Questo codice implementa un filtro Artifact Component Specific Rejection (ACSR) per i dati EEG/MEG. Ecco una descrizione di alto livello di ciò che fa ogni funzione:

1. ACSR\_filter: Questa è la funzione principale che prende in input il segnale grezzo da utilizzare per addestrare i parametri del filtro (v\_rest), il segnale grezzo da filtrare (v\_target) e la lunghezza della finestra (Ws). Restituisce il segnale filtrato (v\_filtered).

function [v\_filtered] = ACSR\_filter(v\_rest,v\_target,Ws)

1. ACSR\_init: Questa funzione inizializza il filtro ACSR dividendo il segnale in finestre e calcolando le magnitudini massime per ogni finestra. Queste magnitudini massime vengono poi utilizzate per identificare gli artefatti nel segnale.

[mmags,~] = ACSR\_init(v\_rest,Ws);

overlap=round(Ws/2);

ev=Window\_division(v\_target,Ws,overlap);

fv=[];

1. Window\_division: Questa funzione divide il segnale in finestre di una certa lunghezza, con un certo sovrapposizionamento tra le finestre. Questo è utile per analizzare il segnale in segmenti più piccoli.

function [ output ] = Window\_division( data,windows,overlap)

[m,n]=size(data);

if min([m,n])==1

data=reshape(data,1,[]);

end

N\_ch=size(data,1);

output=cell(N\_ch,1);

End=[windows:(windows-overlap):length(data)];

for fi=1:N\_ch

sig=data(fi,:);

Sig=[];

for fj=1:length(End)

Sig=[Sig sig(:,End(fj)-windows+1:End(fj))'];

end

output{fi}=Sig;

end

if N\_ch==1

output=output{1}; end end

1. Artifact\_removal: Questa funzione rimuove gli artefatti dal segnale. Se lo stato è ‘init’, calcola le magnitudini delle componenti del segnale. Se lo stato è ‘ind\_app’, rimuove le componenti del segnale che hanno una magnitudine superiore a un certo valore (fornito come parametro).

function [varargout] = Artifact\_removal(input,statev,params)

switch statev

case 'init' % input= windows by trials

target\_fft=zeros(size(input));

for ii=1:size(input,2)

target=input(:,ii);

target\_fft(:,ii)=fft(target);

end

mags=abs(target\_fft);

if size(mags,2)==1

mmags=mags;

else

mmags=max(mags')';

end

varargout{1}=mmags;

varargout{2}=mags;

case 'ind\_app' % windows by 1

mmags=params;

target=input;

freq=fft(target);

comp\_freq=zeros(size(freq));

for ii=1:length(freq)

[a,b]=ACSR\_computation(freq(ii),mmags(ii));

comp\_freq(ii)=a+b\*1i;

end

comp=(ifft(comp\_freq,'symmetric'));

varargout{1}=comp;

end

end

1. ACSR\_computation: Questa funzione calcola le nuove componenti del segnale che sostituiranno le componenti artefatto. Questo viene fatto calcolando la magnitudine e la fase delle componenti originali e poi riducendo la magnitudine se supera un certo valore.

function [a1,b1] = ACSR\_computation(in,mag)

a0=real(in);

b0=imag(in);

s\_a=sign(a0);

p0=atan2(b0,a0);

m0=sqrt(a0.^2+b0.^2);

ratio=tan(p0);

m1=m0-mag;

if m1<0

m1=0;

end

a1=s\_a\*sqrt((m1.^2)/(1+ratio.^2));

b1=a1\*ratio;

end

In sintesi, questo codice implementa un filtro ACSR dividendo il segnale in finestre, identificando gli artefatti in ogni finestra, e poi rimuovendo o riducendo le componenti del segnale che corrispondono a questi artefatti. Il risultato è un segnale pulito in cui gli artefatti sono stati ridotti o eliminati.

Ricorda, questo è solo un esempio generico. La tua implementazione specifica potrebbe richiedere un approccio diverso a seconda della natura dei tuoi dati e degli artefatti.