

A.A. 2020/21

ELABORAZIONE DI DATI E SEGNALI BIOMEDICI - MATLAB

SILVIA FRANCESCA STORTI

FABS :)

NOTA

Questi appunti/sbobinatura/versione “discorsiva” delle slides sono per mia utilità personale, quindi pur avendole revisionate potrebbero essere ancora presenti typos, commenti/aggiunte personali (che anzi, lascio di proposito) e nel caso peggiore qualche inesattezza!

Comunque spero siano utili! 🌸 ✨

**Questa sbobina fa parte della mia collezione di sbobinature,
che è disponibile (e modificabile!) insieme ad altre in questa repo:**

<https://github.com/fabfabretti/sbobinamento-seriale-uniVR>

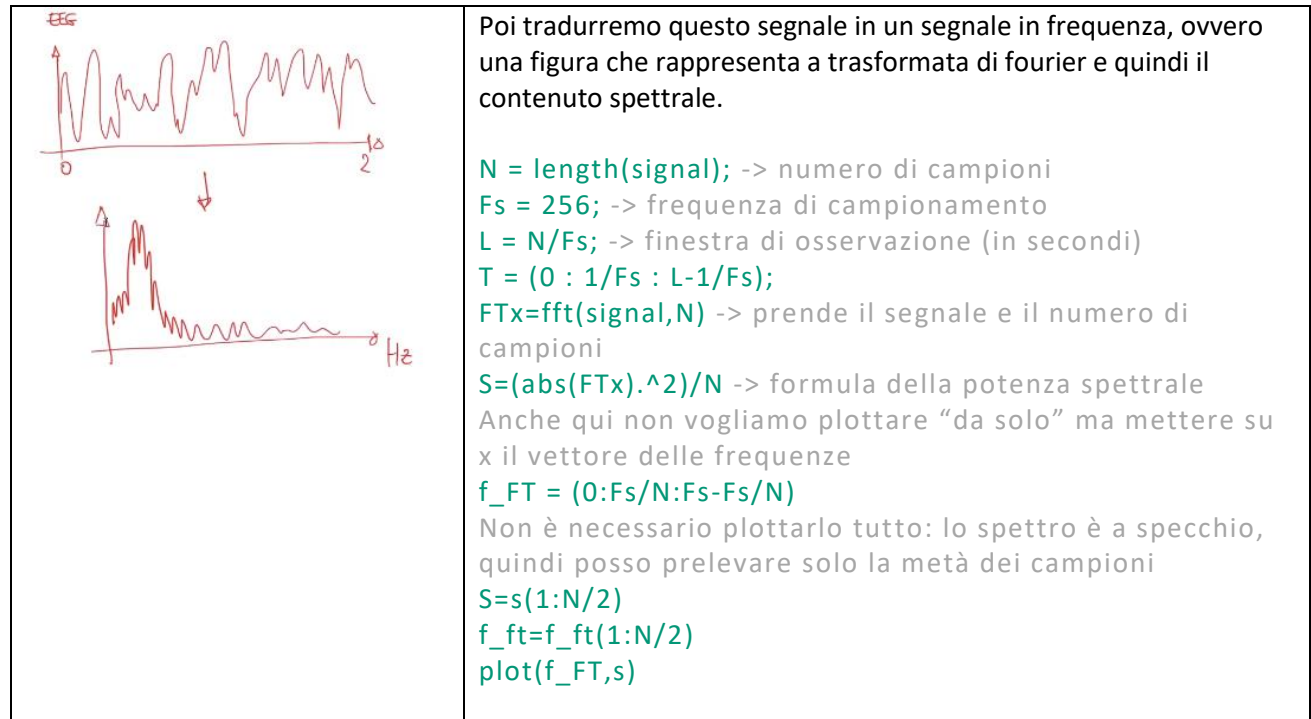
Table of Contents

NOTA	1
Trasformata di Fourier (DFT)	3
Lecture #6 : Funzioni relative a sistemi LTI e filtri (lecture 6).....	4
Lecture #6: Utilizzo e sintesi di filtri	5
Lab #3: EEGLab – Analisi completa via GUI	12
Lab #4: EEGLab – Analisi completa via scripting	17
Lab #5: FSL e FSLeyes – elaborazione di mappe statistiche via GLM	19

Trasformata di Fourier (DFT)

L'intervallo che distanzia due campionamenti è l'intervallo di campionamento, e il suo reciproco la frequenza di campionamento.

In matlab esiste solo la trasformata discreta (obv non possiamo lavorare su sequenze infinite in un calcolatore!)



Lecture #6 : Funzioni relative a sistemi LTI e filtri (lecture 6)

FILTER

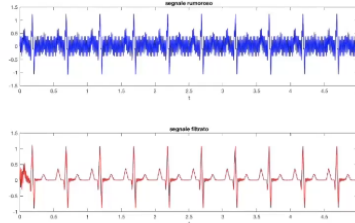
```
b=[b0 b1...bN]
a=[1 a1...aN]
y=filter(b,a,x)
```

I parametri di cui necessita `filter` sono:

- `b` vettore dei coefficienti b_k
- `a` vettore dei coefficienti a_k
- `x` segnale da filtrare

```
a = [1, a1,...,aN];
b = [b0, b1,...,bM];
y = filter(b,a,x);

figure
subplot(211)
plot(x)
title('segnale rumoroso')
subplot(212)
plot(y)
title('segnale filtrato')
```



Usiamo la funzione `filter` per creare e applicare un filtro. Dato in input vettori coi coefficienti + segnale di input, butta fuori il segnale filtrato.

Usiamo `Filter` anziché `Conv` in quanto `Conv`.

ROOTS e POLY

`x = roots(C)` si calcolano le radici del polinomio i cui coefficienti, in ordine di grado decrescente, sono contenuti nel vettore `C`. In uscita si ottiene il vettore delle radici.

Es. Si consideri il sistema ARMA

$$y(n) = 0.8y(n-1) + x(n)$$

$$H(z) = \frac{1}{1 - 0.8z^{-1}} = \frac{z}{z - 0.8}$$

$$\begin{aligned} \text{zeri} &= \text{roots}([1 \ 0]) \\ \text{poli} &= \text{roots}([1 \ -0.8]) \end{aligned}$$

$$\begin{aligned} \text{zeri} &= 0 \\ \text{poli} &= 0.8 \end{aligned}$$

```
poli = roots(a)
zeri = roots(b)
```

`y = poly(A)` si convertono le radici contenute nel vettore `A` in polinomio. In uscita si ottiene il vettore dei coefficienti del polinomio.

Es. Si considerino le radici contenute nei vettori `zeri` e `poli`

$$\text{zeri} = [0 \ 1 \ 1]; \text{poli} = [0];$$

$$\begin{aligned} \text{num} &= \text{poly}(\text{zeri}) \rightarrow \text{num} = [1 \ -2 \ 1 \ 0] \\ \text{den} &= \text{poly}(\text{poli}) \rightarrow \text{den} = [1 \ 0] \end{aligned} \quad \rightarrow \quad H(z) = \frac{z^3 - 2z^2 + z}{z}$$

La funzione `roots` estrude le soluzioni dai coefficienti a_k e b_k . Attenzione a metterli in ordine decrescente!

FREQZ

```
[H,F] = freqz(b,a,N,Fs)
```

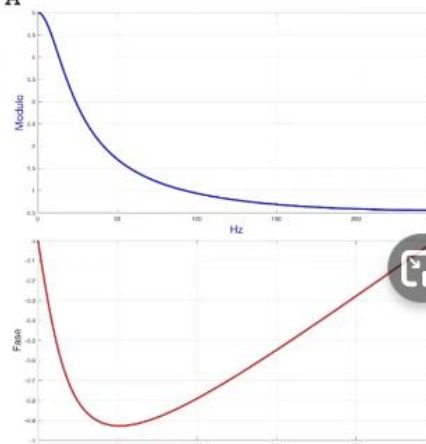
calcola $H(f)$ corrispondente ai polinomi `b` e `a` su `N` campioni, equispaziati fra 0 e $F_s/2$, della risposta in frequenza del sistema ARMA a coefficienti in `B` ed `A`

Es. $N = 1024$, $F_s = 500$ Hz

`freqz` crea un vettore reale `F` e un vettore complesso `H` che contengono 1024 valori equispaziati fra 0 e 250 (= metà della frequenza di campionamento) e i relativi campioni della risposta in frequenza del sistema ARMA

$$y(n) = -0.8y(n-1) + x(n)$$

```
[H,F] = freqz([1], [1-0.8], 1024, 500);
figure(1)
plot(F,abs(H)) modulo di H
xlabel('Hz'); ylabel('Modulo')
figure(2)
plot(F,angle(H)) fase di H
ylabel('Fase'); xlabel('Hz');
```



La funzione `freqz` permette di calcolare la risposta in frequenza dell'ARMA dando campioni e coefficienti. Restituisce un dato `H` che permette di estrarre modulo e fase della risposta in frequenza, usando rispettivamente `abs` e `angle`.

Lecture #6: Utilizzo e sintesi di filtri

1. SINTESI FILTRO DI BUTTERWORTH

- Definisco le specifiche del filtro analogico: $\omega_p, \omega_s, r_p, r_s$
- Cerco l'ordine minimo n ideale da utilizzare nel filtro per rendere l'errore fra filtro ideale e reale, usando la funzione **buttord**
`[n,Wn] = buttord(Wp,Ws,Rp,Rs)`
 - > **Input:** banda passante, banda oscura, ripple in banda passante, ripple in banda oscura.
 - > **Output:** ordine del modello, frequenza di taglio naturale (=mi fa abbassare il tutto di 3 dB)
- Applico la funzione **butter** per ottenere i parametri dell'equazione alle differenze.
`[b,a] = butter(n,Wn)`
 - > **Input:** ordine e frequenza di taglio naturale (calcolati da buttord)
 - > **Output:** vettori dei coefficienti dell'eq alle differenze del filtro.
- Visualizzo l'approssimazione del filtro analogico

2. SINTESI FILTRO DI CHEBYSHEV

- Definisco le specifiche del filtro analogico: $\omega_p, \omega_s, r_p, r_s$
- Cerco l'ordine minimo n ideale da utilizzare nel filtro per rendere l'errore fra filtro ideale e reale, usando la funzione **cheb1ord**
`[n,Wp] = cheb1ord(Wp,Ws,Rp,Rs)`
 - > **Input:** banda passante, banda oscura, ripple in banda passante, ripple in banda oscura.
 - > **Output:** ordine del modello, frequenza di taglio naturale, che è un polinomio
- Applico la funzione **cheby1** per ottenere i parametri dell'equazione alle differenze.
`[b,a] = cheby1(n,Rp,Wp)`
 - > **Input:** ordine e frequenza di taglio naturale (calcolati da cheb1ord), ripple passante
 - > **Output:** vettori dei coefficienti dell'eq alle differenze del filtro.
- Visualizzo l'approssimazione del filtro analogico

3. SINTESI FILTRO ELLITTICO

- Definisco le specifiche del filtro analogico: $\omega_p, \omega_s, r_p, r_s$
- Cerco l'ordine minimo n ideale da utilizzare nel filtro per rendere l'errore fra filtro ideale e reale, usando la funzione **ellipord**. **Posso anche non utilizzarla e fornirlo manualmente secondo le specifiche.**
`[n,Wn] = ellipord(Wp,Ws,Rp,Rs)`
 - > **Input:** banda passante, banda oscura, ripple in banda passante, ripple in banda oscura.
 - > **Output:** ordine del modello, frequenza di taglio naturale, che è un polinomio
- Applico la funzione **cheby1** per ottenere i parametri dell'equazione alle differenze.
`[b,a] = ellip(n,Rp,Rs,Wp)`
 - > **Input:** ordine e frequenza di taglio naturale (calcolati da ellipord), ripple passante
 - > **Output:** vettori dei coefficienti dell'eq alle differenze del filtro.
- Visualizzo l'approssimazione del filtro analogico

Esempio – Esercizio 1

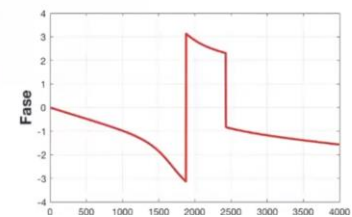
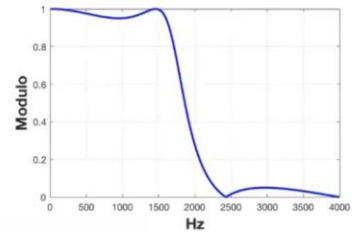
(delta_p e delta_s sono i ripple massimi; dovrò convertirli in decibel)

(Le frequenze di taglio andranno normalizzate)

Otteniamo un ordine ottimo = 3; il filtro è il filtro migliore possibile con le mie richieste.

```
% SPECIFICHE: costanti del progetto
fp = 1600; % banda passante [0,fp] 0 - 1.6kHz
fs = 2400; % banda oscura [fs,Fs/2] 2.4 - 4kHz
Fs = 8000; % frequenza di campionamento
delta_p = 0.95;
delta_s = 0.05;
RpdB = -20*log10(delta_p); % ripple in banda passante (dB)
RsdB = -20*log10(delta_s); % ripple in banda oscura (dB)
Wp = fp/(Fs/2); % freq sup banda passante NORMALIZZATA!
Ws = fs/(Fs/2); % freq inf banda passante NORMALIZZATA!

% stimo l'ordine del filtro
[N,Wn] = ellipord(Wp,Ws,RpdB,RsdB);
% progetto il filtro ellittico
[b,a] = ellip(N,RpdB,RsdB,Wn);
% calcolo la risposta in frequenza
[H,F] = freqz(b,a,1024,Fs);
```



Proviamo a modificare la banda di transizione, stringendola:

```
% Se diminuisce la banda di transizione
fs2 = 1800;
Ws2 = fs2/(Fs/2); % freq inf banda passante NORMALIZZATA!
[N2,wn2] = ellipord(Wp,Ws2,RpdB,RsdB);
[b2,a2] = ellip(N2,RpdB,RsdB,wn2);

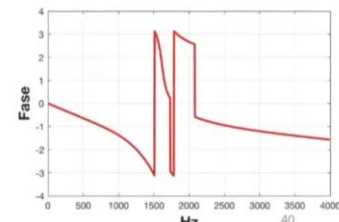
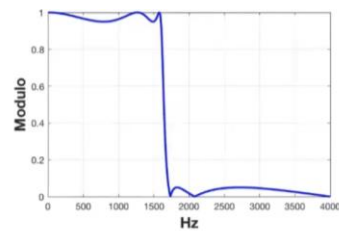
% calcolo la risposta in frequenza
[H2,F2] = freqz(b2,a2,1024,Fs);
```

**SE DIMINUISCE LA
BANDA DI
TRANSIZIONE**

RISULTATI

N2 = 5 → AUMENTA L'ORDINE

```
wn2 = 0.4000
b2 = 0.1147 0.0812 0.1901 0.1901 0.0812 0.1147
a2 = 1.0000 -1.6192 2.3735 -1.6858 0.9173 -0.2138
```



L'ordine del modello aumenta, e aumentano sempre di più le oscillazioni in banda passante e oscura. In generale se voglio diminuire la transizione devo aumentare l'ordine.

Proviamo a ricalcolare per diminuire il ripple:

```
% Se diminuisce anche il ripple in banda passante
delta_p2 = 0.99;
RpdB2 = -20*log10(delta_p2);
[N3,wn3] = ellipord(Wp,Ws2,RpdB2,RsdB);
[b3,a3] = ellip(N3,RpdB2,RsdB,wn3);

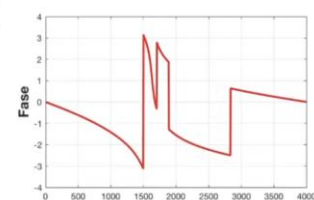
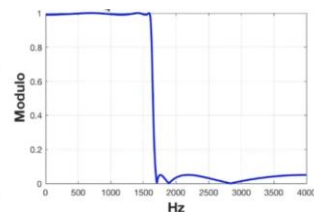
% calcolo la risposta in frequenza
[H3,F3] = freqz(b3,a3,1024,Fs);
```

**SE DIMINUISCE ANCHE
IL RIPPLE IN BANDA
PASSANTE**

RISULTATI

N3 = 6 → AUMENTA L'ORDINE

```
wn3 = 0.4000
b3 = 0.1168 0.0687 0.2696 0.1488 0.2696 0.0687 0.1168
a3 = 1.0000 -1.6341 2.7348 -2.1624 1.5326 -0.5330 0.1317
```



Anche qui aumenta l'ordine! Questo comporta un peggioramento della fase.

1. SINTESI FILTRO PARKS MCCLELLAN

- Definisco le specifiche del filtro analogico: ω_p , ω_s , r_p , r_s
- Cerco l'ordine minimo n ideale da utilizzare nel filtro per rendere l'errore fra filtro ideale e reale, usando la funzione **firpmord** o imponendolo noi.

[n,fo,ao,w] = firpmord(f,a,dev)

- > **Input:** f frequenza di taglio, a ampiezze desiderate, dev = $[(10^{(rp/20)}-1)/(10^{(rp/20)}+1) \cdot 10^{(-rs/20)}]$ con rp passband ripple in dB e rs stopband ripple in dB.
- > **Output:** ordine del modello, frequenza normalizzata fo, ampiezza della banda in frequenza ao, pesi.

- Applico la funzione **firpm** per ottenere i parametri dell'equazione alle differenze.

b = firpm(n,f,a)

Ha diverse opzioni possibili, fra cui quella di restituire un parametro err che fornisce l'errore massimo.

- > **Input:** ordine calcolato da firpmord, un vettore f che contiene le frequenze $[0 \ \omega_p \ \omega_s \ \pi]$, vettore a che contiene le ampiezze $[1 \ 1 \ 0 \ 0]$ in
- > **Output:** vettore dei coefficienti dell'eq alle differenze del filtro.

- Visualizzo l'approssimazione del filtro analogico

Esempio – Esercizio 2

In questo caso abbiamo scelto manualmente l'ordine del modello.

```
% SPECIFICHE: costanti del progetto
fp = 1600; % banda passante [0,fp] 0 - 1.6kHz
fs = 2400; % banda oscura [fs,Fs/2] 2.4 - 4kHz
Fs = 8000; % frequenza di campionamento
delta_p = 0.95;
delta_s = 0.05;
RpdB = -20*log10(delta_p); % ripple in banda passante (dB)
RsdB = -20*log10(delta_s); % ripple in banda oscura (dB)
Wp = fp/(Fs/2); % freq sup banda passante normalizzata
Ws = fs/(Fs/2); % freq inf banda passante normalizzata

f = [0 Wp Ws 1];
m = [1 1 0 0];
n = 5; % ordine del filtro

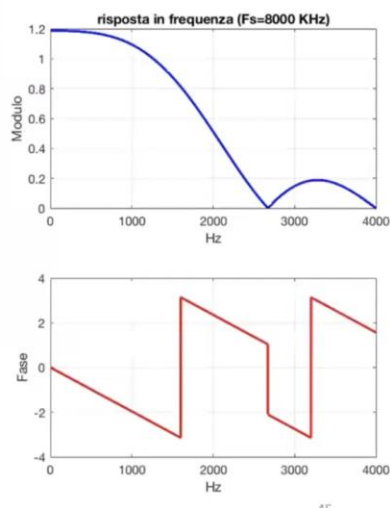
% progetto il filtro FIR
[b,err] = firpm(n,f,m);

% calcolo la risposta in frequenza
[H,F] = freqz(b,1,Fs/2,Fs);
```

```
% visualizzo modulo e fase della risposta in frequenza
figure(1)
subplot(211)
plot(F,abs(H),'b')
xlabel('Hz')
ylabel('Modulo')
title(['risposta in frequenza (Fs=' num2str(Fs) ' KHz)']);
grid on
subplot(212)
plot(F,angle(H),'r')
ylabel('Fase')
xlabel('Hz')
grid on
```

RISULTATI

```
b = -0.0807 0.1967 0.4777 0.4777 0.1967 -0.0807
err = 0.1874
```



Se provo ad aumentare l'ordine del filtro noto che diminuisce l'errore.

```
%% Se aumento l'ordine del filtro
n2 = 15; % ordine del filtro

% progetto il filtro FIR
[b2,err2] = firpm(n2,f,m);

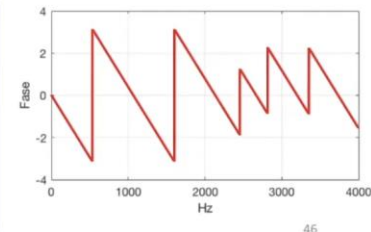
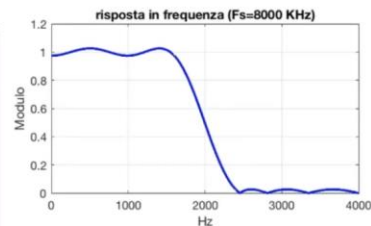
% calcolo la risposta in frequenza
[H2,F2] = freqz(b2,1,Fs/2,Fs);
```

**SE AUMENTO
L'ORDINE DEL FILTRO**

RISULTATI

```
b2 = Columns 1 through 13
-0.0117 -0.0237 0.0288 0.0356 -0.0539 -0.0820
0.1460 0.4480 0.4480 0.1460 -0.0820 -0.0539
0.0356
Columns 14 through 16
0.0288 -0.0237 -0.0117

err2 = 0.0257 → DIMINUISCE L'ERRORE
```



Se provo a diminuire la banda di trasmissione, a parità di ordine del filtro noto un aumento dell'errore (per farla venire più ripida devo generare più ripple tipo:

```
%% Se si riduce anche la banda di transizione
fs2 = 1800; % banda oscura [fs,Fs/2] 2.4 - 4kHz
Ws2 = fs2/(Fs/2); % freq inf banda passante NORMALIZZATA!
f2 = [0 Wp Ws2 1];

% progetto il filtro FIR
[b3,err3] = firpm(n2,f2,m);

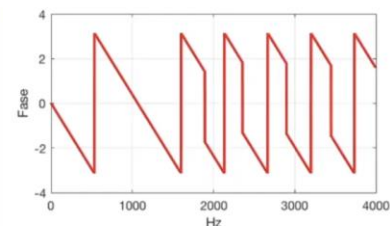
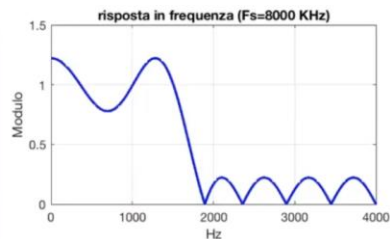
% calcolo la risposta in frequenza
[H3,F3] = freqz(b3,1,Fs/2,Fs);
```

**SE SI RIDUCE ANCHE LA
BANDA DI TRANSIZIONE**

RISULTATI

```
b3 = Columns 1 through 8
-0.0542 0.1122 0.0821 -0.0075 -0.0869 -0.0231
0.1935 0.3946
Columns 9 through 16
0.3946 0.1935 -0.0231 -0.0869 -0.0075 0.0821
0.1122 -0.0542

err3 = 0.2214 → AUMENTA L'ERRORE
```



Provo a dare più peso ai ripple in banda passante (= lo riduco di più) che a quelli in banda oscura:

```
%% Se si riduce il ripple in banda passante
w1 = 10^1;

% progetto il filtro FIR
[b4,err4] = firpm(n2,f2,m,w1);

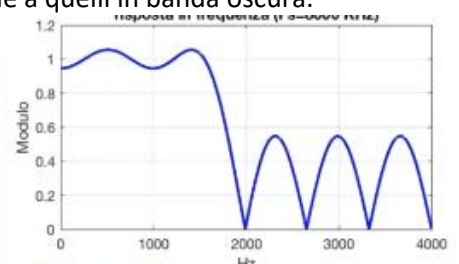
% calcolo la risposta in frequenza
[H4,F4] = freqz(b4,1,Fs/2,Fs);
```

**SE SI RIDUCE IL RIPPLE
IN BANDA PASSANTE**

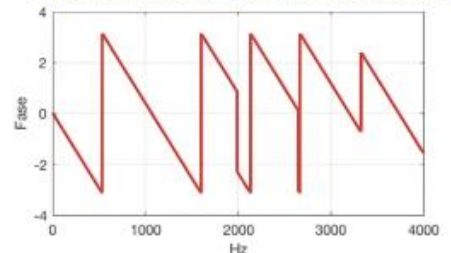
RISULTATI

```
b4 = Columns 1 through 13
-0.0107 -0.1113 0.1829 -0.0452 -0.1124 -0.0240
0.1991 0.3942 0.3942 0.1991 -0.0240 -0.1124
-0.0452
Columns 14 through 16
0.1829 -0.1113 -0.0107

err4 = 0.5466
```



→ AUMENTA IL RIPPLE IN BANDA OSCURA



Aumenta il ripple in banda oscura! Vale il viceversa.

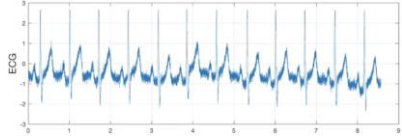
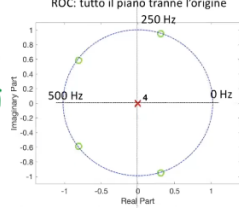
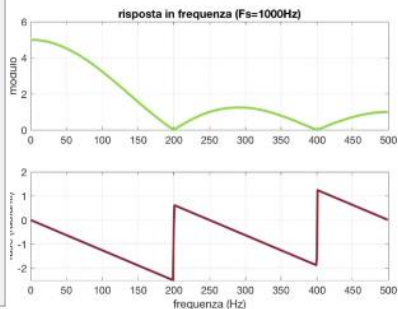
Filtraggio F-B (=senza cambio di fase)


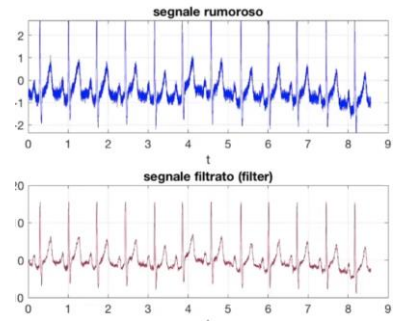
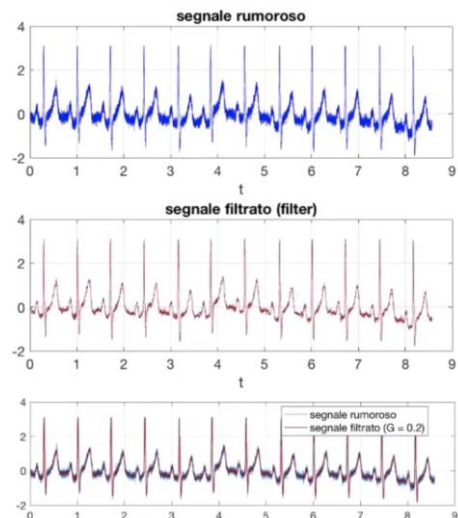
```
b=[b0 b1...bM]  
a=[1 a1...aN]  
y=filtfilt(b,a,x)
```

È sufficiente utilizzare la funzione `filtfilt`.

Esempio – Esercizio 3

Applicare il filtro FIR $y(n) = x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-4)$ alla traccia di un ECG fornito [DatiEs3_ecg_hf_calgaryuni.mat]

<p>1. Caricare il segnale ECG e visualizzarlo</p>	<p>Fs = 1000 Hz</p> <pre>%% 1. Carico il tracciato ECG da filtrare load('DatiEs3_ecg_hf_calgaryuni.mat'); Fs = 1000; % frequenza di campionamento Tc = 1/Fs; % periodo di campionamento t = (1:numel(ecg))/Fs; % vettore dei tempi % visualizzo il tracciato ECG grezzo figure(1) plot(t,ecg) xlabel('Time (s)','fontsize',16) ylabel('ECG','fontsize',16) grid on</pre> 
<p>2. Scrivere la funzione di trasferimento e verificare la stabilità del sistema</p>	<p>Risolvere il filtro come rapporto</p> $H(z) = (1 + z^{-1} + z^{-2} + z^{-3} + z^{-4}) \cdot \frac{z^4}{z^4}$ $\Rightarrow \frac{z^4 + z^3 + z^2 + z + 1}{z^4}$ <pre>zeri = roots[1 1 1 1 1]; poli = roots[1 0 0 0 0]; figure(2) zplane(zeri,poli)</pre>  <p>ROC: tutto il piano tranne l'origine</p> <p>I poli sono tutti in 0 e hanno quindi modulo < 1 → SISTEMA STABILE.</p>
<p>3. Valutare modulo e fase della risposta in frequenza</p>	<pre>[H,F]=freqz(B,A,N,Fs) !!! ATTENZIONE: Freqz prende il vettore dei coefficienti valutati come rapporto di polinomi in z^-1. Roots, invece, vuole polinomi in z.</pre> <pre>B = [1 1 1 1 1]; A = 1; N = 500; [H,F] = freqz(B,A,N,Fs); figure(3) subplot(211) plot(F,abs(H)) title(['risposta in frequenza', (Fs= num2str(Fs) 'Hz')]); ylabel('modulo') subplot(212) plot(F,angle(H)); ylabel('fase (radianti)') xlabel('frequenza (Hz)')</pre>  <p>→ Notiamo che il filtro amplifica le frequenze basse e attenua le frequenze alte. Si annulla esattamente a 200 e 400 Hz.</p>

<p>4. Capire dalla risposta in frequenza il motivo della risposta nulla (a regime) al segnale sinusoidale per certe frequenze critiche, tipo 200 Hz.</p>	<div data-bbox="437 271 956 763" data-label="Text"> <pre>%% 4. Filtraggio di un segnale sinusoidale Fn = 200; % frequenza critica x_sin = sin(2*pi*Fn*t); y_sin = filter(B,A,x_sin); figure (4) subplot(211) plot(t(1:200),x_sin(1:200)) title('input') xlabel('t (s)') subplot(212) plot(t(1:200),y_sin(1:200)) title('output') xlabel('t (s)')</pre> </div> <div data-bbox="986 232 1528 837" data-label="Figure"> <p>Consideriamo la frequenza critica $F_n = 200\text{Hz}$.</p>  <p>Se la passiamo attraverso il filtro succede l'impensabile:</p> <p>Risposta nulla (a regime) al segnale sinusoidale per certe frequenze critiche</p> <p>È colpa dei ripple del filtro che ci mandano a 0 in quella frequenza :(</p> </div>
<p>5. Applicare il filtro all'ECG e confrontare i segnali in ingresso e uscita</p>	<p>È d'uopo, per abitudine, togliere la baseline(media) dal segnale d'ingresso.</p> <div data-bbox="437 920 1086 1346" data-label="Text"> <pre>%% 5. Filtraggio del segnale ECG % tolgo la media al segnale ecg2 = ecg - mean(ecg); y = filter(B,A,ecg2); figure (4) subplot(211) plot(t,ecg2) title('segnale rumoroso') xlabel('t') subplot(212) plot(t,y) title('segnale filtrato (filter)') xlabel('t')</pre> </div> <div data-bbox="1114 913 1513 1234" data-label="Figure">  </div>
<p>6. Ottenere segnale filtrato con guadagno unitario</p>	<p>Notiamo a destra che le scale di ampiezza del segnale sono differenti: poiché il guadagno al centro della banda passante è pari a 5, l'uscita varia su una scala maggiore [+10 -10] dell'ingresso [-2 +2]. Per avere guadagno unitario basta moltiplicare tutto per 0.2.</p> <div data-bbox="437 1469 1031 2007" data-label="Text"> <pre>G = 0.2; ecgfilt1 = filter(G*B,A,ecg2); figure (5) subplot(211) plot(t,ecg2) title('segnale rumoroso') xlabel('t') subplot(212) plot(t,ecgfilt1) title('segnale filtrato (filter)') xlabel('t') figure(6) hold on plot(t,ecg2) plot(t,ecgfilt1) xlabel('t') legend('segnale rumoroso', 'segnale filtrato (G = 0.2)') grid on; box on;</pre> </div> <div data-bbox="1058 1469 1517 1984" data-label="Figure">  </div>

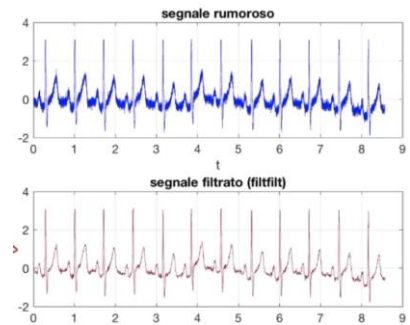
7. Implementiamo il filtro in versione F-B e confrontiamo col filtro normale

Tutto uguale ma uso `filtfilt` anziché `filter`.

```
%% 7. filtraggio del segnale usando F
filtfilt

G = 0.2;
ecgfilt2 = filtfilt(G*B,A,ecg2);

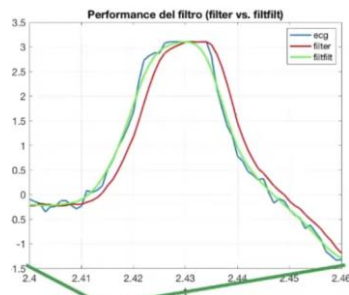
figure (7)
subplot(211)
plot(t,ecg2)
title('segnale rumoroso')
xlabel('t')
subplot(212)
plot(t,ecgfilt2)
title('segnale filtrato (filtfilt)')
xlabel('t')
```



Valutazione nel tempo

Capire esattamente la differenza con quello di prima può essere complesso; analizziamo la performance del filtro `filter` vs `filtfilt` prelevando una piccola finestra e zoommandola.

```
% Valutazione della performance del filtro
(filter vs. filtfilt)
figure (8)
plot(t,ecg2,t,ecgfilt1,'r',t,ecgfilt2,'g')
title('Performance del filtro (filter vs.
filtfilt)'),
xlabel('t'),
xlim([2.4 2.46]),
legend('ecg','filter','filtfilt');
```



Notiamo che il filtro `filtfilt` evita che il segnale filtrato risulti ritardato rispetto all'originale.

Valutazione nella risposta in frequenza

Calcoliamo la risposta in frequenza come prodotto fra H e il suo coniugato. Notiamo la fase

lineare del `filtfilt`.

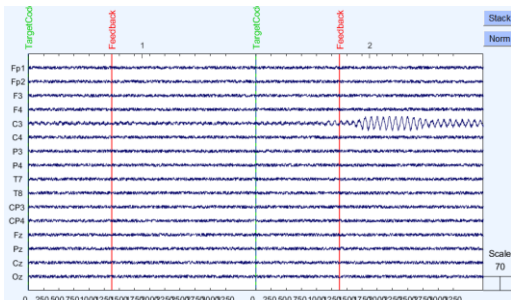
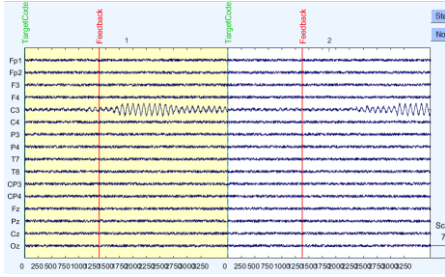
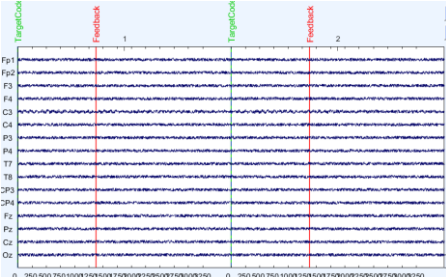
```
H_FB=H.*conj(H);

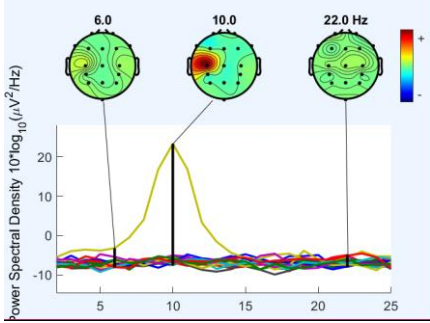
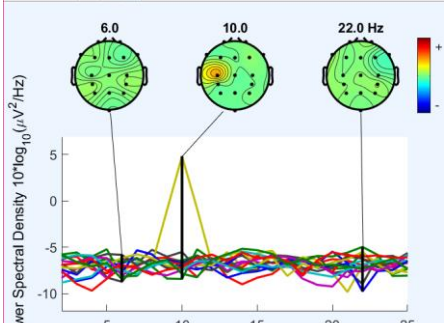
figure (9)
subplot(211),
plot(F,abs(H),F,abs(H_FB),'r')
ylabel('modulo'),
legend('filter','filtfilt');
subplot(212),
plot(F,angle(H),F,angle(H_FB),'r'),
xlabel('frequenza Hz'),
ylabel('fase (radianti)'),
legend('filter','filtfilt');
```


Lab #3: EEGlab – Analisi completa via GUI

Es.1 : Estrarre epoche da segnale simulato.

Segnale simulato vuol dire che si parte da un segnale di fondo a cui si aggiunge una componente a 10 Hz su alcuni canali (in questo caso solo il C3, quinta traccia, su alcune porzioni). Il file ha una matrice di 16x46080 canali, dove 16 è il numero di canali e il resto il numero di campioni. La frequenza di campionamento è 256 Hz

<p>Estraiano le epoche</p> <p>Tools → Extract epochs</p>	<ul style="list-style-type: none"> - Time-locking event type: è un'impostazione che ci viene data - Epoch limits: determina gli intervalli di tempo da amntenere prima e dopo il marker selezionato (noi mettiamo 0,3.75 quindi partiamo dal marker e guardiamoo i successivi 3.75 secondi) <ul style="list-style-type: none"> ▪ Name: sarà un nuovo dataset in cui sono presenti i nuovi trial. - Epoch limits: possiamo specificare dove c'è la parte di pausa pre-task; serve a rimuovere la baseline, ovvero la media del segnale da ogni singola epoca Baseline: è l'attività di base di riferimento. Qui sono millisecondi!! Con lo script possiamo anche toglierla da tuttoil segnale anziché solo da una eèpca- <ul style="list-style-type: none"> ▪ Fa la media dalla zona a riposo e la rimuove dall'epoca - Questo genera un nuovo dataset, separato dal primo. Questo ci permette di avere un versionamento del dataset e di tornare al tracciato grezzo.
<p>Visualizzazione plot</p> <p>Plot→channel data scroll</p>	<ul style="list-style-type: none"> - Ora che abbiamo estratto le epoche, la visualizzazione via mostrerà il tempo diviso in epoche (e non solo in ms). <p>Osservando le epoche noto che solo alcune epoche hanno la modulazione in banda alfa.</p> 
<p>Distinzione epoche</p> <p>Edit→select epoch or event</p>	<p>Abbiamo delle epoche in posizione 1 o 2;</p> <p>1= no attività 2= sì attività</p> <p>Le salvo in due dataset separati (quindi quando estraggo i 2 devo tornare a lavorare sul dataset intero)</p> <div style="display: flex; justify-content: space-around;"> <div data-bbox="491 1686 938 1995"> <p>1: solo epoche no banda alfa</p>  </div> <div data-bbox="970 1686 1417 1995"> <p>2: solo epoche con banda alfa</p>  </div> </div>

<p>Analisi in frequenza</p> <p>Plot→channel spectra and maps</p>	<ul style="list-style-type: none"> - Siamo interessati solo alla parte eseguita, quindi vogliamo eliminare il pretrial. - Le mappe non sono normalizzate, ma mettendo il limite con 'maplimits' posso dare lo stesso limite ad entrambe e renderle confrontabili. <p>Epoch time range to analyze (ms): [1250 3746] (to remove the baseline activity before the cursor appears)</p> <p>Percent data to sample: 100</p> <p>Frequencies to plot as scalp maps (Hz): 6 10 22</p> <p>Spectral and scalp map option: 'electrodes', 'on', 'maplimits', [-25 25]</p> <div style="display: flex; justify-content: space-around;"> <div data-bbox="491 539 922 898"> <p>Epoche 1 (banda alfa):</p>  </div> <div data-bbox="970 539 1417 898"> <p>Epoche 2 (no banda alfa):</p>  </div> </div>
---	--

Via script:

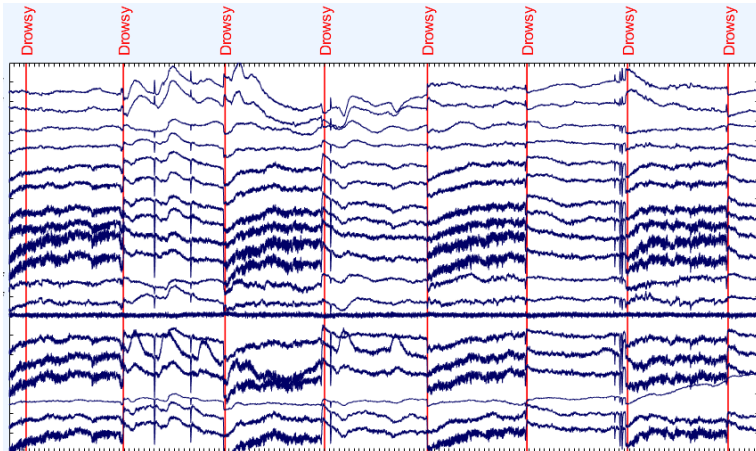
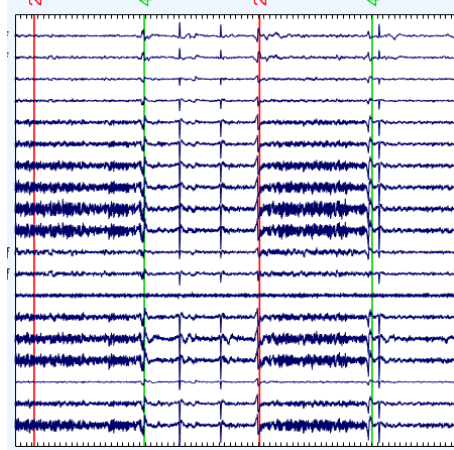
```
% 1. Apro l'interfaccia e creo le variabili
[ALLEEG EEG CURRENTSET ALLCOM] = eeglab;
EEG = pop_loadset('TestData.set');



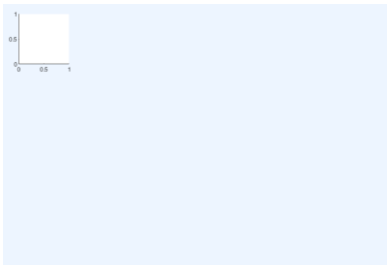
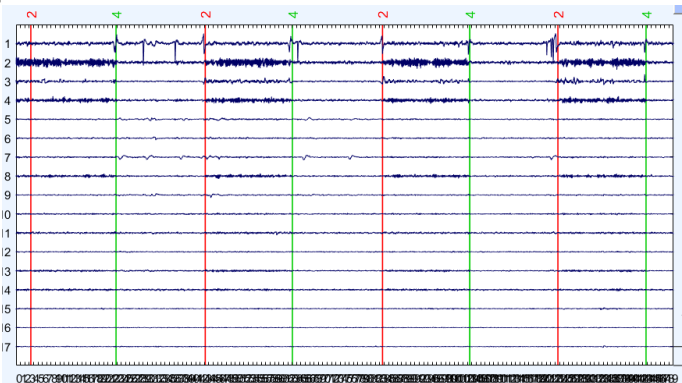

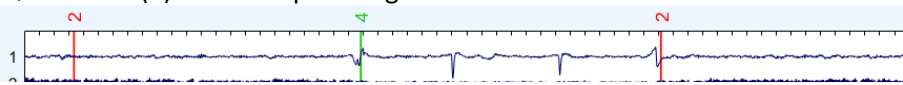
% Plotto il dato e posso aggiungere cose:
% - sample rate: srate
% - canale locations: eloc_file
% - eventi: events
% - lunghezza finestra temporale: winlength
% - colore tracce: color + {'k'}
eegplot(EEG.data, 'srate', 256, 'eloc_file', EEG.chanlocs, 'events', EEG.event, 'winlength', 15, 'color', {'k'});

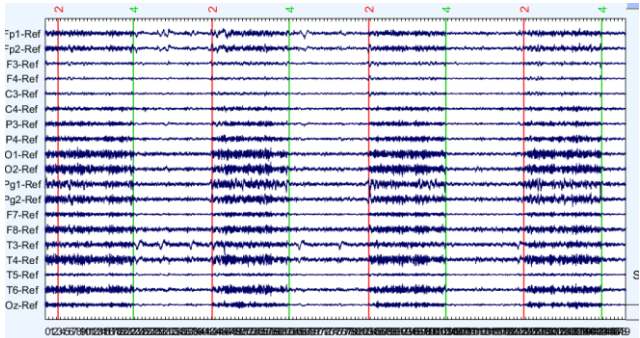
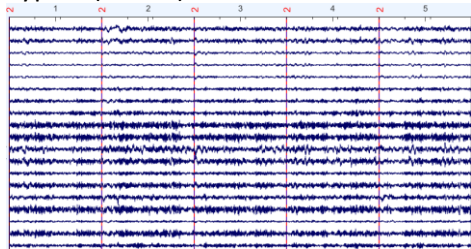
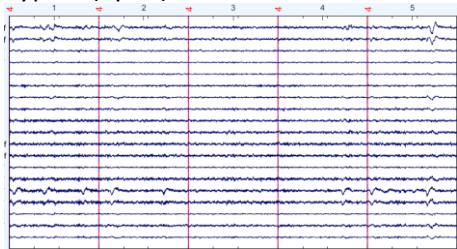
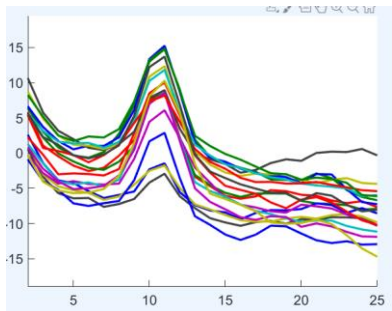
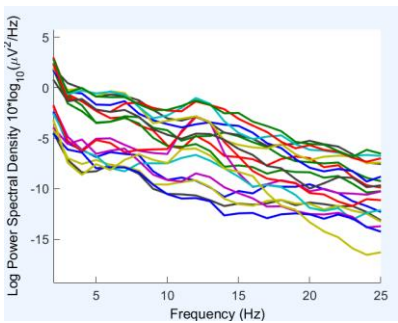
% Estrazione epoche:
% - newname: da il nuovo nome
% - epochinfo: mi da info sulle epoche
EEG=pop_epoch(EEG, {'TargetCode'}, [0 3.75], 'newname', 'Epochs', 'epochinfo', 'yes');
%Ora il trial ha dimensioni diverse

%Per aggiornare l'interfaccia uso redraw
eeglab redraw
```

Es.2 : Analisi di segnale REALE.

<div>Carichiamo file EDF+</div> <div>Load→ BIOSIG</div> <div>EEG = pop_biosig('data_s ub1_EOEC.edf');</div>	<div>Dato di tipo edf+, ovvero al suo interno ha dati riguardo agli eventi</div> <div><ul style="list-style-type: none">- 35x61750 con sr=250 Hz- Ci interessano solo alcuni canali 1:10 13:18 20:22</div> <div></div> <div>Mooolto rumoroso (es. componenti lente)</div>																																																								
<div>Carichiamo locazioni</div> <div>Edit→Channel location</div> <div>Matlab a dx</div>	<div>Carichiamo le locations, salvate nel file sfp</div> <div>EEG.chanlocs = pop_chanedit(EEG, 'load', {'elettrodes_10_20_SEI.sfp', 'filetype', 'autodetect'});</div> <div>[ALLEEG EEG CURRENTSET] = eeg_store(ALLEEG, EEG)</div>																																																								
<div>Ridenominazione eventi</div> <div>(mini script preso dalle slide)</div>	<div><table><thead><tr><th>type</th><th>edftype</th><th>latency</th><th>urevent</th></tr></thead><tbody><tr><td>'Drowsy'</td><td>1</td><td>844</td><td>1</td></tr><tr><td>'Drowsy'</td><td>1</td><td>5706</td><td>2</td></tr><tr><td>'Drowsy'</td><td>1</td><td>10792</td><td>3</td></tr><tr><td>'Drowsy'</td><td>1</td><td>15766</td><td>4</td></tr><tr><td>'Drowsy'</td><td>1</td><td>20908</td><td>5</td></tr><tr><td>'Drowsy'</td><td>1</td><td>25882</td><td>6</td></tr><tr><td>'Drowsy'</td><td>1</td><td>30912</td><td>7</td></tr><tr><td>'Drowsy'</td><td>1</td><td>35941</td><td>8</td></tr><tr><td>'Drowsy'</td><td>1</td><td>41027</td><td>9</td></tr><tr><td>'Drowsy'</td><td>1</td><td>45945</td><td>10</td></tr><tr><td>'Drowsy'</td><td>1</td><td>51031</td><td>11</td></tr><tr><td>'Drowsy'</td><td>1</td><td>56061</td><td>12</td></tr><tr><td>'Drowsy'</td><td>1</td><td>61426</td><td>13</td></tr></tbody></table></div> <div><div>Command Window</div><div><pre>for i=1:2:13 EEG.event(i).type = 2; EEG.urevent(i).type = 2; end for i=2:2:13 EEG.event(i).type = 4; EEG.urevent(i).type = 4; end</pre></div><div>eyes closed (EC)</div><div>eyes open (EO)</div><div>Per uniformarli li rinominiamo :)</div></div>	type	edftype	latency	urevent	'Drowsy'	1	844	1	'Drowsy'	1	5706	2	'Drowsy'	1	10792	3	'Drowsy'	1	15766	4	'Drowsy'	1	20908	5	'Drowsy'	1	25882	6	'Drowsy'	1	30912	7	'Drowsy'	1	35941	8	'Drowsy'	1	41027	9	'Drowsy'	1	45945	10	'Drowsy'	1	51031	11	'Drowsy'	1	56061	12	'Drowsy'	1	61426	13
type	edftype	latency	urevent																																																						
'Drowsy'	1	844	1																																																						
'Drowsy'	1	5706	2																																																						
'Drowsy'	1	10792	3																																																						
'Drowsy'	1	15766	4																																																						
'Drowsy'	1	20908	5																																																						
'Drowsy'	1	25882	6																																																						
'Drowsy'	1	30912	7																																																						
'Drowsy'	1	35941	8																																																						
'Drowsy'	1	41027	9																																																						
'Drowsy'	1	45945	10																																																						
'Drowsy'	1	51031	11																																																						
'Drowsy'	1	56061	12																																																						
'Drowsy'	1	61426	13																																																						
<div>Primo filtraggio</div> <div>Tools→Filter→ Basic FIR filter.</div> <div>EEG = pop_eegfilt(EEG, 0,30,[],[0],0,1,'f ir1',0); EEG = pop_eegfilt (EEG, 1,0,[],[0],0,1,'fi r1',0);</div>	<div>Non è un filtro ideale, quindi non perfetto, ma tendenzialmente riusciamo ad applicare un filtro passabanda che mantiene 1-30 Hz. Ora è molto più pulito!!!</div> <div><ul style="list-style-type: none">- Bande troppo lente sono weird- Bande troppo alte sono probabilmente contrazioni muscolari.</div> <div>Sono ancora presenti artefatti:</div> <div><ul style="list-style-type: none">- Chiusura occhi: si genera attività ampia in banda alfa.- Blinking: si vedeeeee</div> <div>Possiamo capire che è blinking se abbiamo un video accanto all'EEG.</div> <div>Se la frequenza fosse stata sotto i 30Hz avrei avuto bisogno di un filtro notch(??)</div> <div></div>																																																								

Cambiare la referenza fra elettrodi Tools→Rereference→ Compute average reference	→ Calcolo rispetto alla media Tools->interpolate electrodes → Un elettrodo non registra bene (es. perde aderenza): se l'artefatto persiste su tutta la registrazione conviene interpolare le tracce nelle vicinanze.
Analisi delle componenti indipendenti Tools→Decompose data by ICA→fastICA.	Togliere il blinking con il passabanda è difficile perché il blinking sta proprio nelle frequenze che mi interessano. Devo fare una scomposizione. Quindi scompongo il segnale e seleziono le componenti del rumore.
Scomposizione e selezione delle componenti Tools→Classify components using ICLabels→View extended components properties	Proprio <i>visivamente</i> dovrò riconoscere le componenti sminchiate e dirgli di levarle.  Eeglab mi chiede dei parametri che alla prof non chiede, e poi mi manda a quel paese.  
Visualizzazione delle componenti Plot→ component activation scroll	L'algoritmo ICA può creare ad ogni iterazione componenti con numeri diversi!! Quindi i numeri a sinistra possono variare da una iterazione all'altra.  Questa componente (2) modula bene la condizione occhi aperti occhi chiusi:  Quest'altra (1) invece è quella legata al blink:  infatti <ul style="list-style-type: none"> - Non c'è in corrispondenza di occhi chiusi - Potenziale stranino quando chiude gli occhi (ha senso, il movimento è simile anche se volontario)

<p>Rimuovere le componenti brutte</p> <p>Tools → Remove components from data</p>	<p>Non c'è piuuuuuuuu!!!! :D</p> 
<p>Estraiamo epoche</p> <p>Tools → extract epochs</p>	<p>Se ci chiede roba di baseline lasciamo tutto vuoto per rimuovere la media da tutto il tracciato automaticamente :)</p> <p>Prendiamo epoche da 18 secondi (lunghette, ma whatever bro)</p>
<p>Estraiamo le epoche separatamente per tipo</p> <p>Edit → select epochs or events</p>	<p>Prendiamo prima solo le type 2 e poi solo le type 4.</p> <div data-bbox="754 772 1126 994"> <p>Dataset 1:Raw dataset</p> <p>Dataset 2:FIR filtered dataset</p> <p>Dataset 3:FIR filtered + remove blink</p> <p>Dataset 4:EOEC epochs</p> <p>Dataset 5:EOEC epochs</p> <p>Dataset 6:EOEC T2 (closed) only</p> <p><input checked="" type="checkbox"/> Dataset 7:EOEC T4 (open) only</p> </div> <div style="display: flex; justify-content: space-around;"> <div data-bbox="395 1001 868 1279"> <p>Type 2 (closed)</p>  </div> <div data-bbox="895 1001 1353 1279"> <p>Type 4 (open)</p>  </div> </div>
<p>Analisi in frequenza</p> <p>Plot → Channel Spectra and Maps</p>	<div data-bbox="560 1317 1318 1525"> <p>Epoch time range to analyze [min_ms max_ms]: 0 17996</p> <p>Percent data to sample (1 to 100): 100</p> <p>Frequencies to plot as scalp maps (Hz): 6 11 22</p> <p>Apply to EEG ERP BOTH: EEG</p> <p>Plotting frequency range [lo_Hz hi_Hz]: 2 25</p> <p>Spectral and scalp map options (see topoplot): 'electrodes','on','maplimits',[-15 15]</p> </div> <ul style="list-style-type: none"> - Epoch time: 18 secondi come prima - 100% da analizzare - (a me non faceva scegliere le frequenze tbf) <div style="display: flex; justify-content: space-around;"> <div data-bbox="395 1704 788 2051"> <p>Type 2 (closed)</p>  </div> <div data-bbox="895 1704 1294 2063"> <p>Type 4 (open)</p>  </div> </div>

Lab #4: EEGlab – Analisi completa via scripting

Es.1 : Fai l'esercizio precedente via scripting

```
%% Caricamento file
% Apro l'interfaccia e creo le variabili
[ALLEEG EEG CURRENTSET ALLCOM] = eeglab;

% Carico il file .edf solo nei canali 1-10,13-18,20-22
EEG = pop_biosig('data_sub1_EOEC.edf','channels',[1 2 3 4 5 6 7 8 9 10 13 14 15 16 17 18 20 21 22]);

%Carico le locatione lo salvo in chanlocs
EEG.chanlocs = pop_chanedit(EEG,'load',{'elettrodes_10_20_SEI.sfp','filetype','autodetect'});

%Memorizziamo il dataset in eeglab (crea il nuovo dataset in ALLEEG)
[ALLEEG EEG CURRENTSET] = eeg_store(ALLEEG, EEG);

%Modifico il label degli eventi
for i=1:2:13
    EEG.event(i).type= 2;EEG.urevent(i).type= 2;
end
for i=2:2:13
    EEG.event(i).type= 4;EEG.urevent(i).type= 4;
end

eeglab redraw

%Disegno un plot
eegplot(EEG.data,'srate',256,'eloc_file',EEG.chanlocs,'events',EEG.event,'winlength',150,'color',{'k'});

%% Filtraggio FIR 1-30Hz

%%Applico passa basso e passa alto
EEG = pop_eegfilt(EEG, 0,30,[],[0],0,1,'fir1',0);
EEG = pop_eegfilt(EEG, 1,0,[],[0],0,1,'fir1',0);

%Salviamo nuovo dataset
[ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG,EEG,CURRENTSET, 'setname','filtered');
eeglab redraw

%Plot del tracciato corrente; la precedente viene chiusa, dunque per
%vederla devo fare ALLEEG(1).data come primo parametro
eegplot(EEG.data,'srate',256,'eloc_file',EEG.chanlocs,'events',EEG.event,'winlength',150,'color',{'k'});

%% Rimozione della baseline

%Rimuovo la media
EEG = pop_rmbase(EEG,[],[]);

%Salviamo nuovo dataset
[ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG,EEG,CURRENTSET, 'setname','Baseline removed');
eeglab redraw

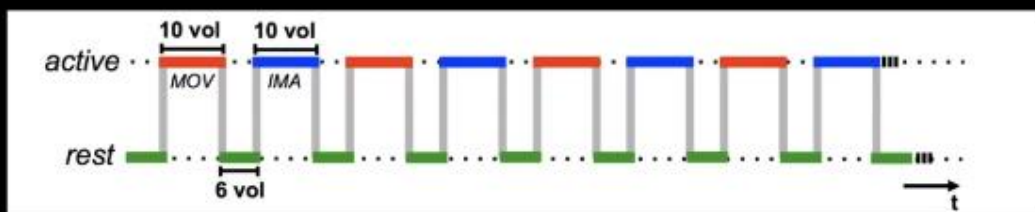
%Plot
eegplot(EEG.data,'srate',256,'eloc_file',EEG.chanlocs,'events',EEG.event,'winlength',150,'color',{'k'});
```

<pre>%% Rireferenziazione usando la referencia media %Rireferenzio sulla media (default) EEG = pop_reref(EEG, []); %Salviamo nuovo dataset [ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG, EEG, CURRENTSET, 'setname', 'Rereferenced'); eeglab redraw</pre>	
<pre>%% Interpolazione canali %Interpolazione; diamo in ingresso il numero della traccia (non il label!!) %EEG = pop_interp(EEG,13,'spherical'); %Salviamo nuovo dataset [ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG, EEG, CURRENTSET, 'setname', 'Interpolation'); eeglab redraw</pre>	
<pre>%% Analisi ICA %selfexplanatory [EEG.com] = pop_runica(EEG, 'icatype', 'fastica'); %Salviamo nuovo dataset [ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG, EEG, CURRENTSET, 'setname', 'Interpolation'); eeglab redraw %Plottiamo le componenti. % !!! pop_eegplot != eegplot!!! pop_eegplot(EEG, 0, 0, 0); %Plottiamo le mappetopografiche con pop_topoplot %pop_topoplot(EEG, 0, [1:size(EEG.icawinv, 2)], 'EEG_FIL_BAS_AVE_INT_ICA', [4 5], 0);</pre>	
<pre>%% Rimuoviamo la componente brutta (2) %Rimuoviamo EEG = pop_subcomp(EEG, 2, 0); %Plot eegplot(EEG.data, 'srate', 256, 'eloc_file', EEG.chanlocs, 'events', EEG.event, 'winlength', 150, 'color', {'k'});</pre>	

Lab #5: FSL e FSLeves – elaborazione di mappe statistiche via GLM

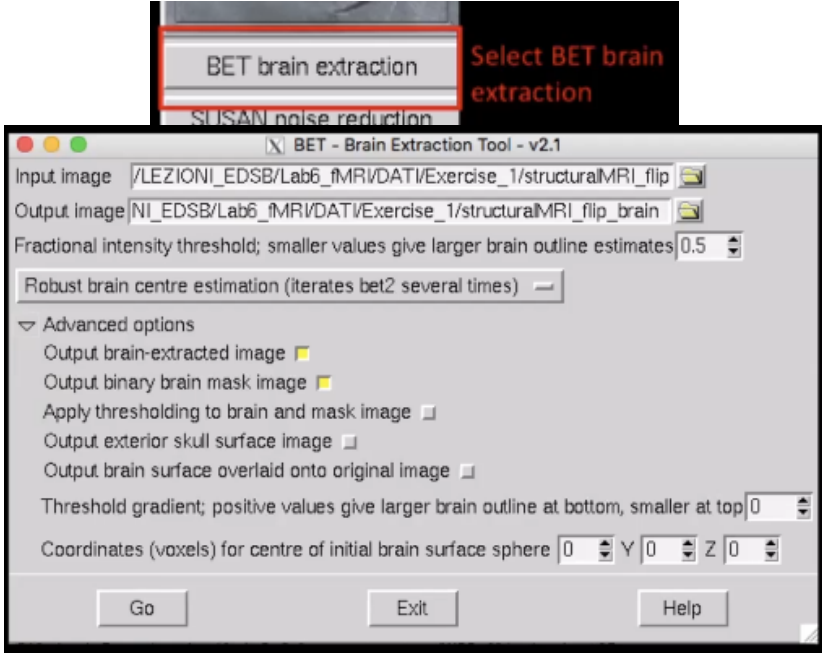
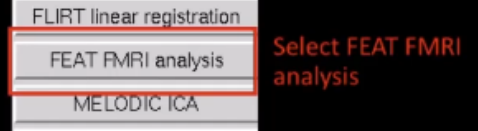
EXPERIMENTAL DESIGN

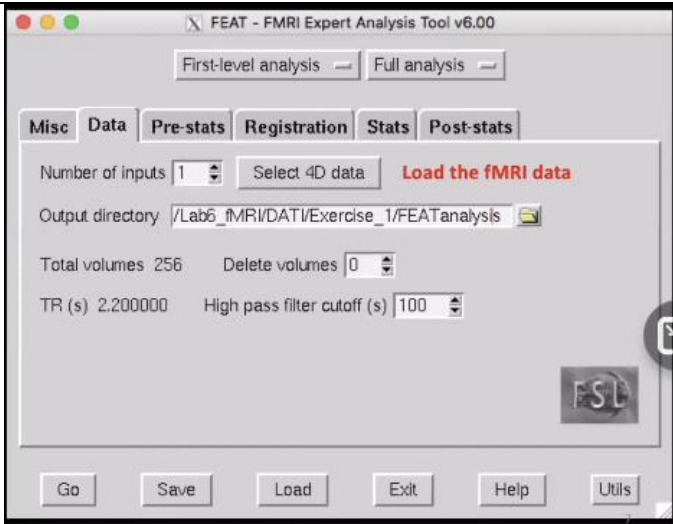
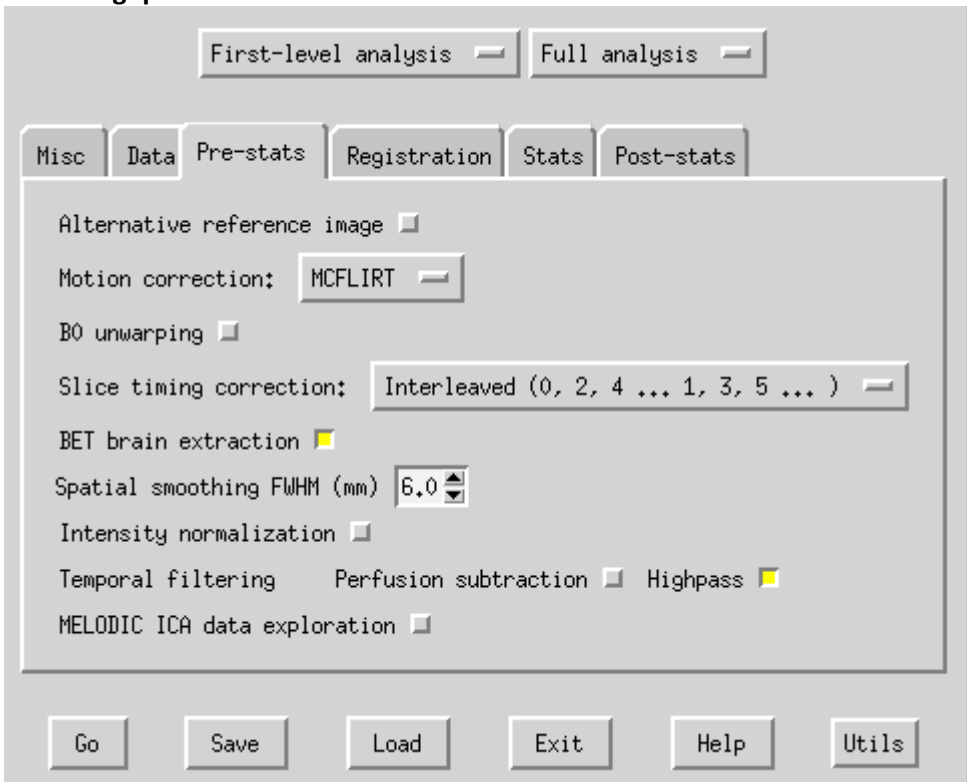
- The subject was instructed to lie still in the scanner, to keep the eyes closed but not to fall asleep while resting-state images were collected
- The protocol consisted of a **BLOCK DESIGN PARADIGM** alternating between a **RIGHT-HAND MOVEMENT** and the **IMAGERY OF A RIGHT-HAND MOVEMENT** (8 blocks each, 10 volumes for each block) and a rest condition (16 blocks, 6 volumes for each block)
- Protocol: **Protocol_mot1.txt**, **Protocol_ima2.txt** in the folder **"Exercise1"** (Moodle)

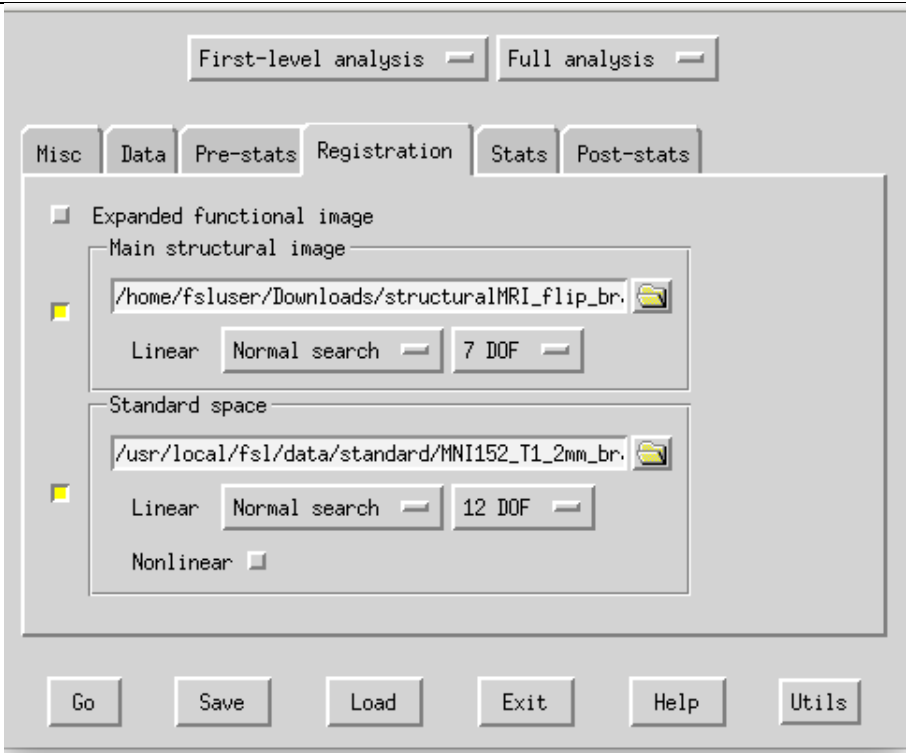


Strutture dati	
structuralMRI.nii.gz : risonanza magnetica di base; sequenza T1 (sequenza volumetrica strutturale). Dato in 3 dimensioni (volume). fMRI.nii.gz : volumi cerebrali durante la sequenza magnetica funzionale. Dato in 4 dimensioni (volume * 256 campioni nel tempo).	Acquisizione da soggetto sano volontario magnete 3 tesla. 256 volumi Ripetizioni da 2.2 secondi 36 fette Voxel da 3x3x3 mm ²

Visualizzare MRI e fMRI	<p><code>fsleyes <file></code></p> <p>Conviene aprire un fsleyes per ciascuna delle due.</p> <div data-bbox="429 1368 1466 1612" data-label="Image"> </div> <p>Posso visualizzare le due immagini sovrapposte mettendo entrambi i file dopo fsleyes.</p>
Riorientare l'immagine MRI per renderla omogenea con la fMRI	<p><code>fslswapdim <input> <a> <c> [output]</code></p> <p>Prende in input l'immagine e 3 valori a,b,c che indicano la rotazione e il nome del risultato. A,b,c sono i nuovi assi in termine dei vecchi assi; possiamo dare -x/x,-y/y,-z/z o anche RL LR AP PA SI IS (right left, anterior superior, superior inferior). Noi usiamo RL PA IS.</p> <div data-bbox="557 1935 1334 2078" data-label="Image"> </div>

<p>Estrarre dalle immagini MRI il contenuto dei tessuti cerebrali (Per esempio, rimuovere ossa e collo).</p>	<pre>bet structuralMRI_flip.nii.gz structuralMRI_flip_brain_t05 -f 0.5 -R -m -v</pre> <p>Bisogna fare molto attenzione perché potrebbe succedere di portare via pezzi di tessuto che ci interessano, in particolare dato che quello che ci interessa è proprio il limite esterno del cervello (la corteccia!)</p> <pre>Main bet2 options: -o generate brain surface outline overlaid onto original image -m generate binary brain mask -s generate approximate skull image -n don't generate segmented brain image output -f <f> fractional intensity threshold (0->1); default=0.5; smaller values give larger brain outline estimates -g <g> vertical gradient in fractional intensity threshold (-1->1); default=0; positive values give larger brain outline at bottom, smaller at top -r <r> head radius (mm not voxels); initial surface sphere is set to half of this -c <x y z> centre-of-gravity (voxels not mm) of initial mesh surface. -t apply thresholding to segmented brain image and mask -e generates brain surface as mesh in .vtk format</pre> <p>Variations on default bet2 functionality (mutually exclusive options):</p> <pre>(default) just run bet2 -R robust brain centre estimation (iterates BET several times) -S eye & optic nerve cleanup (can be useful in SIENA - disables -o option) -B bias field & neck cleanup (can be useful in SIENA)</pre> <p>Interfaccia:</p> 
<p>Analisi - Preparazione all'analisi</p>	<p>Scrivere il file da zero è molto laborioso, quindi è meglio eseguire l'interfaccia una prima volta e poi modificare il file.</p> 

	 <p>First level analysis = se selezionata, sto facendo analisi u un solo soggetto e non su un gruppo Full analysis: settiamo sia il preprocessing che l'analisi statistica</p> <p>Potrebbe essere necessario eliminare i primi volumi (potrebbero non essere ancora stabili).</p>
Analisi - Pre processing (pre-stats)	<ul style="list-style-type: none"> - Slice timing correction: bisogna dire come sono state acquisite; l'informazione ci viene fornita. - Estrazione BET vista prima - Smoothing spaziale: 6 mm (in generale deve essere il doppio della dimensione del voxel) - Filtro highpass 
Analisi – Registrazione	<ul style="list-style-type: none"> - Carichiamo l'immagine T1 nel campo main structural image (Già filtrata e flipata) - Possiamo scegliere uno spazio standard, ovvero a quale "media di cervelli" fare riferimento. - I gradi di libertà nella seconda devono essere alti (not sure why 31.27)

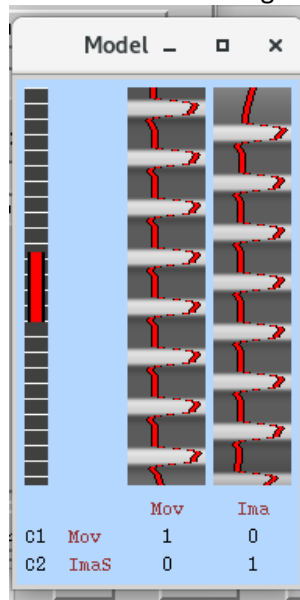
	
<p>Analisi – Costruire il GLM</p>	<div data-bbox="379 981 928 1048"> <p>on Stats Po → Full model setup</p> </div> <p>Bisogna inserire tutti i predittori, dettati dal protocollo (che trovi nel file .txt). Glielo passo direttamente e fa tutto lui! :D Il protocollo txt è sostanzialmente l'onda quadra, solo che dovremo dare ogni stato separatamente.</p> <div data-bbox="383 1276 912 1818"> <p>Movimento</p> <p>1 2</p> <p>EV name: Mov</p> <p>Basic shape: Custom (1 entry per volume)</p> <p>Filename: /home/fsluser/Downloads/Protocolo</p> <p>Convolution: Double-Gamma HRF</p> <p>Phase (s): 0</p> <p><input type="checkbox"/> Orthogonalise</p> <p><input type="checkbox"/> Add temporal derivative</p> <p><input checked="" type="checkbox"/> Apply temporal filtering</p> <p>View design Efficiency Done</p> </div> <div data-bbox="960 1276 1490 1818"> <p>Movimento immaginato</p> <p>1 2</p> <p>EV name: Ima</p> <p>Basic shape: Custom (1 entry per volume)</p> <p>Filename: /home/fsluser/Downloads/Protocolo</p> <p>Convolution: Double-Gamma HRF</p> <p>Phase (s): 0</p> <p><input type="checkbox"/> Orthogonalise</p> <p><input type="checkbox"/> Add temporal derivative</p> <p><input checked="" type="checkbox"/> Apply temporal filtering</p> <p>View design Efficiency Done</p> </div> <p>Creati questi due regressori posso applicare il contrasto; posso crearne due tipi (uno per stato).</p>

Contrasts F-tests

Paste Title EV1 EV2

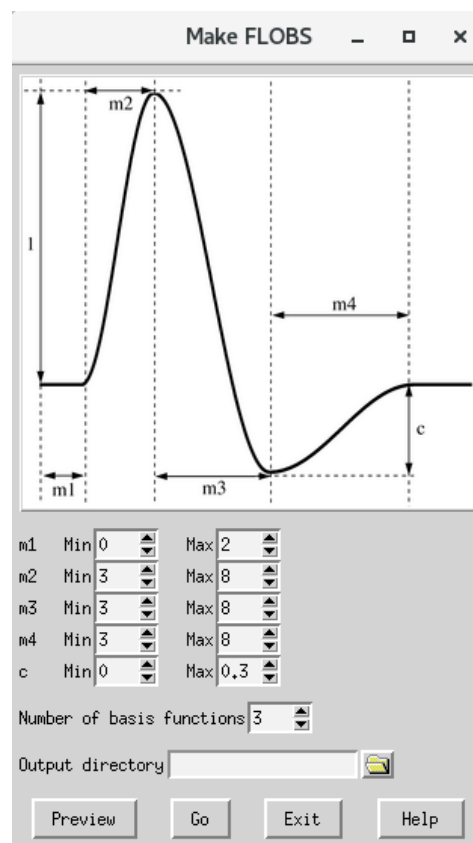
OC1	<input type="checkbox"/>	Mov	<input type="text" value="1"/>	<input type="text" value="0"/>
OC2	<input type="checkbox"/>	ImaS	<input type="text" value="0"/>	<input type="text" value="1.0"/>

Risultato bellissimissimo: essa è la descrizione dalla design matrix!



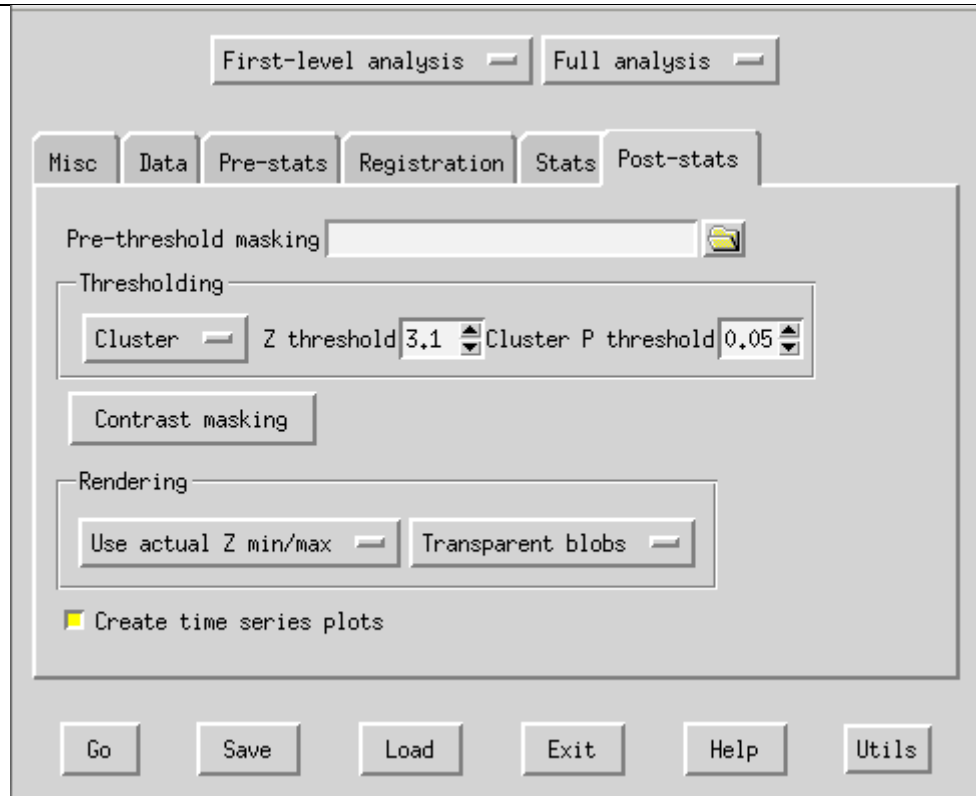
Visualizzare la HRF (Hemodynamic response function)

Make_flobs (sulla bash)



Analisi - Applicazione

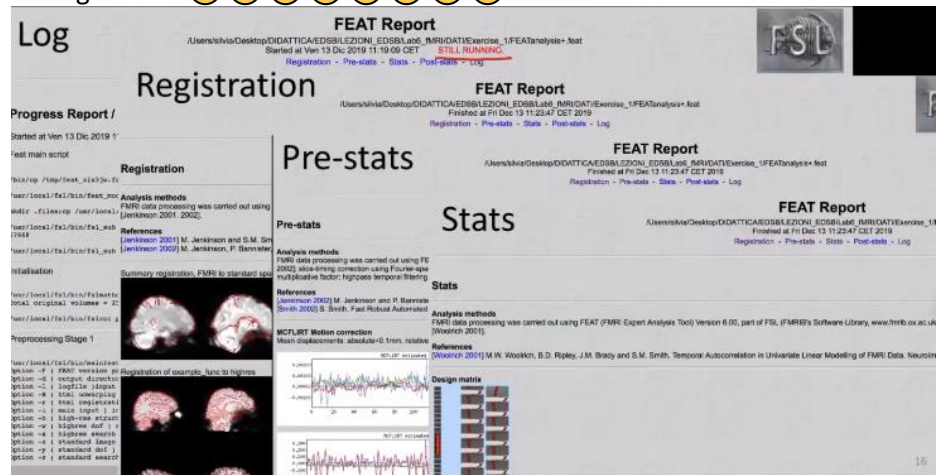
della soglia statistica



Avvia l'analisi

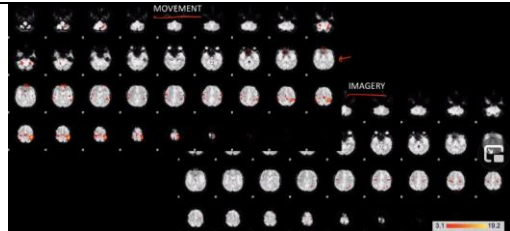


Si crea una fighissima pagina HTML che si aggiorna mano mano facendoti vedere cosa sta facendo e il log e tutto 🤖🤖🤖🤖🤖🤖🤖🤖🤖🤖



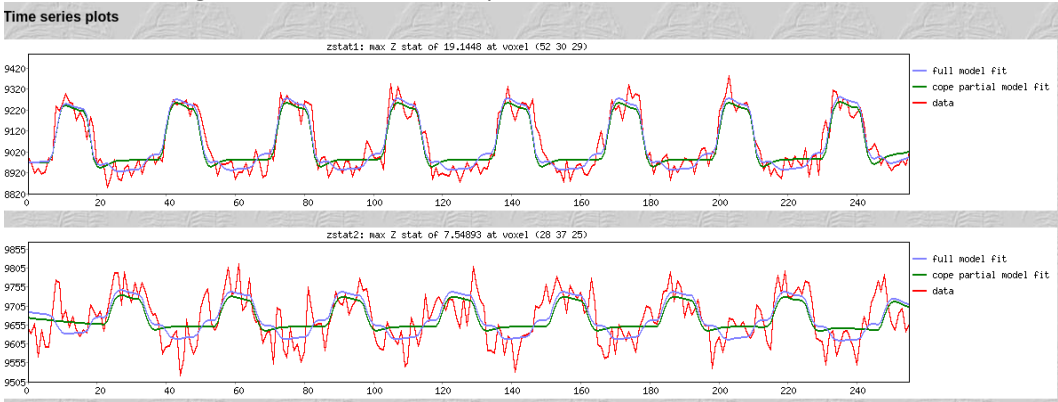
Ci mette una vita.

Report –
Mappe
statistiche

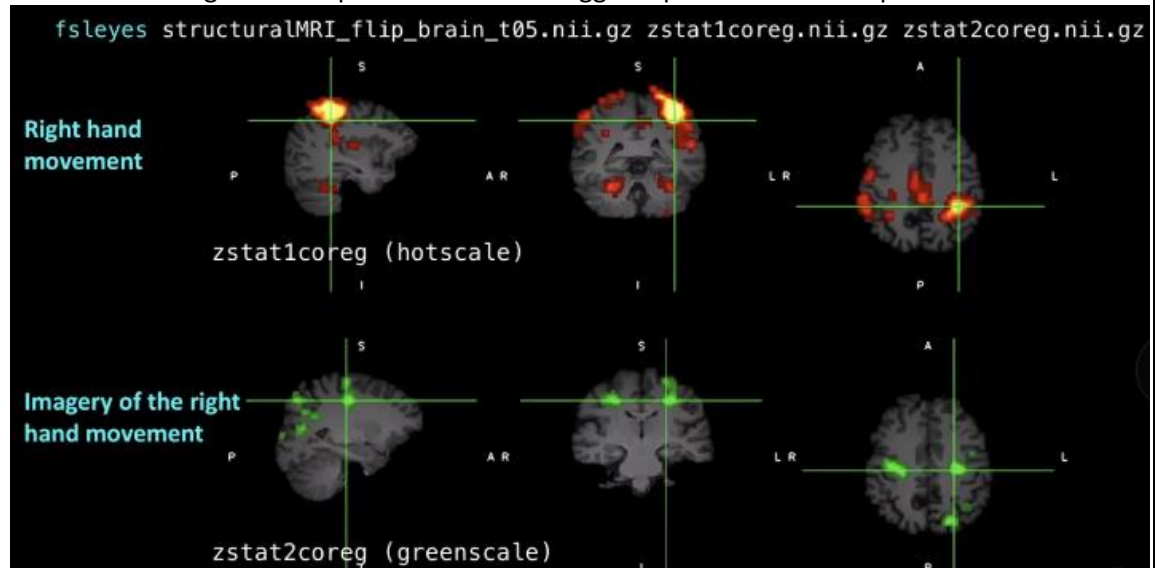


Viene proposto il volume cerebrale affettato.

Movimento: come prevedibile si attiva molto bene l'attivazione sul lato opposto del movimento

	Imagery: abbiamo un'attivazione ma mooolto più piccina.	
Report – time series	<p>Crea vari grafici, fra cui</p> <ul style="list-style-type: none"> - Per il voxel con valore statistico più alto - Per il segnale mediato, il cluster più intenso <p>Time series plots</p>  <p>Il dato in rosso è il dato grezzo, e sovrapposto in blu vediamo il full model fit (ovvero il modello creato dalla design matrix) moltiplicato per i valori trovate per quel voxel o per cluster di voxel.</p>	
File design.fsf	<pre># FEAT version number set fmri(version) 6.00 # Are we in MELODIC? set fmri(inmelodic) 0 # Analysis level # 1 : First-level analysis # 2 : Higher-level analysis set fmri(level) 1 # Which stages to run # 0 : No first-level analysis (registration and/or group stats only) # 7 : Full first-level analysis # 1 : Pre-processing # 2 : Statistics set fmri(analysis) 7 # Use relative filenames set fmri(relative_yn) 0 # Balloon help set fmri(help_yn) 1 # Run Featwatcher set fmri(featchat_yn) 1 # Cleanup first-level standard-space images set fmri(sscleanup_yn) 0 # Output directory</pre>	<p>feat design.fsf</p> <p>Con tutte queste belle operazioni abbiamo creato un file che si chiama design.fsf. Questo tipo di file può essere aperto e modificato per riadattare protocollo e immagine al nuovo dato.</p> <p>Lanciando feat design.fsf avviamo direttamente la funzione feat dal file, senza passare dalla interfaccia grafica.</p>
Registrazione delle mappe di attivazione sulla MRI strutturale	<p>flirt [options] -in <inputvol> -ref <refvol> -applyxfm -init <matrix> -out <outputvol></p> <p>I parametri necessari si troveranno in .feat, dove in particolare avremo le due mappe (zstat1 e zstat2). Uso la funzione flirt con:</p> <ul style="list-style-type: none"> - -in (input) : output delle operazioni precedenti (tresh_zstat1.gz) -in .feat/thresh_zstat1.nii.gz - -ref (target): la MRI strutturale, che viene automaticamente copiata in reg/highres; -ref .feat/reg/highres.nii.gz - -applyxfm -init: prende i parametri (generati in automatico dalle operazioni precedenti). .feat/reg/example_func2highres.mat - -out: il nome da dare in output. -out zstat1coreg 	

Questo bordello è necessario per spostare/adattare le mappe statistiche alle mappe strutturali; in particolare, al paziente vengono fatte prima quelle strutturali e poi quelle funzionali. Ciò significa che potenzialmente il soggetto potrebbe essersi spostato.



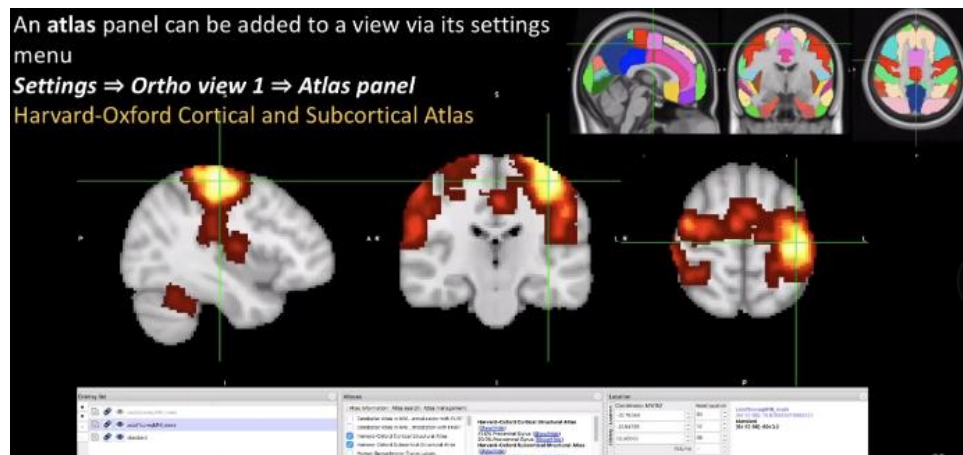
Possiamo impostare dal visualizzatore delle colormap diverse.

Registrare le mappe statistiche allo spazio standard (MNI?)

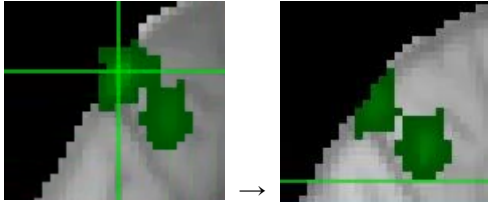
Si vogliono unire le mappe generate a una MRI *standardizzata*. Fom la stessa cosa di prima ma:

Si prende l'immagine BOLD (non quella grezza ma quella già unita a una MRI strutturale

- -in (input) : in questo caso vuole la versione **già registrata alla fMRI strutturale singola**
-in zstat1coreg.nii.gz
- -ref (target): vorremo la MRI standard,
-ref .feat/reg/standard.nii.gz
- -applyxfm -init: qui vorremo passare dallo spazio highres del soggetto a quello standard.. quindi.
.feat/reg/highres2standard.mat
- -out: il nome da dare in output.
-out zstat1coregMNI



L'anatomia è template! Possiamo aggiungere un pannello atlante per vedere subito a quale zona funzionale corrispondono le zone attivate

<p>Creare e applicare una maschera binaria.</p>	<p>fslmaths</p> <p>È utile in quanto può aiutarci a eliminare quelle attivazioni che andrebbero al di fuori della mappa standard; questo potrebbe accadere se lo smoothing è stato eccessivo.</p>  <p>Innanzitutto creiamo la maschera:</p> <pre>fslmaths .feat/reg/standard.nii.gz -bin -thr 0 standard_mask</pre> <p>Poi applichiamo la maschera alla mappa di attivazione con l'opzione -mas:</p> <pre>fslmaths zstat1coregMNI.nii.gz -mas standard_mask.nii.gz zstat1cor</pre>
<p>Cluster</p>	<p>Quantifica le attivazioni cerebrali</p> <div data-bbox="379 801 1490 1361"> <pre>>cluster Usage: cluster --in=<filename> --thresh=<value> [options] echo Clustering... cluster \ -i zstat1coregMNI.nii.gz \ -t 3.1 \ -o cluster_index1 \ --osize=cluster_size1 > cluster_info1.txt cluster \ -i zstat2coregMNI.nii.gz \ -t 3.1 \ -o cluster_index2 \ --osize=cluster_size2 > cluster_info2.txt echo done</pre> <div data-bbox="884 824 1477 875"> <p>Cluster takes the input image and thresholds it at 3.1</p> </div> <div data-bbox="1007 913 1490 1352"> <p>Outputs (images)</p> <p>Cluster Index (-o): this is an image where every cluster is assigned a unique number (from 1 to N) and the intensity of every voxel in a cluster is set to this unique number. e.g. every voxel in cluster number 3 will have an intensity of 3</p> <p>Size Index (--osize): every voxel in a cluster has its intensity set equal to the size of that cluster (where size is the number of voxels). This is useful for thresholding clusters based on size.</p> </div> </div>

Note da Eleonora:

- Importante non avere spazi o caratteri speciali nel percorso; esplode.