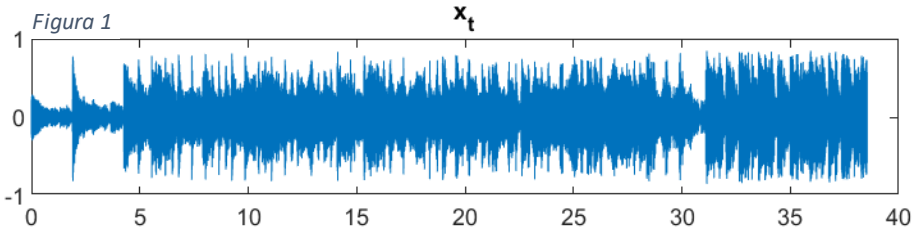
Relazione tesina Matlab 2022 - Matteo De Gobbi 2009765

Parte 1: demodulazione del segnale x_t e ascolto

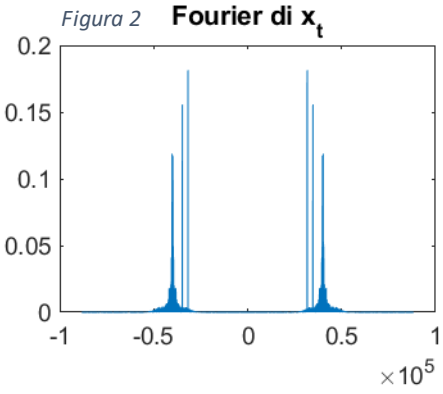
Figura 1



In *Figura 1* vediamo il plot del segnale x_t caricato dal file audio.mat, vediamo che è un segnale di circa 40 secondi e sappiamo che è campionato a F=176400 Hz.

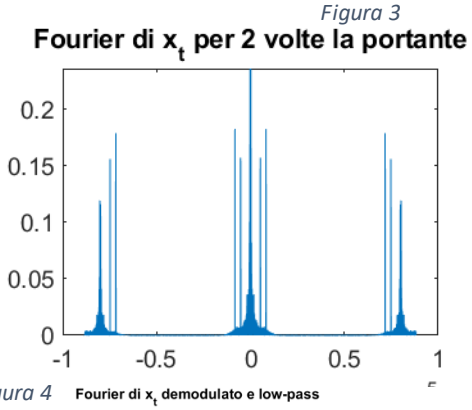
L'approssimazione della trasformata di x_t viene calcolata usando i comandi *fft* e *fftshift* di Matlab. A differenza delle esercitazioni a lezione non è necessario moltiplicare per un esponenziale complesso la TDF perché stiamo considerando un segnale definito in un intervallo [0,40].

Figura 2



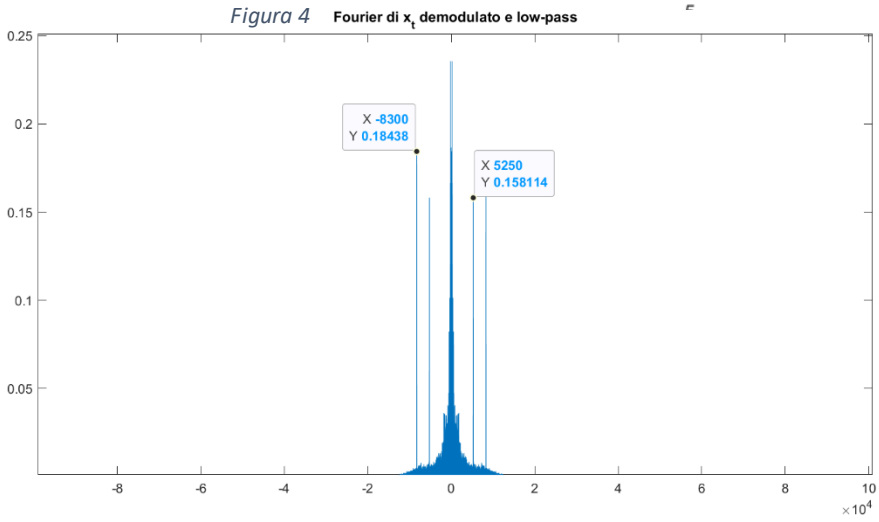
Plottando la TDF di x_t (*Figura 2*) vediamo come ha dei picchi centrali in ±40000Hz più altri 4 picchi in ±34750Hz e in ±31700Hz. Sapendo che $x_t(t) = x(t)\cos(2\pi F_m t)$ e usando la proprietà di modulazione con $X(f) := TDF[x(t)](f)$ abbiamo che $X_t(f) = \frac{1}{2}X(f - F_m) + \frac{1}{2}X(f + F_m)$ quindi guardando il disegno capiamo che $F_m = 40Khz$ (il suggerimento che è multipla di 5KHz è verificato). Ora che conosciamo la frequenza di modulazione possiamo demodulare x_t e ascoltarlo. Per demodulare x_t basta moltiplicarlo per 2 volte la portante e poi filtrarlo con un filtro passa basso in modo da prendere solo la X(f) e non le traslazioni in frequenza.

Figura 3



Questo (*Figura 3*) è il plot della TDF del segnale x_t per 2 volte la portante, vediamo che oltre al supporto centrato in 0 ci sono anche 2 traslazioni con ampiezza metà di quella centrale a ±80KHz. Se ascoltassimo così l'audio queste traslazioni non interferirebbero perché sono ad una frequenza troppo alta per l'udito ma filtriamo comunque perché come vedremo dopo non filtrare darebbe problemi nel caso di campionamento. Ho scelto come frequenza di stopband 40KHz perché è a metà tra 0 e 80000Hz. Ascoltando il segnale usando le funzioni *play* e *audioplayer* sentiamo Starman di David Bowie con un fischio acuto sovrapposto alla canzone. Guardando la *Figura 4* vediamo dei picchi in ±5250 e ±8300 che corrispondono agli artefatti presenti nella canzone. Per provarlo basta generare due sinusoidi in Matlab

Figura 4



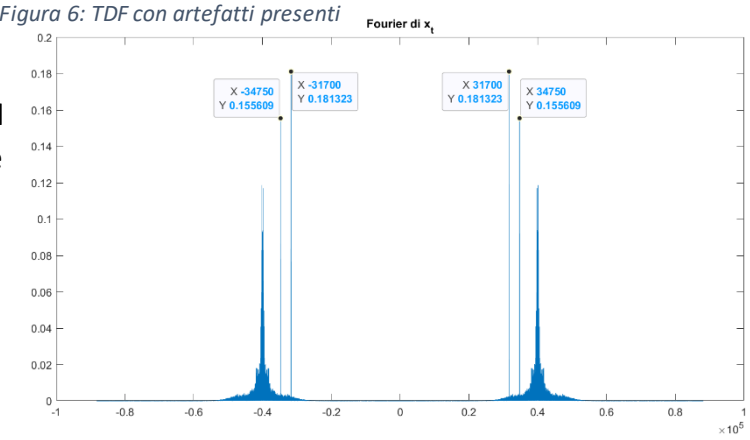
alle frequenze degli artefatti e ascoltandole con *audioplayer* viene un rumore simile a quello presente nel segnale demodulato.

Figura 5: come generare un artefatto simile a quello nella canzone

```
load audio.mat
N = length(x_t);
t=0:1/F:(N-1)*(1/F);
v=sin(2*pi*5250*t)+sin(2*pi*8300*t);
p=audioplayer(v,F);
```

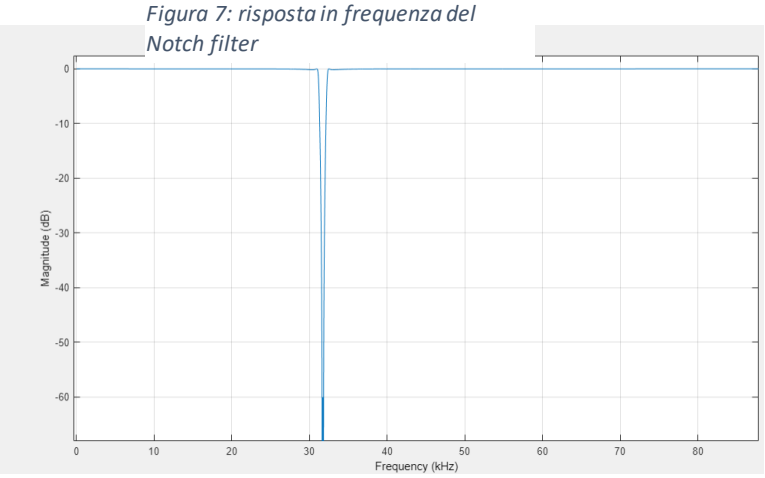
Parte 2: rimozione artefatti dal segnale

Figura 6: TDF con artefatti presenti



Ora noi vogliamo rimuovere questi artefatti dal segnale e guardando la *Figura 6* vediamo come sono presenti delle componenti in frequenza a ±31700Hz e ±34750Hz nel segnale modulato x_t, queste componenti quando demoduliamo il segnale si manifestano nell'artefatto che abbiamo ascoltato prima. Quindi rimuovendo queste componenti dovremmo poter rimuovere l'artefatto dal segnale demodulato.

Figura 7: risposta in frequenza del Notch filter

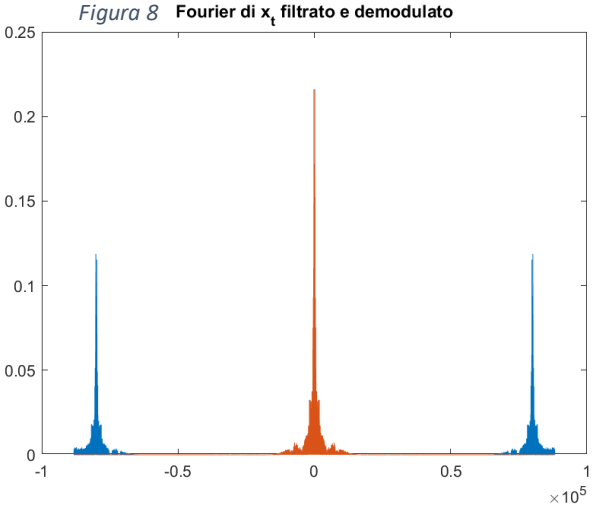


Per questo scopo si possono usare dei Notch filter che permettono di attenuare le componenti in frequenza in un intervallo molto stretto in modo da non perdere troppo del segnale originale quando rimuoviamo gli artefatti. Su Matlab per creare i Notch filter usiamo *NF_design* e passiamo come parametri: passo di campionamento $T = 1/F$ dove F=176400Hz è la frequenza di campionamento del segnale e come frequenza di maggior attenuazione abbiamo per il primo filtro 31700Hz e per il secondo 34750Hz.

In *Figura 7* vediamo la risposta in frequenza del primo filtro (in dB). Si può notare come il filtro attenua in un piccolo intervallo attorno a 31.7KHz con la massima attenuazione in 31.7KHz.

Dopo aver filtrato il segnale lo demoduliamo moltiplicando per 2 volte la portante, se ora si ascolta il segnale demodulato non è più presente l'artefatto e si sente solo la canzone (dopo aver moltiplicato per 2 volte la portante sarebbe da filtrare con un low-pass filter ma per ora non è necessario, lo faremo quando dobbiamo campionare nella parte 3 per evitare l'aliasing). In *Figura 8* vediamo la TDF del segnale dopo averlo moltiplicato per 2 volte la portante. Se filtriamo con un filtro passa basso otterremo una TDF uguale alla componente arancione in figura (ne parlerò meglio nella parte del campionamento).

Figura 8 **Fourier di x_t filtrato e demodulato**

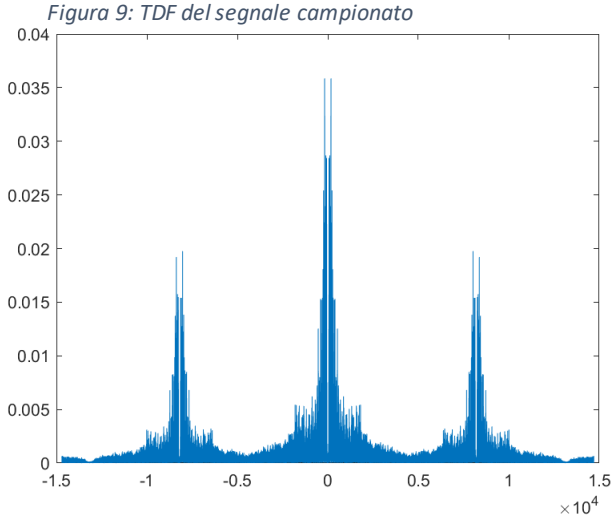


Parte 3: campionamento

Vogliamo ottenere il segnale x_c campionando x(t) con una frequenza di campionamento $F_c = 29400$ Hz che notiamo

essere 1/6 della frequenza F del segnale caricato con *load audio.mat.* Notiamo che F dev'essere per forza un multiplo intero di Fc altrimenti non avremmo i campioni necessari, avendo solo a disposizione una discretizzazione del segnale continuo. Per ottenere x_c quindi prendiamo un elemento ogni 6 dal vettore da "campionare"; questo si può fare attraverso lo slicing dei vettori in Matlab con *x_c=x_filt_dem_rep(1:6:end);* dove *x_filt_dem_rep* e il segnale x_t filtrato con i due Notch filter e moltiplicato per 2 volte la portante.

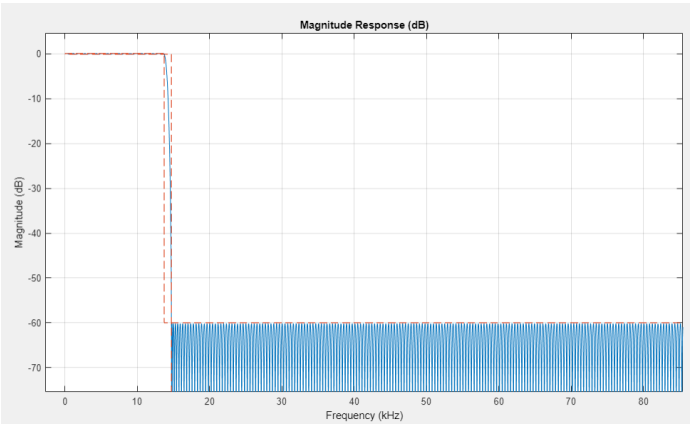
Figura 9: TDF del segnale campionato



Ora ascoltiamo x_c con *audioplayer* (*player3* su Matlab) sentiamo come sono presenti nuovamente degli artefatti. Guardando la TDF di x_c (*Figura 9*) vediamo come sono presenti delle componenti intorno a ±8Khz responsabili degli artefatti che sentiamo nel segnale campionato. Queste componenti sono dovute al fenomeno dell'aliasing, infatti, noi stiamo campionando a Fc=29400Hz ma se guardiamo la TDF del segnale che stiamo campionando (in *Figura 8*) vediamo come le due componenti in blu (che prima non sentivamo nell'audioplayer perché a frequenza superiore a quella udibile) si trovano a circa ±80Khz e quindi in questo caso non valgono le ipotesi del Teorema di Shannon che necessita Fc>2B dove in questo caso B è circa 90000Hz.

Per poter avere un segnale campionato senza artefatti possiamo usare un filtro passa basso come filtro antialiasing prima di campionare, in modo che valga il Teorema di Shannon. Per creare un filtro passa basso su Matlab usiamo la funzione *LPF_design* e come parametro passiamo il passo di campionamento (del segnale originale non di x_c) 1/F e la frequenza di stopband 29400/2 in modo che valga Fc>2B e non ci sia aliasing. In pratica stiamo tenendo solo la parte arancione della TDF in *figura 8.*

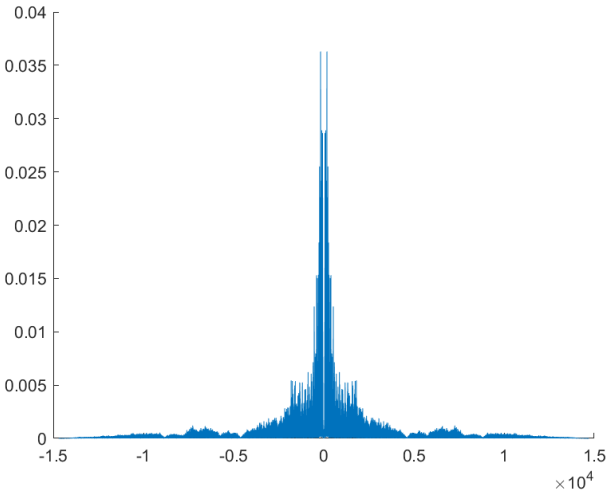
Figura 10: risposta in frequenza del filtro passa basso



In *figura 10* vediamo la risposta in frequenza del filtro passa basso che inizia ad attenuare da frequenze intorno ai 14KHz.

Quindi applichiamo questo filtro con la funzione *filter* di Matlab al segnale *x_filt_dem_rep* e poi campioniamo sempre attraverso lo slicing dei vettori di Matlab e ora, ascoltando questa versione del segnale campionato dove abbiamo usato il filtro anti aliasing, possiamo sentire Starman senza artefatti.

Figura 11: TDF del segnale campionato usando il filtro antialiasing



Infatti, analizzando la TDF in *figura 11* vediamo come non sono più presenti i picchi dovuti all'aliasing (a differenza della *figura 9)*.

Codice commentato:

```
%% carico i dati dal file audio.mat
clear all
close all
clc
load audio.mat

%% plotto x_t e X_t
N = length(x_t);
t=0:1/F:(N-1)*(1/F);%genero il vettore dei tempi
figure(1)
subplot(311)
plot(t,x_t)
title("x_t");

fc = F/(N); %passo di campionamento del vettore delle
frequenze
X_t = (1/F)*fft(x_t,N);
X_t = fftshift(X_t);

f = fc*(-N/2:N/2-1); %vettore delle frequenze
figure(2)
subplot(221)
plot(f,abs(X_t))
title("Fourier di x_t")
Fm=40000; %Hz oppure 40KHz
%% Plotto segnale demodulato
x=2*cos(2*pi*Fm*t).*x_t;%moltiplico per 2 volte la portante
X=(1/F)*fft(x,N);
X=fftshift(X);

figure(2)
subplot(222)
plot(f,abs(X))
title("Fourier di x_t per 2 volte la portante")

HLPdemod=LPF_design(1/F,40000);%filtro passa basso per
demodulare correttamente
x=filter(HLPdemod,x);
X=(1/F)*fft(x,N);
X=fftshift(X);
figure(1)
subplot(312)
```

```
plot(t,x)
title("x_t demodulato e low-pass")
figure(2)
subplot(223)
plot(f,abs(X))
title("Fourier di x_t demodulato e low-pass")
player1 = audioplayer(x,F);%audio con artefatti

%% filtro x_t
HNF1=NF_design(1/F,31700);%notch filter per eliminare gli
artefatti
HNF2=NF_design(1/F,34750);
x_filt=filter(HNF1,x_t);
x_filt=filter(HNF2,x_filt);
%decommentare se si vuole la trasformata
%X_filt=(1/F)*fft(x_filt,N);
%X_filt = fftshift(X_filt);

%% demodulo e ascolto di nuovo
%lowpass per evitare l'aliasing quando campiono
HLPdemod=LPF_design(1/F,29400/2);

x_filt_dem_rep=2*x_filt.*cos(2*pi*Fm*t);%demodulo con due
volte la portante
x_filt_dem=filter(HLPdemod,x_filt_dem_rep);%filtro per
antialiasing
X_filt_dem=(1/F)*fft(x_filt_dem,N);
X_filt_dem = fftshift(X_filt_dem);
figure(1)
subplot(313)
plot(t,x_filt_dem)
title("x_t filtrato e demodulato")
figure(2)
subplot(224)
plot(f,abs(fftshift((1/F)*fft(x_filt_dem_rep,N))));
hold on
plot(f,abs(X_filt_dem))
title("Fourier di x_t filtrato e demodulato")
player2 = audioplayer(x_filt_dem_rep,F);%audio senza artefatti
%% campionamento
x_c=x_filt_dem_rep(1:6:end);%campiono con lo slicing
N_camp=length(x_c);%numero di campioni
```

```
f_camp=fc*(-N_camp/2:N_camp/2-1);% nuovo vettore delle
frequenze
player3=audioplayer(x_c,F/6);%audio con artefatti
X_c=(1/F)*fft(x_c,N/6);
X_c = fftshift(X_c);
x_cap=x_filt_dem(1:6:end);%campiono con lo slicing
X_cap=(1/F)*fft(x_cap,N/6);
X_cap = fftshift(X_cap);
player4=audioplayer(x_cap,F/6);%audio senza artefatti
figure()
plot(f_camp,abs(X_c))
hold on
plot(f_camp,abs(X_cap))
title("TDF senza filtro AA / con filtro AA")
```