

SETTIMANA MATEMATICA

Laboratorio “La matematica dei suoni”

Parte 3: Eliminare il rumore

5 – 7 febbraio 2020

Supponiamo che il suono di nostro interesse sia stato contaminato dal rumore. Ad esempio, potremmo ascoltare una voce proveniente da una ricetrasmittente su un canale disturbato, oppure la radio in una zona con molte interferenze.

Cosa succede in questo caso? Il segnale originale $s(t)$ viene modificato, ad esempio aggiungendo del rumore incognito $n(t)$. Chiamiamo il segnale disturbato

$$\hat{s}(t) := s(t) + n(t).$$

Ci piacerebbe comprendere come riconoscere ed eliminare questo rumore $n(t)$.

1 Qualche nuovo comando MATLAB

Per gli esercizi di oggi ci faranno comodo delle istruzioni di MATLAB leggermente più avanzate di quelle che abbiamo utilizzato fino ad ora.

1.1 Operazioni vettoriali

Spesso, si preferisce utilizzare operazioni definite direttamente su vettori. Ad esempio, potremmo sommare o moltiplicare elemento ad elemento due vettori v_1 a v_2 :

```
v1 = [ 1, 2, 3 ];  
v2 = [ 4, 5, 6 ];
```

```
w = v1 + v2;  
w = v1 .* v2;
```

Notiamo l'utilizzo dell'operatore `.*` invece che semplicemente `*`. Questo è richiesto dal fatto che MATLAB normalmente effettua operazioni tra matrici (che al momento non ci interessano, ma darebbero un errore in questo caso specifico).

Su vettori si possono utilizzare operazioni booleane. Ad esempio, supponiamo di voler importare a zero tutti gli elementi negativi di un vettore v . Allora, possiamo creare il vettore w_j uguale ad 1 se $v_j \geq 0$, a 0 altrimenti, e poi moltiplicare v e w elemento per elemento. Per fare questo, utilizziamo la convenzione che `true == 1`, e `false == 0`, ed eseguiamo le istruzioni:

```
w = (v >= 0);  
t = v .* w;
```

Il vettore t sarà uguale a v nelle componenti positive, e zero altrimenti. Useremo queste istruzioni in seguito per filtrare i segnali. Provate a farvi stampare i vettori w e t in questo esempio per comprendere bene il funzionamento.

Esercizio 1.1. *Per prendere confidenza con queste istruzioni, realizzate uno script¹ (i.e., un file con estensione `.m` che esegua una serie di comandi) che definisca un vettore v con alcune componenti di modulo maggiore di 1 e alcune di modulo minore di 1. Si usino le istruzioni vettoriali per impostare quelle di moduli maggiore di 1 esattamente ad 1. Hint: si usi la funzione `abs` per calcolare il modulo. Come cambierebbe questo codice utilizzando un ciclo `for`?*

2 Usare i coefficienti di Fourier

Come primo metodo, utilizzeremo due funzioni realizzate per determinare i coefficienti della serie di Fourier di un segnale campionato su N punti in un intervallo di tempo $[0, T]$. La versione continua del segnale si rappresenta con

$$s(t) = \frac{a_0}{2} + \sum_{i=0}^{\lceil \frac{N-1}{2} \rceil} a_i \cos(2\pi i \omega t) + b_i \sin(2\pi i \omega t), \quad \omega = \frac{1}{TN} \quad (1)$$

Nota: la sommatoria qui va solo fino a $\lceil \frac{N-1}{2} \rceil$ perché, con un sampling di N punti a disposizione, queste sono le massime frequenze che possiamo correttamente rappresentare senza rischiare l'aliasing, come da teorema di Shannon.

Esercizio 2.1. *Caricate uno dei segnali disturbati tramite il comando*

```
>> [s, Fs] = audioread('nome.wav');
```

Il valore Fs vi darà la frequenza di campionamento, da passare come secondo parametro a `sound`, nel caso vogliate ascoltarlo, usando il comando `sound(s, Fs)`. Determinate i coefficienti della sua somma di Fourier (1) utilizzando la funzione `[a, b] = seriefourier(s)`. Osservate il loro modulo con i comandi

```
>> semilogy(abs(a));
>> semilogy(abs(b));
```

che producono un grafico in scala logaritmica. Osserviamo che il segnale originale è codificato dai pochi picchi che si vedono, e che possiamo distinguere dal resto. Possiamo provare a recuperarlo selezionando solo quei picchi, mantenendo solo i coefficienti di Fourier che siano più grandi (in modulo) del 5% del coefficiente massimo. Questo si può fare in MATLAB con il comando

```
>> a2 = a .* (abs(a) > 0.005 * max(abs(a)));
>> b2 = b .* (abs(b) > 0.005 * max(abs(b)));
```

A questo punto possiamo ricostruire il segnale:

```
>> s2 = suonodaserie(a, b);
>> sound(s2, Fs);
```

¹Potete creare uno script creando sul pulsante New in alto a sinistra in MATLAB, e salvarlo come `nomefile.m`. Dopodiché, a patto di essere nella stessa cartella, potete eseguire tutti i comandi all'interno in sequenza digitando `nomefile` nella command windows.

in modo da eliminare il rumore. Provate a variare il coefficiente di taglio 0.005; noterete che incrementandolo il rumore sparisce, ma appaiono altri effetti poco piacevoli. In alternativa, provate a copiare i coefficienti azzerando tutti quelli da un certo punto in poi (ed eliminando le frequenze alte). Questo eliminerà il rumore, ma renderà il suono più cupo: non è facile trovare il punto di taglio ideale. Potete provare con i comandi:

```
>> a2 = a; a2(8000:end) = 0;
>> b2 = b; b2(8000:end) = 0;
```

Per poi ricostruire il suono come prima.

Un'alternativa per ridurre il rumore può essere filtrare le frequenze più alte. Questo funziona perché il segnale originale ha la parte predominante nelle frequenze relativamente basse (diciamo fra i 200 e i 2000 Hz), mentre il rumore bianco è composto da tutte le frequenze disponibili. Ovviamente, quest'operazione non ridurrà il rumore del tutto, perché la parte con frequenze coincidenti con il segnale "buono" non verranno eliminate.

3 Filtri di lunghezza finita

Nella pratica è raro potersi permettere di filtrare il rumore calcolando i coefficienti della serie di Fourier del segnale di interesse, per poi modificarli. Questo calcolo richiede un processore sufficientemente potente, ed è impossibile da fare in tempo reale.

Per questa ragione, spesso si preferisce pulire il rumore applicando dei filtri come quelli visti nel precedente laboratorio, ovvero determinare un vettore w per cui l'output del comando

```
>> s2 = conv(s, w)
```

abbia approssimativamente gli stessi coefficienti di Fourier nelle componenti in bassa frequenza ma renda molto piccoli quelle delle componenti in alta frequenza.

Esiste una branca della matematica e dell'ingegneria che si occupa di questo problema, nota come *filter design*. Capire come costruire un filtro di questo tipo può essere complesso, ma almeno ci possiamo accontentare di determinare l'effetto che ci aspettiamo che abbia sulle varie frequenze.

Osserviamo che dato un segnale ottenuto come somma di altri due,

$$s(t) = s_1(t) + s_2(t)$$

si ottiene immediatamente che la convoluzione di s con w è la somma della convoluzioni di s_1 ed s_2 con w ; in linguaggio MATLAB:

```
>> conv(s, w) == conv(s1, w) + conv(s2, w)
```

Esercizio 3.1. Si provi a giustificare l'affermazione precedente usando l'interpretazione di conv come prodotto di polinomi.

Se consideriamo un segnale ottenuto come somma di seni e coseni (ovvero, grazie alla serie di Fourier, ogni segnale di interesse!), allora per capire come agisce il filtro sulle frequenze possiamo applicarlo singolarmente a funzioni del tipo:

$$s_\omega(t) = \sin(2\pi\omega t),$$

che corrisponde alla frequenza ω .

Per il prossimo esercizio, potrete aver bisogno di cicli for, che permettono di eseguire delle operazioni per un indice che varia su un intervallo; ad esempio, potremmo creare un vettore contenente i numeri $\frac{1}{1+j}$ per $j = 1, \dots, 100$ con la seguente sintassi:

```
v = [];
for j = 1 : 100
    v(j) = 1 ./ (1 + j); % MATLAB allunga i vettori automaticamente
end
```

Esercizio 3.2. Si provi a caricare i filtri utilizzando il comando `load filtri`, che definirà delle variabili $w1$, $w2$, $w3$, $w4$, $w5$. Determinare che effetto hanno questi filtri sulle varie frequenze, costruendo una discretizzazione del tempo

```
>> t = 0 : (1/44100) : 1;
```

e valutando l'attenuazione della frequenza f calcolando:

```
>> s = sin(2*pi*t*f); hf = max(abs(conv(s, w)));
```

Considerare le frequenze da 0 a 10000 Hz definendo $f = 1 : 10 : 10000$, per poi applicare il test su $f(j)$, definendo un vettore di attenuazioni hf . Il codice assomiglierà al seguente:

```
t = 0 : (1/44100) : 1;
f = 1 : 10 : 10000;
hf = [];
for j = 1 : length(f)
    hf(j) = max(abs(conv(sin(2*pi*f(j)*t), w1)));
end
```

ovviamente di volta in volta rimpiazzando $w1$ con $w2$, $w3$, ecc. Una volta completato il test, plottare il risultato con il comando

```
>> semilogy(f, hf);
```

Esercizio 3.3. Provare l'effetto dei filtri sulla registrazione `digits-noise.wav`, che è contaminata da rumore bianco in cui le frequenze alte sono predominanti. Quale filtro funziona meglio? Perché?