni di Matlab: polyval e polyfit.

```
poly = polyfit(x,y,ordine);
```

Con polyfit verrano estratti i coefficienti di un polinomio di ennesimo grado (in questo caso il grado del polinomio è definito dalla variabile “ordine”) che si adattano meglio per i dati di y.
Quindi avendo un polinomio di ennesimo grado del tipo:

$$p(x)=p_1x_n+p_2x_{n-1}+\ldots+p_{nx}+p_{n+1}$$

Verrebbero estratti tutti i valori p_x.
Questi sono i 18 valori estratti con la funzione polyfit:

1.0059 0.0003 -4.5638 -0.0013 8.6509 0.0021. -8.8803 -0.0018 5.3506
0.0009 -1.9191 -0.0003 0.3986 0.0000 -0.0458 -0.0000 0.0039 0.0000

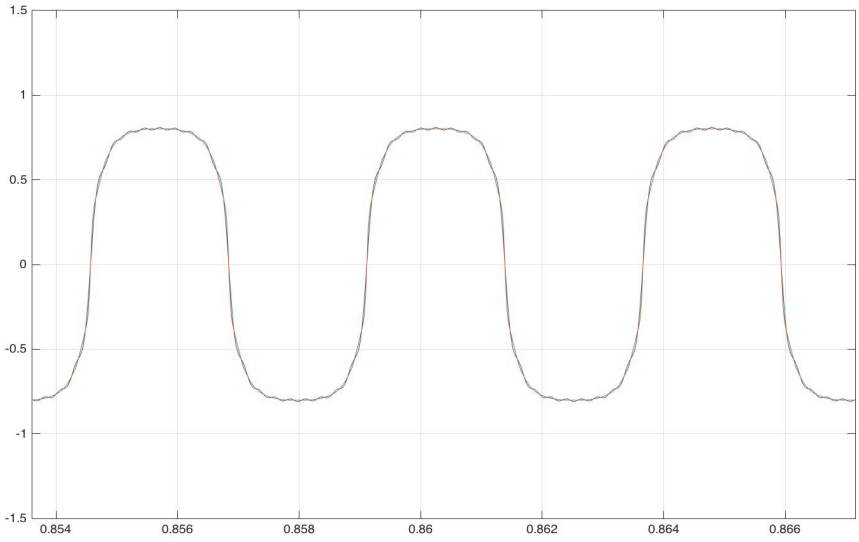
I coefficienti sono rappresentati in ordine decrescente.
Ora bisogna calcolare il valore del polinomio ottenuto con polyfit per i valori di x della sinusoide pura, questo si fa con polyval:

```
y1 = polyval(poly, x1);
```

Ora che abbiamo ottenuto il polinomio e i valori dei coefficienti possiamo applicarlo inizialmente alla sinusoide pura per avere un primo confronto:

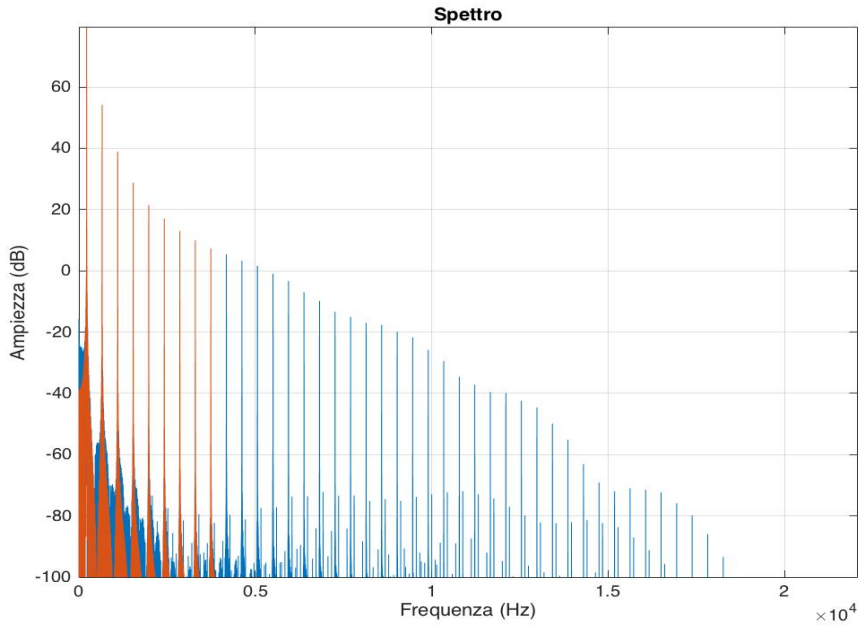
```
y2 = polyval(poly,x);
```

Si può evincere dal grafico che le due onde sono molto simili.
Ovviamente aumentando il numero di valori o l'ordine del polinomio aumenterà diminuirà anche la differenza tra le due onde, che tenderanno a coincidere.



24) Confronto tra la sinusoide distorta originale e la sinusoide distorta ricreata

Osservando lo spettro dell'onda distorta ricreata e quello dell'onda distorta originale senza distorsione di fase, si può notare che i primi armonici sono uguali



25) Analisi dello spettro della distorsione ricreata e di quella originale senza sfasamento

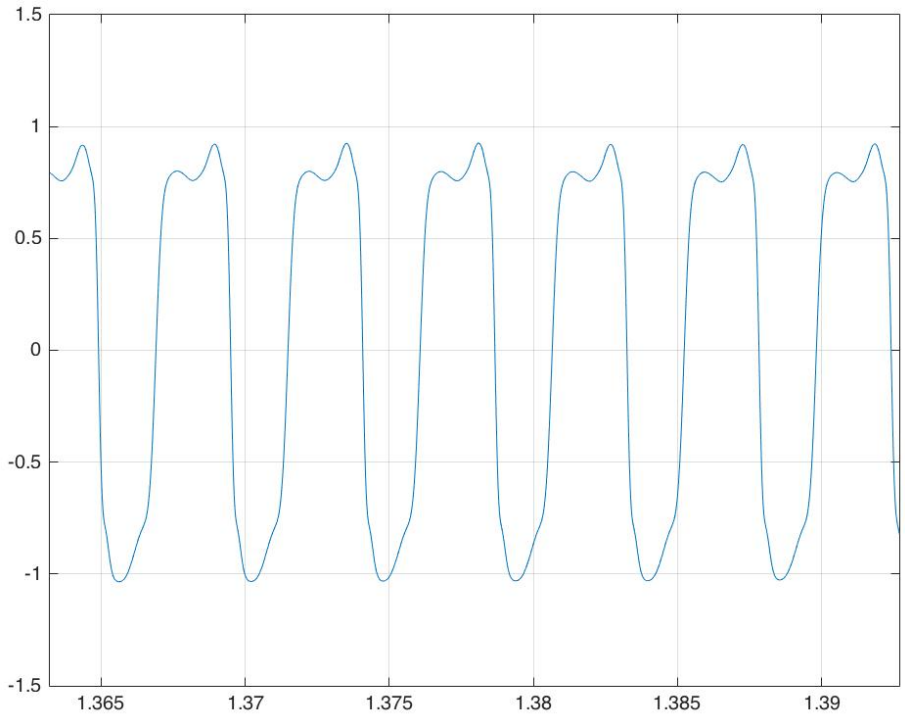
Ovviamente gli armonici non saranno più uguali dopo certi valori, questo perché è stato utilizzato un polinomio di diciassettesimo grado per ricreare la distorsione, se fosse stato usato un polinomio di grado maggiore ci sarebbero state più armoniche coincidenti.
Ora è possibile applicare il polinomio per riprodurre la distorsione originale alla nota di chitarra in dry

4.4 Applicazione della distorsione

In questa fase finale verrà applicato alla nota di chitarra in dry il polinomio ricreato a partire dall'analisi della sinusoide distorta:

```
y3 = polyval(poly,guitar*gain);
```

Oltre al polinomio, il segnale della chitarra è stato moltiplicato per un gain, ovvero un valore che aumenta la potenza del segnale.
Il gain, in questo caso, è settato a 10.



26) Forma d'onda della distorsione ricreata

Essendo il fuzz il tipo di distorsore riprodotto, la forma d'onda della nota distorta tende ad avere una forma simile a quella di un'onda quadra, questo perché la funzione di trasferimento di un fuzz è la stessa di quella di un clipper, quindi oltre una certa soglia il segnale viene semplicemente “tagliato”.
Se si dovesse procedere ad un ascolto per confrontare la nota di chitarra distorta direttamente col plugin utilizzato e quella distorta tramite la ricreazione della distorsione si noterebbero ben poche differenze.

In questo capitolo si vedrà la realizzazione dell'applicativo in Juce, un software finalizzato allo sviluppo di plug in.

5.1 Tutorial di Juce

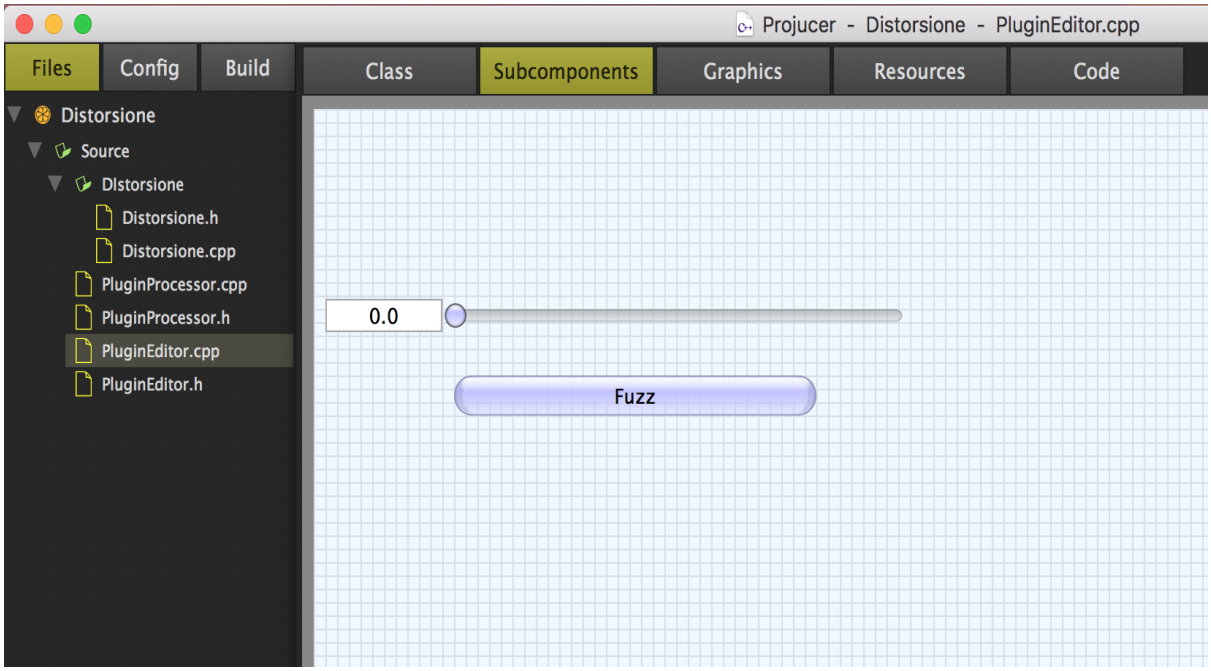
Sul sito della Redwood audio è presente un tutorial di Juce per impostare la base del plugin.
Una parte di codice viene implementata automaticamente da Juce, per poi essere integrata manualmente tramite l'inserimento di frammenti di codice predefiniti forniti dal sito stesso.

5.2 Impostazioni generali

Prima di passare alla scrittura del codice è necessario impostare delle caratteristiche del plugin, infatti, al momento della creazione del progetto, sarà necessario inserire la piattaforma su cui verrà compilato il codice (in questo caso è Xcode, perché il plugin verrà eseguito su un software Apple) il tipo di versione (32 o 64 bit) e il nome della compagnia o del singolo sviluppatore.

5.3 Grafica

Una volta completata la fase di preparazione dell'ambiente di lavoro, è possibile creare l'interfaccia grafica del plugin (quella di questo fuzz sarà molto semplice).
Per accedere all'implementazione dell'interfaccia grafica bisogna aprire il file Plugineditor.cpp e accedere alla sezione “subcomponents”.



27) Creazione dell'interfaccia grafica del plugin

Lo slider rappresenta il livello di gain del pedale di distorsione, mentre il bottone “Fuzz” permette di attivare e disattivare la distorsione

5.4 Codice

Dopo aver impostato il progetto e seguito il tutorial, va implementata la parte di codice relativa alla distorsione del segnale.
Per questo bisogna recuperare i valori del vettore trovati con la funzione polyfit nel progetto di Matlab, questi valori serviranno a riscrivere il polinomio per creare la distorsione.
Viene dichiarato un vettore di numeri decimali in cui vengono messi tutti i valori, dal coefficiente dell'esponente più basso a quello più alto.

```
float arraypolinomio[18] = {0.000053898, 3.88271530, -0.002592918,
-45.792737557, 0.037626616... };
```

dopodiché si moltiplica il segnale in uscita per il valore di gain (controllato tramite lo slider).
Questo procedimento si fa sia per il canale destro che per il canale sinistro

```
 *LeftSample *= m_gain;
 *RightSample *= m_gain;
```

Infine si applica, sia al canale destro che al canale sinistro, il polinomio per distorcere il suono

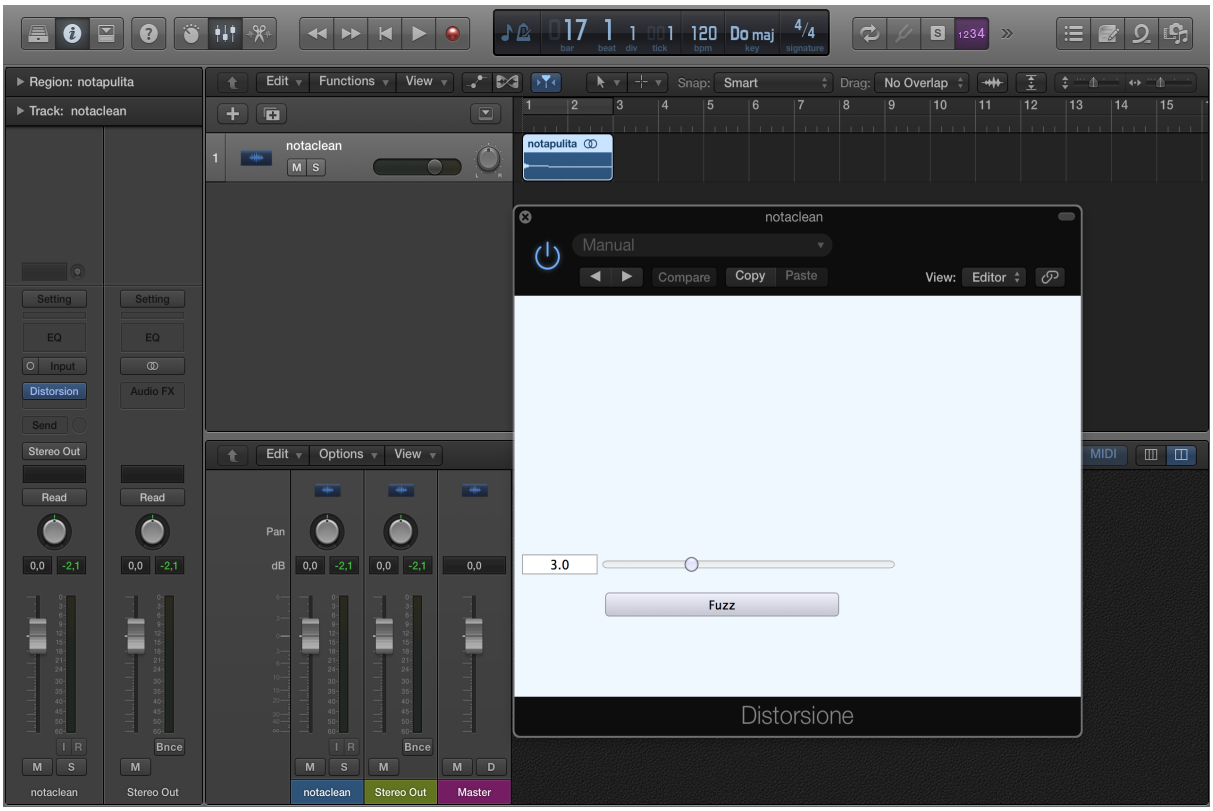
```
for (int i = 0; i<18; i++){
      outputL += arraypolinomio[i] * pow(*LeftSample, i );
   }
   *LeftSample = outputL;

   for (int j = 0; j<18; j++){
      outputR += arraypolinomio[j] * pow(*RightSample, j );
   }
   *RightSample = outputR;
```

Il codice non può essere compilato con Juce, ma va compilato con Xcode (nel caso si usi un sistema operativo Apple) o con altri software che permettono di eseguire un codice in C++.

5.5 Installazione

Questo plugin è stato progettato per Logic pro, quindi l'installazione è molto semplice, basta inserire il file *distorsione.cpp* nella cartella “components”, ovvero la cartella contenente tutti i plugin che possono essere utilizzati con Logic.

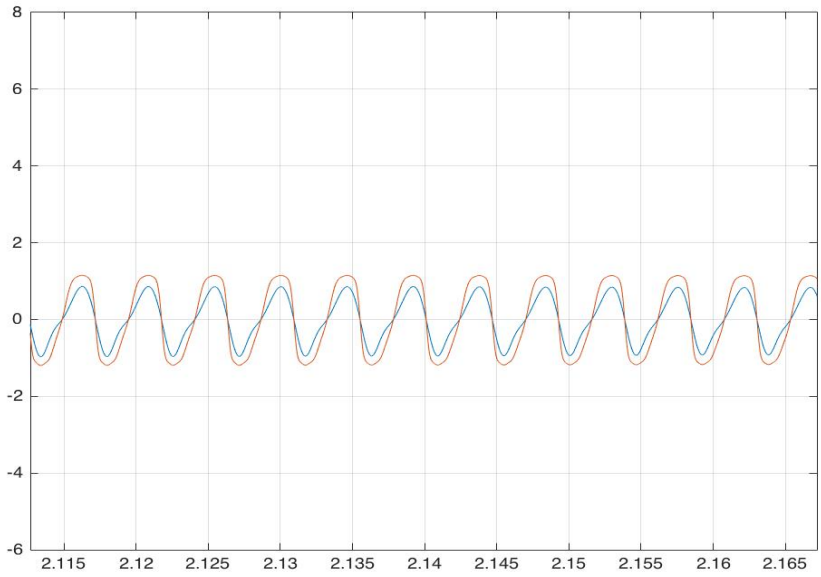
Il plugin verrà subito scansionato e riconosciuto da logic e potrà essere utilizzato su qualsiasi traccia.



28) Il plugin installato su logic

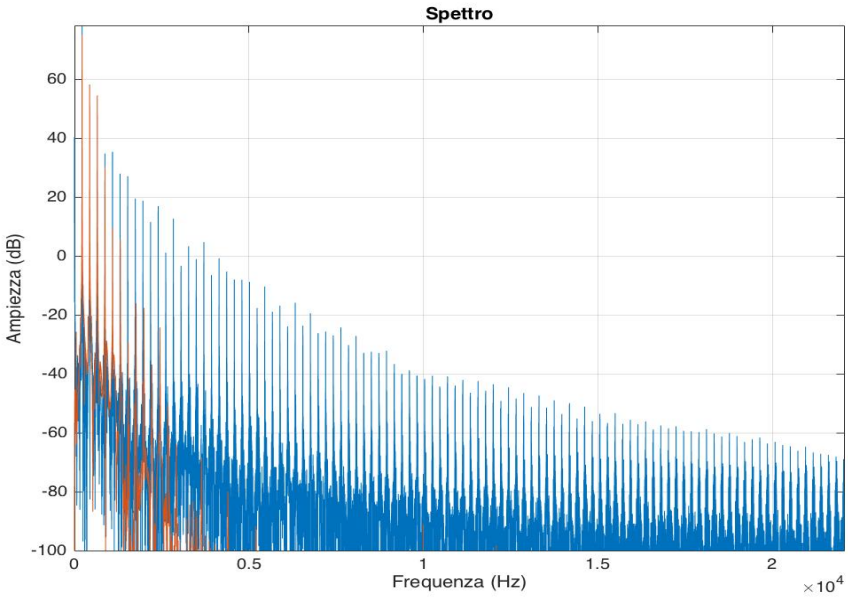
6.1 Confronto tra nota distorta e nota in dry

Ora, dopo l'implementazione del plugin e dell'analisi delle forme e degli spettri dell'onda, si può passare dal confronto tra le sinusoidi al confronto tra la nota distorta studiata per ricreare la distorsione e la nota distorta ricreata a partire dalla nota in dry.



29) Confronto tra la nota in dry (blu) e la nota distorta ricreata (rosso)

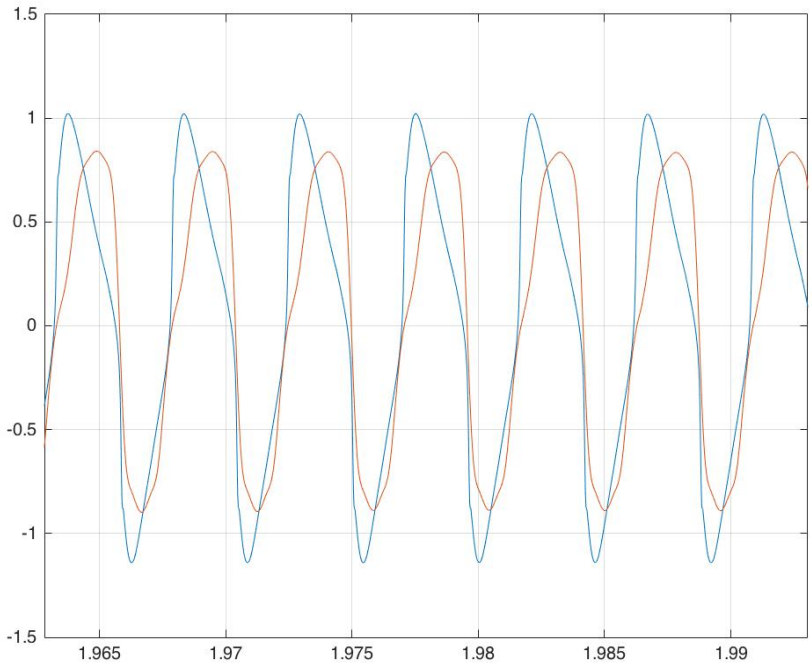
Analizzando invece lo spettro delle due onde possiamo notare che i segnali hanno poche armoniche in comune.



30) Confronto tra gli spettri della nota in dry (rossa) e la nota distorta ricreata (blu)

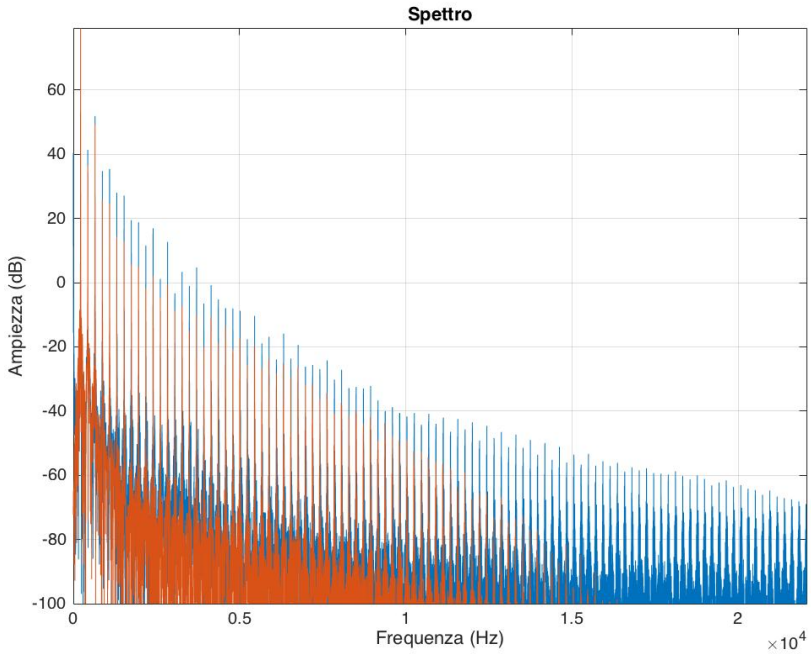
6.2 Confronto tra le distorsioni

Un confronto di spettri più interessante è quello tra la nota distorta originale e quella ricreata con il polinomio (con Matlab).



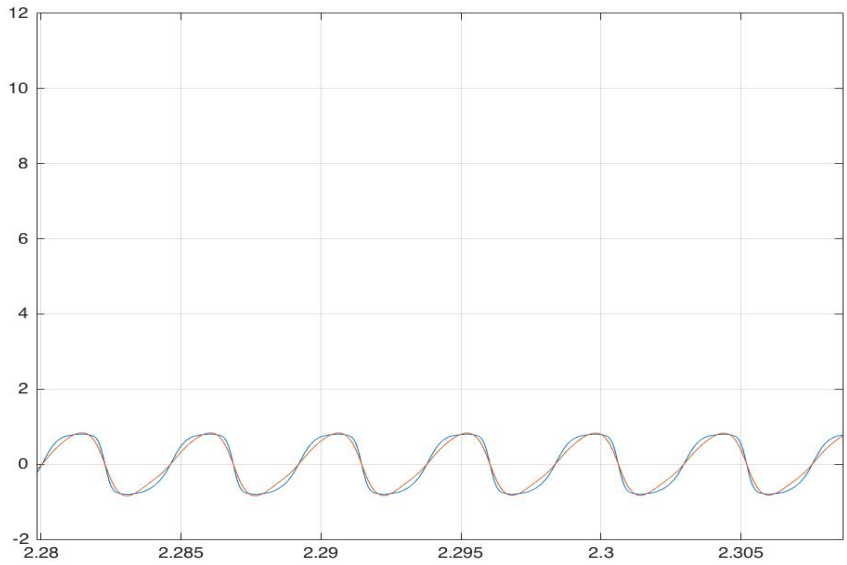
31) Confronto tra la nota distorta originale (blu) e quella ricreata (rossa)

Da questo grafico si nota che le onde sono sfasate, questo perché nel ricreare la distorsione è stata eliminata la distorsione di fase.
Analizzando gli spettri delle due onde si può notare che le due hanno molte armoniche in comune fino ad una certa frequenza, dopodiché le armoniche della distorsione ricreata tenderanno ad essere sempre meno presenti.
Questo è dettato dall'ordine del polinomio utilizzato per ricreare il polinomio della distorsione originale, perché più si utilizza un valore alto, più saranno le armoniche che i due segnali avranno in comune.



32) Confronto tra lo spettro della nota distorta originale (blu) e quella ricreata (rosso)

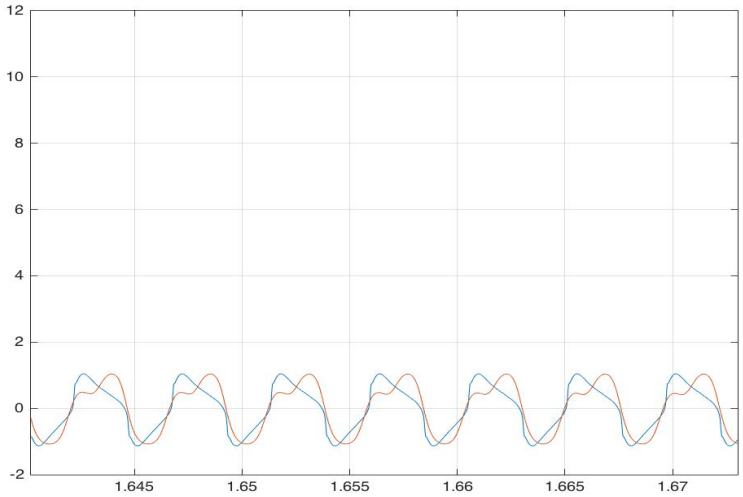
Ora che è stata ricreata la nota distorta si può passare ad un confronto diretto tra la nota distorta ricreata in Matlab e quella ricreata col plugin.



33) Confronto tra la nota distorta ricreata in matlab (blu) e quella ricreata col plugin (rossa)

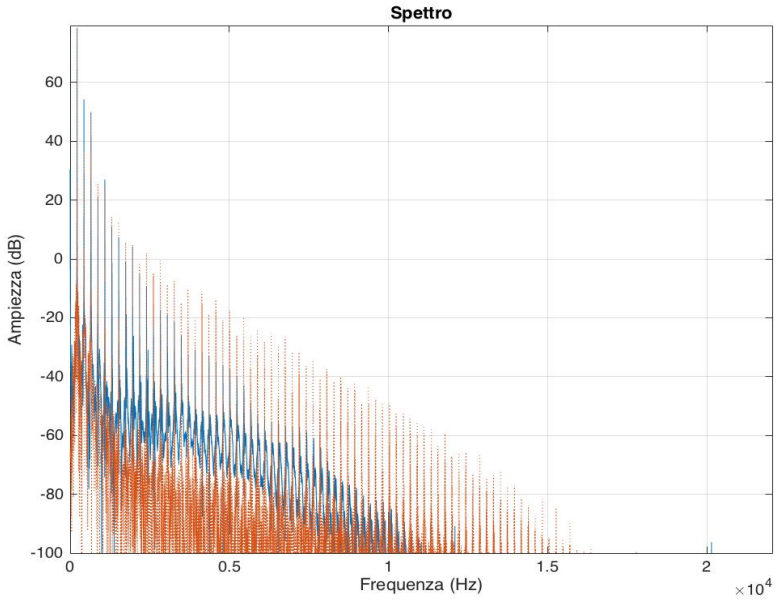
Si può osservare che le due forme d'onda sono molto simili tra loro, ma sopratutto si nota che non è presente uno sfasamento dell'onda, poiché la distorsione di fase è stata eliminata in entrambi i casi.

Le onde non coincidono in maniera precisa, perché i coefficienti del polinomio utilizzati in Juce sono stati approssimati a 9 cifre dopo la virgola, mentre Matlab utilizza i coefficienti completi.
Infine rimane l'ultimo test da fare, ovvero tra la nota ricreata con Juce e la nota distorta col plugin originale.



34) Confronto tra la nota distorta originale (blu) e quella distorta col plugin (rossa)

Le forme d'onda sono sfasate, perché nel plugin di Juce è stata eliminata la distorsione di fase presente nella nota originale.
Passando ad un'analisi dello spettro si può vedere come le armoniche coincidono.



35) Differenza di spettro tra l'onda distorta originale (blu) e quella distorta col plugin (rossa)

7.1 Conclusioni

L'obbiettivo iniziale era quello di ricreare un distorsore digitale esistente partendo da un analisi di forme e spettri d'onda, cercando di capire il funzionamento che sta alla base di plugin digitali.
Lo studio è stato completato con successo, sono stati analizzati tutti i suoni ed è stata ricreata una versione semplificata molto simile del plugin originale.
Nonostante i plugin digitali siano più economici degli effetti o degli amplificatori analogici, hanno comunque un costo abbastanza elevato, questo perché il lavoro che c'è dietro alla creazione di un qualsiasi effetto digitale è molto complesso e richiede determinate conoscenze informatiche e matematiche, quindi nonostante un determinato plugin esegua delle funzioni elementari per un musicista, la sua implementazione potrebbe aver richiesto parecchio studio e tempo.

7.2 Sviluppi futuri

Il plugin sviluppato è molto semplice, ha un interfaccia minimale con un controller per il gain e un pulsante di attivazione.
Si potrebbe pensare ad alcuni sviluppi futuri per questo software, questi potrebbero essere perfezionamenti o cambiamenti significativi.
Alcuni esempi di aggiornamento sono:

- Arricchimento dell'interfaccia grafica
- Aggiunta di funzioni (come un controllo del volume o un filtro)
- Integrazione con un set di altre due o tre distorsioni con un selettore
- Implementazione di una possibile compatibilità con un foot-switch MIDI per un possibile utilizzo del plugin in una situazione live

Oltre a questi sviluppi si potrebbe anche perfezionare il plugin stesso, aumentando la precisione di creazione della distorsione, modificare la formula polinomiale stessa per adattarla meglio a simulatori di casse o all'aggiunta di impulsi.

[1] Recordingology.com	https://recordingology.com	(22/11/2016)
[2] Mathworks.com	https://it.mathworks.com	(22/11/2016)
[3] Juce.com	https://www.juce.com	(22/11/2016)
[4] Musicradar.com	http://www.musicradar.com	(22/11/2016)
[5] Screaminfx.com	http://screaminfx.com	(22/11/2016)
[6] Rs-met.com	http://www.rs-met.com	(22/11/2016)
[7] Msp.ucs.edu	http://msp.ucsd.edu	(22/11/2016)
[8] Redwoodaudio.net	http://www.redwoodaudio.net	(22/11/2016)
[9] Ludovico.net	http://www.ludovico.net	(22/11/2016)
[10] Sito del LIM	http://www.lim.di.unimi.it	(22/11/2016)

Vorrei ringraziare tutti coloro che mi hanno accompagnato in questo percorso.

Un ringraziamento speciale va alla mia famiglia per tutto il supporto che mi ha dato.

Un grazie va anche ai miei compagni di corso, in particolare ad Alessandro Defendenti, Federico Durosini, Maria Elena Silletti e Paolo Boracchi, per aver sostenuto degli esami insieme a me e per avermi aiutato nei momenti di difficoltà.

Un ultimo ringraziamento va alla mia ragazza e a tutti i miei amici che mi hanno sostenuto durante questi quattro anni di studi.