Tesi di laurea triennale

Sviluppo e utilizzo, in linguaggio Python, della libreria BioSPPy e di funzioni create in integrazione, per l'acquisizione e l'elaborazione del segnale ECG

Sapienza Universita' di Roma

Corso di Laurea in Ingegneria Clinica

Relatore: Lorenzo Piazzo

Autore: Umberto Serraino

Indice

1 Introduzione

Questa tesi ha come obiettivo quello di creare una simulazione in Python del segnale elettrocardiogramma (ECG) e di operazioni di elaborazione del segnale, facendo utilizzo soprattutto della libreria BioSPPy e di alcuni script creati, in integrazione. Queste funzioni vengono introdotte facendo una fork del pacchetto BioSPPy, tramite Github. Il segnale ECG che andremo ad elaborare ed analizzare viene prelevato da un database chiamato PhysioNet, che presenta diversi segnali acquisiti in vari soggetti sani e soggetti che presentano patologie. Nel nostro caso, abbiamo a che fare con 11 registrazioni di soggetti diversi in condizione di salute, le quali sono contenute nella sottocartella 'dati'. La registrazione denominata 'rec0' è un caso particolare in cui è presente anche il disturbo dell'alimentazione di rete a 50 Hz.

1.1 Prerequisiti

E' molto importante far presente che tutte queste funzioni elaborate come integrazione del pacchetto BioSSpy sono state sviluppate tramite Python 3.10.4, in un pc con un sistema operativo 'Windows 11'. Prima di poter utilizzare questa fork, si deve possedere anche alcune librerie, che sono state molto importanti per lo sviluppo di alcune funzioni: Numpy, Matplotlib e Biosppy.

1.2 Come installare queste librerie?

Ecco una guida rapida per installare le librerie:

1.Premi il tasto Windows + R per aprire la finestra di dialogo Esegui. Quindi digita "cmd" e premi Invio per aprire una finestra di comando.

2.Digita i seguenti comandi:

```
>>pip install numpy

>>pip install matplotlib

>>pip install biosppy
```

Qualora queste installazioni fallissero, bisogna controllare se il pip è stato aggiunto alla variabile path, e in caso contrario, aggiungerlo.

1.3 Come scaricare la fork della libreria BioSPPy?

Una volta che sono stati soddisfatti tutti i prerequisiti, si acceda al seguente link:

https://github.com/SerrainoUmberto/BioSPPy.git

Una volta aperto il link, bisogna premere su 'Code' e successivamente su ' Download ZIP'. A questo punto partirà il download di una cartella denominata 'Tesi_us'. All'interno di essa, troviamo altre sottocartelle, quali 'Dati','Ecg' e 'Wrk'. Affinchè il programma funzioni correttamente, la cartella Tesi_us e tutto il suo contenuto, deve essere copiato sul desktop del computer. Successivamente vedremo degli esempi applicativi di come far partire queste funzioni.

2 Che cos'è BioSPPy ?

BioSPPy si tratta di una libreria scritta in Python ed è molto utile per l'elaborazione dei segnali biomedici. Esso presenta delle funzioni che permettono, dato segnali biomedici sottoforma di array in ingresso, di estrarre diverse caratteristiche di essi.

Ad esempio:

• Esistono funzioni permettono di ottenere un array del segnale ECG filtrato, facendo passare l'array in ingresso in un opportuno filtro con frequenze di taglio già predefinite, si può pure ottenere un array della posizione dei picchi dell'onda R del complesso QRS, la frequenza cardiaca; e si può decidere se ottenere solo questi valori in array oppure anche plottarli.

• Esistono funzioni che riguardano il segnale EEG che permettono sempre di ottenere un segnale filtrato sempre facendolo passari in filtri predefiniti, ma anche di ottenere la potenza media del segnale nelle varie bande del segnale stesso.

• Esistono funzioni riguardanti segnali EDA (attività elettrodermica) ed estrarre caratteristiche come i picchi delle risposte delle conducibilità cutanea.

• Esistono anche funzioni per l'elebarazione anche di segnali riguardanti la respirazione e segnale elettromiografico.

Inoltre possiamo trovare diversi tool per analizzare il segnale nel dominio del tempo e nel dominio della frequenza.

Ad esempio possiamo calcolare la potenza media in una banda di frequenza, fare la derivata del segnale in ingresso, decidere noi la banda dove fare l'opportuno filtraggio (passa-basso, passa-alto, passa-banda, filtro Notch) , che tipo di filtro utilizzare (Finite Impulse Response filter, Butterworth filter, Chebyshev filters, Bessel filter), ottenere i coefficienti al numeratore e al denominatore della funzione che definisce il filtro, calcolare lo spettro di potenza del segnale unilatero e, in caso, calcolarlo pure con il periodogramma di Welch; la media, la devianza standard, etc.

Inoltre presenta altri tipi di funzioni non inerenti alla elaborazione dei segnali biomedici, come delle funzioni per operare sui file, per plottare, di lavorare sulle directory, tecniche di clustering per la selezione di elementi omogenei in un insieme di dati, e altri tipi di utility. Tuttavia, per il nostro scopo non ci sono di alcun aiuto, e quindi non le definiamo.

2.1 Panoramica generale delle funzioni BioSSPy applicabili al segnale ECG

Nella libreria sono presenti diverse funzioni per l'elaborazione del segnale ECG:

```
>>biosppy.signals.ecg.christov_segmenter(signal, sampling_rate)
```

Si tratta di algoritmo definito da Christov, con lo scopo di trovare gli indici di collocazione del picco R del complesso QRS. I parametri in ingresso sono:

• signal – Campioni del segnale ECG;

• sampling rate – Frequenza di campionamento (Hz), che è opzionale ed in caso è preimpostata a 1000 Hz.

Il parametro in uscita è:

• rpeaks – Campioni che indicano l'indice in cui si trova il picco R del complesso QRS.

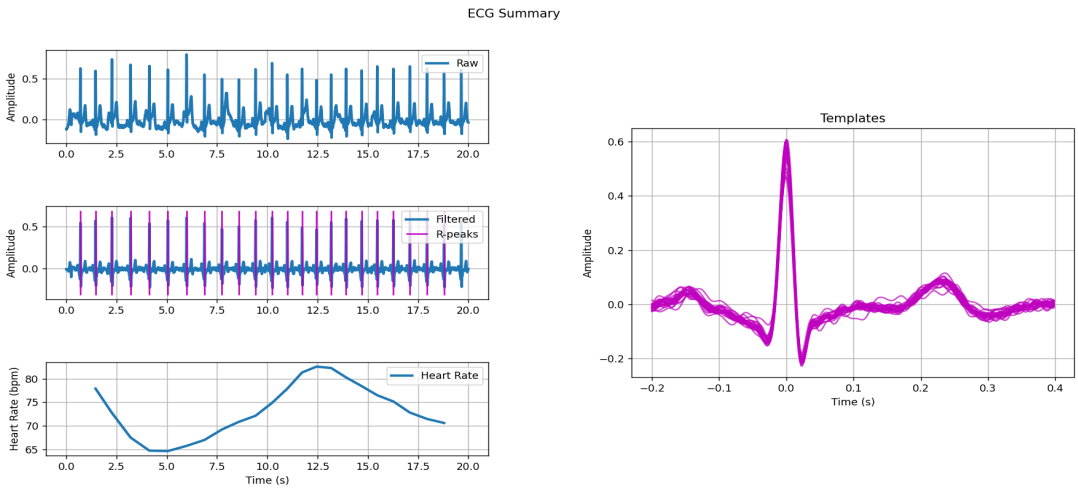

Figura 1: Applicazione della funzione sottostante su un vettore di 10000 campioni.

```
>>biosppy.signals.ecg.ecg(signal, sampling_rate, show=True)
```

Si tratta di una funzione che elabora un segnale ECG ed estrae le caratteristiche del segnale rilevanti utilizzando i parametri predefiniti. I parametri che riceve in ingresso sono:

• signal – Campioni del segnale ECG;

• sampling rate – Frequenza di campionamento (Hz), che è opzionale ed in caso è preimpostata a 1000 Hz;

• show – Se è impostato su True, viene mostrato un grafico di riferimento.

I parametri in uscita sono:

• ts – Campioni che fanno riferimento all'asse del tempo del segnale ECG

• filtered – Campioni del segnale ECG filtrato (i parametri del tipo di filtro sono già preimpostati nella funzione)

• rpeaks – Indici di posizione del picco R.

• templates ts – Campioni che fanno riferimento all'asse temporale dell'array templates (secondi).

• templates – Campioni che fanno riferimento a un modello del segnale durante un battito cardiaco.

• heart rate ts – Campioni che fanno riferimento all'asse del tempo della frequenza cardiaca (secondi).

• heart rate – Campioni della frequenza cardiaca istantanea (bpm).

```
>>biosppy.signals.tools.filter_signal(signal, ftype, band,
order, frequency, sampling_rate)
```

Effettua un filtraggio del segnale in base ai parametri definiti. I parametri in ingresso sono:

• signal – Campioni del segnale da filtrare.

• ftype – Tipo di filtro: Filtro a risposta all'impulso finita ("FIR"); Filtro Butterworth ("butter"); Filtri Chebyshev ("cheby1", "cheby2"); Filtro Bessel ("bessel"); Filtro ellittico ('ellip');

• band – Filtro passabasso('lowpass'); Filtro passa-alto('highpass'); Filtro passa-banda('bandpass'); Filtro Notch ('bandstop').

• order - Ordine del filtro.

• frequency – Frequenza di taglio: nel filtro passa-basso e passa alto, dobbiamo mettere solo una frequenza; mentre nei filtri passa-banda e filtro Notch, dobbiamo mettere una coppia di frequenze.

• sampling rate – Frequenza di campionamento (Hz), ed è opzionale in quanto, in caso, è preimpostata.

I parametri in uscita sono:

• signal – Campioni del segnale filtrato

• sampling rate – Frequenza di campionamento (Hz)

• params – Parametri del filtro.

```
>>biosppy.signals.tools.welch_spectrum(signal, sampling_rate, size,
overlap, window='hanning', pad, decibel=True)
```

Calcola lo spettro di potenza di un segnale (unilaterale). I parametri in ingresso sono:

• signal – Campioni del segnale in ingresso;

• sampling rate – Frequenza di campionamento (Hz), ed è opzionale in quanto, in caso, è preimpostata.

• decibel – Se è impostata su True, ritorna la potenza del segnale in dB.

• size - Numero di campioni in ogni segmento di Welch; è opzionale in quanto preleveva l'equivalente di 1 secondo di segnale.

• overlap- Numero di campioni che si sovrappongono al segmento precedente; è opzionale, in quanto è impostato di default alla metà.

• window - Tipo di finestra da utilizzare.

• pad- Aggiungere degli zeri ai campioni in maniera tali che si raggiunge una potenza di 2; è opzionale.

I parametri in uscita sono:

• freqs – Array di frequenze, a cui corrisponde i campioni dello spettro calcolato.

• power – Array dello spettro di potenza.

3 Funzioni create come integrazione del pacchetto BioSPPy

3.1 Load_Ecg

La funzione *load_ecg* permette di caricare uno dei file presente nella sottocartella 'dati', dando come input il numero del file scelto. Di tutti i campioni presenti nel file (10000), ne preleva solo un quantitativo pari a quanto definito in input alla funzione. Questo array di campioni del segnale ECG scelto, viene salvato nella sottocartella 'Wrk' con il nome di *ecg.npy*. Vediamo un esempio di applicazione:

```
>>load_ecg(num_file,campioni)
```

Se mettiamo num file=2 e campioni=1024, la funzione preleverà dal file $rec2.txt$ presente nella sottocartella 'Dati', un numero di campioni pari a 1024 e li salverà nel file *ecg.npy* della sottocartella 'Wrk'

3.2 Plot_Ecg

La funzione *plot_ecg* permette di visualizzare un segnale ECG nel tempo e rappresentarlo negli assi cartesiani con le opportune unità di misura. Il parametro da dare in ingresso a questa funzione è il nome del file presente nella sottocartella 'Wrk' del segnale di cui si vuole fare il plot, sottoforma di stringa. Vediamo un esempio di applicazione:

```
>>plot_ecg(nome_file_da_plottare)
```

Immaginiamo di voler plottare il segnale salvato nel file *ecg.npy* nella sottocartella 'Wrk, che è stato precedentemente caricato dalla funzione *load_ecg*. Quello che si va

a fare è scrivere la funzione $plot\_ecg('ecg')$, cioè il nome del file che precede il *.npy*. Scrivendo questa funzione, viene quindi visualizzato l'andamento temporale del segnale ECG scelto.

3.3 Ecg_add_sin0

La funzione ecg_add_sin0 permette di sommare al segnale ECG scelto, una funzione sinusoidale di ampiezza e frequenza variabile. Questa funzione riceve in ingresso un vettore x che rappresenta il segnale ECG e l'ampiezza e la frequenza della sinusoide da aggiungere al segnale ECG. In uscita si ottiene un vettore z che rappresenta il segnale ECG disturbato dalla sinusoide.

$z = ecg\_add\_sin0(x, a, fo)$

3.4 Ecg_add_sin

La funzione *ecg_add_sin* e' un'interfaccia per la funzione ecg_add_sin0, che rappresenta il cuore della funzione. La funzione *ecg_add_sin* si occupa di caricare il file scelto, contenente il segnale ECG e, in particolare, quanti campioni utilizzare di questo segnale ECG. Successivamente somma al segnale ECG, una sinusoide di ampiezza a e di frequenza f0, e salva il segnale modificato in un altro file, denominato *ecg_seno.npy*, che si troverà nella sottocartella 'Wrk'. Vediamo come funzione questa funzione. Digitiamo il seguente codice:

```
>>ecg_add_sin(n_file,n_campioni,a,fo)
```

Quello che succede è che al segnale ECG contenuto nella sottocartella 'Dati', di numero n_file, di cui scegliamo $n\_campioni$, viene sommato un segnale sinusoidale di ampiezza a e frequenza f0. Il segnale ECG modificato viene salvato con il nome di *ecg_seno.npy*

3.5 Ecg_zero0

La funzione ecg_zero0 azzera tutti i valori del segnale ECG in un determinato intervallo temporale. I parametri di ingresso sono il vettore x, che rappresenta il segnale ECG da modifiare e gli estremi dell'intervallo temporale in cui il segnale verra' azzerato. In uscita la funzione restituisce il vettore y che rappresenta il segnale ECG azzerato nell'intervallo temporale [to,t1] espresso in secondi.

$y = ecg\_zero0(x, to, t1)$

3.6 Ecg_zero

La funzione *ecg_zero* riceve in ingresso il numero del file che contiene il segnale ECG da modificare, e gli intervalli di tempo in cui annullare il segnale ECG. Successivamente la funzione salva il segnale ECG modificato all'interno di un nuovo file nella sottocartella 'Wrk', con il nome di *ecg_zero.npy*. Questa funzione si basa sulla funzione *ecg_zero*0 precedentemente definita. Si consideri un esempio di uso dell'applicazione; digitando il seguente codice:

```
>>ecg_zero(nu_file,to,t1)
```

vengono prelevati tutti i campioni contenuti nel file, e viene azzerato il segnale ECG, contenuto nel file numero nu_file per un intervallo di tempo pari a [to,t1] secondi e poi viene salvato il nuovo segnale ECG nella sottocartella 'Wrk'.

3.7 Ecg_noise0

La funzione ecg noise aggiunge al segnale ECG un rumore bianco filtrato tra due frequenze e scalato di una quantita' che permette di ottenere un determinato rapporto segnale-rumore specificato tra i parametri della funzione, in ingresso . La funzione riceve in ingresso un vettore x che rappresenta il segnale ECG da modificare, l'intervallo di frequenze [fo,f1] attraverso il quale verra' filtrato un rumore bianco e il rapporto segnale-rumore espresso in [dB] che il segnale modificato in uscita dovra' avere.

$z = ecg\_noise0(x, f0, f1, SNRdB)$

3.8 Ecg_noise

La funzione *ecg_noise* e' l'interfaccia per la funzione *ecg_noise*0 che rappresenta il cuore della funzione. I parametri di ingresso di *ecg_noise* sono il nome del file che contiene il segnale ECG da modificare, quanti campioni prelevare del segnale ECG, l'intervallo rappresentato dalle frequenze [fo,f1] attraverso il quale verra' filtrato, con un filtro passa-banda, un rumore bianco e il rapporto segnale-rumore espresso in [dB] che il segnale in uscita dovra' avere. Digitando il seguente comando:

```
>>ecg_noise(n_file,n_campioni,f0,f1,SNRdB)
```

si produce il segnale somma del segnale ECG e del rumore bianco, e lo salva nella sottocartella 'Wrk' con il nome di *ecg_noise.npy*

3.9 Graf_spettro

La funzione *graf_spettro* richiede in ingresso il nome del file che si trova nella sottocartella 'Wrk' andare a prelevare, e quale finestra utilizzare per rappresentare lo spettro del segnale in esame. Se sigitiamo il seguente comando:

```
>>graf_spettro(nome_file_segnale,finestra)
```

e come $finestra$ mettiamo *boxcar* (ovvero la finestra rettangolare).

Mentre come $nome\_file\_segnale$ mettiamo ad esempio *ecg_noise*, questa funzione preleverà il vettore di campioni del segnale ECG presente nel file selezionato della sottocartella 'Wrk' (in questo caso il segnale ECG a cui è stato aggiunto rumore bianco) e andrà a graficare lo spettro del segnale in esame. In particolare, sull'asse delle ordinate abbiamo la densità spettrale di potenza mV^2/Hz espressa in dB, e sull'asse delle ascisse abbiamo la frequenza in Hz

3.10 Ecg_disturba

La funzione *ecg_disturba* riceve in ingresso il numero del file da cui prelevare il segnale ECG, e ha la probabilità del 50% di applicare ognuno dei tre disturbi (azzerare il segnale ECG in un certo intervallo temporale, aggiungere un rumore bianco filtrato con un filtro passa banda tra f0 e f1, e aggiungere un disturbo sinusoidale di ampiezza a e frequenza fo) e quindi di modificare il segnale ECG, e restituendo in uscita il file modificato sotto il nome di *ecg_distuba.npy*, contenuto nella sottocartella 'Wrk'. k è un parametro che cambia il seed: se k è maggiore di zero, permette di resettare il generatore di numeri random che determina quali disturbi vengono applicati, e quindi ogni qual volta diamo quel valore in ingresso, otteniamo sempre gli stessi disturbi. Se k=0, i valori vengono generati in maniera indipendenti tra loro, e quindi ogni volta che chiamiamo la funzione, i disturbi che vengono aggiunti sono in maniera casuale. Digitando il comando:

```
>>ecg_disturba(n_file, k)
```

se al n_file mettiamo ad esempio 1 e poniamo k=0, noi stiamo prelevando i campioni presenti nella rec1 della sottocartella 'dati' e abbiamo 50% di probabilità di aggiungere ognuno dei tre disturbi definiti precedentemente. Qualora i disturbi non venissero aggiunti oppure azzerasse il segnale in un certo intervallo di tempo e aggiungesse il la sinusoide a una frequenza f0 e il rumore bianco filtrato passa-banda in un intervallo [fo,f1], la funzione emetterà diverse stringhe in cui specifica questa aggiunta oppure no. Qualora al segnale ECG prelevato, venisse aggiunto uno di quei tre disturbi, esso verà salvato nella sottocartella 'Wrk'; diversamente invece non verrà fatto. Qualora noi definissimo un valore di k diverso da 0, ad esempio, k=1, noi al segnale ECG caricato, aggiungeremmo sempre i

disturbi che saranno sempre uguale ogni qual volta noi mettiamo in ingresso alla funzione k=1; stessa cosa succede per k=2, ma con valori diversi, e così via. Infatti definendo dei valori di k diversi da zero, andiamo a resettare il generatore di valori random.

4 Esempi applicativi delle funzioni

Abbiamo scaricato la cartella *Tesi_us* ed è stata posta nel desktop del pc. Per prima cosa dobbiamo aprire il nostro ambiente di lavoro. Premi il tasto Windows + R per aprire la finestra di dialogo Esegui. Quindi digita "cmd" e premi Invio per aprire una finestra di comando. A questo punto digitiamo il comando *python* per aprire il nostro linguaggio di programmazione Python. Essendo che le nostre funzioni si trovano nella sottocartella *Ecg* della cartella *Tesi_us*, dobbiamo impostare la nostra directory di lavoro quest'ultima. Infatti se noi digitiamo:

```
>>import os
```

E successivamente :

```
>>os.getcwd()
```

possiamo notare che la nostra directory di lavoro è diversa della sottocartella *Ecg*. Pertanto digitiamo il seguente comando, inserendo il percorso file della sottocartella 'Ecg':

```
>>os.chdir('percorso file sottocartella Ecg')
```

A questo punto dobbiamo prelevare le funzioni che vogliamo utilizzare. Pertanto digitiamo:

```
>>from tesi_us import *
```

4.1 Funzioni base

A questo punto siamo pronti per cominciare a lavorare. Abbiamo a disposizione 11 registrazioni di campioni del segnale ECG prelevati tutti con frequenza di campionamento Fs pari a 500 (la registrazione 0 è del segnale ECG che presenta disturbo dell'alimentazione di rete a 50 Hz). Decidiamo di prelevare la registriamo numero 4, e di prenderne solo 1024 campioni. Allora digitiamo:

```
>>load_ecg(4,1024)
```

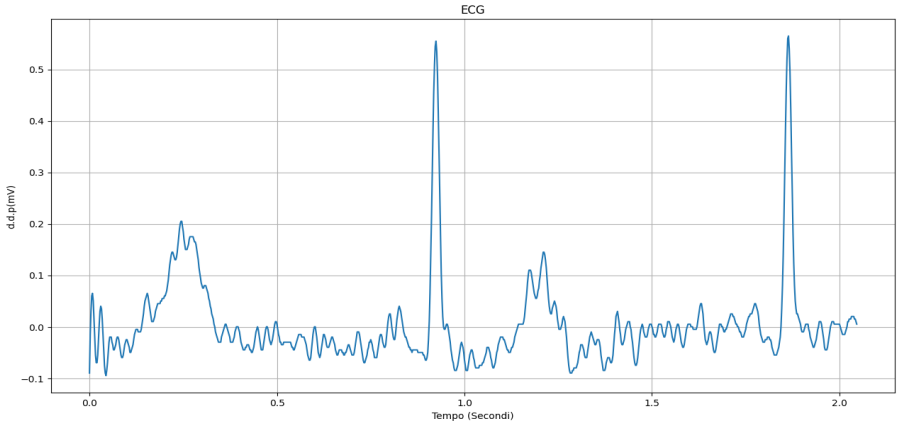

Figura 2: Visualizzazione della registrazione numero 4, di cui abbiamo prelevato 1024 campioni

A questo punto il vettore di campioni verrà salvato nel file *ecg.npy* nella sottocartella 'Wrk'. Pertanto possiamo decidere di visualizzarlo, andando a digitare:

```
>>plot_ecg('ecg')
```

Un'altra funzione che possiamo utilizzare è quella che ci permette di visualizzare lo spettro di densità di potenza del segnale salvato nel file *ecg.npy*. Andiamo a visualizzarlo andando ad utilizzare due tipi di finestrature diverse, in modo tale da cogliere le differenze. Nel primo caso, andiamo a finestrare il nostro segnale con la finestra di Bartlett.

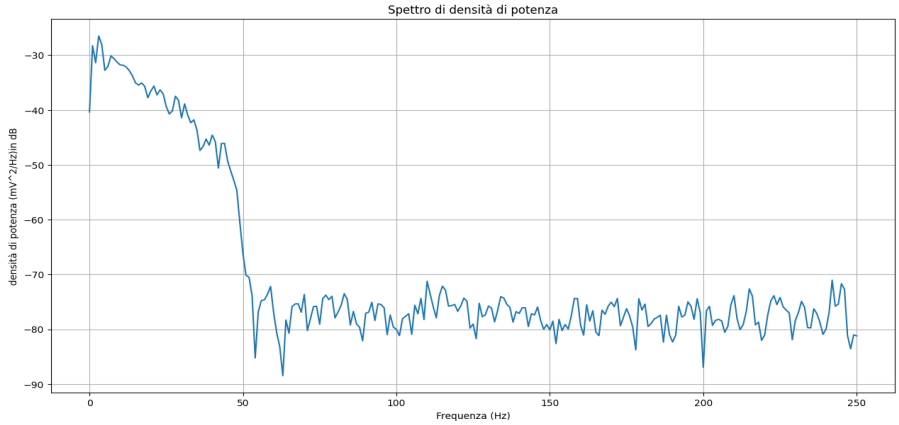

Figura 3: Visualizzazione dello spettro di densità di potenza utilizzando la finestra di Bartlett su 1024 campioni
Pertanto i codici che dobbiamo scrivere sono:

```
>>graf_spettro('ecg','bartlett')
```

Mentre nel secondo caso utilizziamo la finestra rettangolare, che presenta un lobo principale di ampiezza inferiore rispetto a quella triangolare, e quindi lo spettro ha una migliore risoluzione, ma come controindicazione i lobi secondari sono meno attenuati.

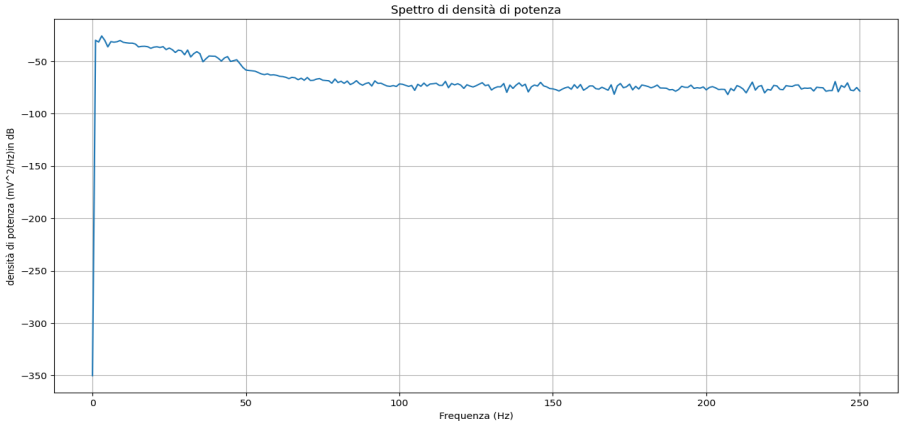

Figura 4: Visualizzazione dello spettro di densità di potenza utilizzando la finestra rettangolare su 1024 campioni

```
>>graf_spettro('ecg','boxcar')
```

4.2 Applicazione manuale dei disturbi

Attraverso l'uso delle funzioni presentate in questo paragrafo, si puo' perturbare il segnale generando e simulando disturbi al segnale ECG. Ad esempio, la funzione *ecg_add_sin* permette di sommare al segnale ECG scelto da una delle registrazioni a disposizione (scegliamo ad esempio la prima registrazione), una funzione sinusoidale di ampiezza 3 e frequenza 78. Infatti se noi facciamo la chiamata:

```
>>ecg_add_sin(1,2048,3,78)
```

quello che facciamo è salvare questo nuovo segnale ECG perturbato dal segnale sinusoidale con il nome di *ecg_seno* nella sottocartella 'Wrk'

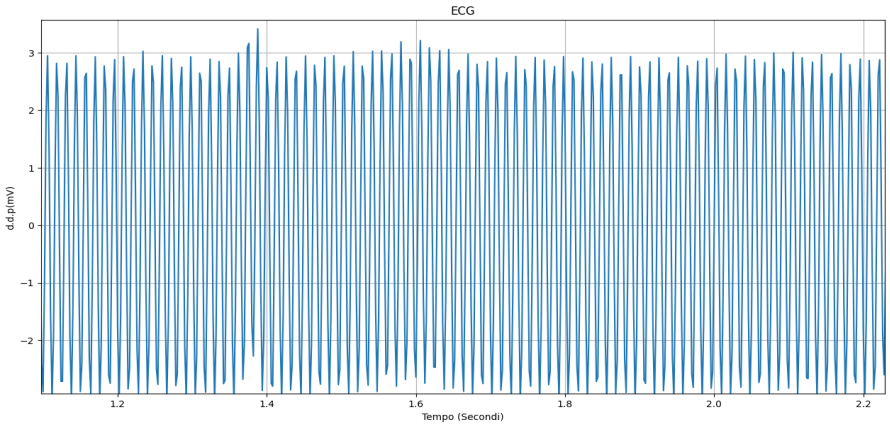

Figura 5: Visualizzazione nel tempo di 2048 campioni del segnale ECG della registrazione 1

Se proviamo a fare la visualizzazione di questi campioni nel tempo, quello che otteniamo è questo in figura 5. Quello che dobbiamo fare è chiamare la funzione *plot_ecg* ma come ingresso questa volta dobbiamo mettere la stringa *ecg_seno*

```
>>plot_ecg('ecg_seno')
```

Vediamo che il segnale ECG è completamente immerso nel rumore dato dalla sinusoide introdotta, tale da non poter più permettere di distinguere il vero segnale ECG. Possiamo utilizzare adesso la funzione $graf\_spettro$ che permette di visualizzare lo spettro proprio per verificare che è stata aggiunta questa sinusoide a frequenza a 78 Hz. Qualora fosse presente questo disturbo, noi possiamo apprezzare un picco proprio in corrispondenza della frequenza f0=78 Hz. Andiamo quindi a chiamare la funzione. Anche qui, questa

volta in ingresso alla funzione, devo specificare che sto prelevando il file *ecg_seno.npy*, e inoltre decido di utilizzare la finestra di Hann, che in genere è molto utilizzata per i segnali biomedici per le sue caratteristiche intermedie di risoluzione e leakage.

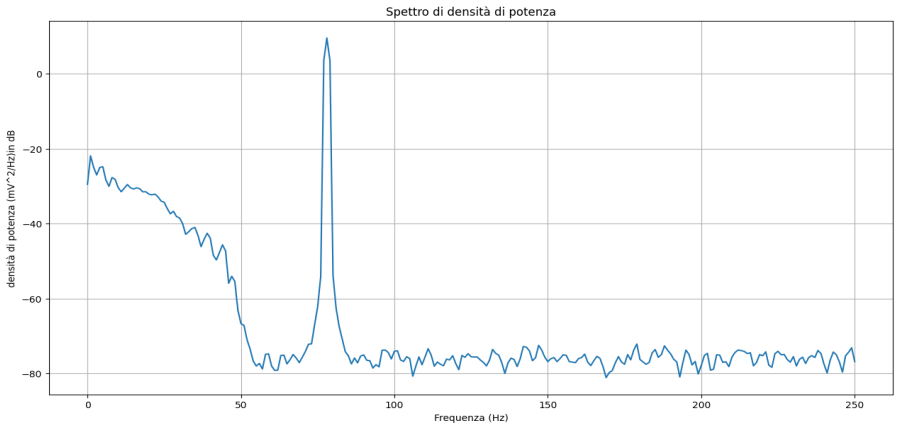

Figura 6: Visualizzazione dello spettro di densità di potenza utilizzando la finestra di Hann su 2048 campioni

```
>>graf_spettro('ecg_seno','hann')
```

Possiamo notare che è presente il picco a 78 Hz.

Un altro tipo di disturbo che possiamo aggiungere, si tratta quello di azzerare il segnale ECG in un certo intervallo di tempo prefissato. Infatti con:

```
>>ecg_zero(5,10,12)
```

noi stiamo prelevando i campioni della registrazione numero 5, e stiamo azzerando il segnale ECG nell'intervallo di tempo compreso tra 10 e 12 secondi. Questo file ECG modificato sarà salvati nella sottocartella 'Wrk' con il nome di *ecg_zero.npy*

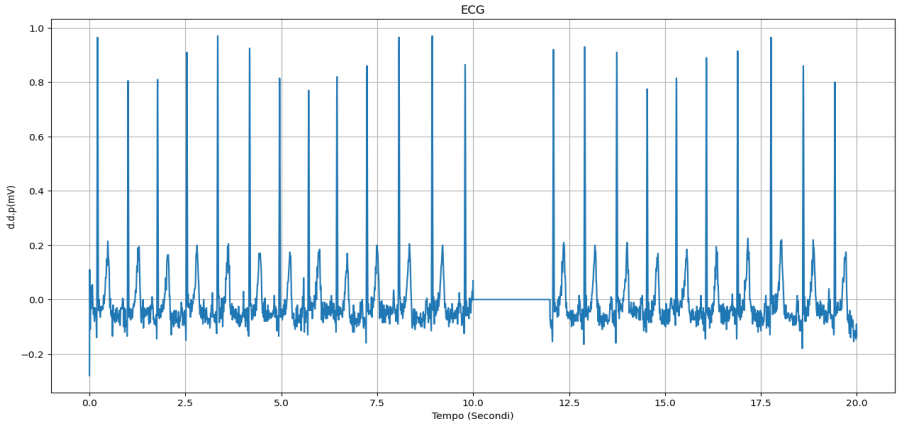

Figura 7: Visualizzazione del segnale ECG azzerato nell'intervallo di tempo [10,12]

Se noi andiamo a plottare il file appena salvato con la funzione:

```
>>plot_ecg('ecg_zero')
```

possiamo notare che otteniamo un andamento del segnale ECG nel tempo che si annulla proprio nei parametri temporali impostati.

Un altro tipo di disturbo che possiamo introdurre si tratta di rumore bianco. Noi sappiamo che il rumore bianco ha uno spettro che si può estendere fino a 10^{12} - 10^{13} Hz. Pertanto quello che andiamo a fare è filtrare il processo bianco con un filtro passa-banda [fo,f1]. Se noi digitiamo la funzione:

```
>>ecg_noise(2,2048,70, 150,-3)
```

stiamo dicendo che vogliamo prelevare 2048 campioni del segnale ECG presenti nella *rec*2.*txt* della sottocartella 'Dati', e gli aggiungiamo un processo bianco filtrato nelle frequenze comprese tra 70 e 150 Hz. Inoltre abbiamo imposto come parametro in ingresso alla funzione che il SNR in dB del segnale in uscita deve essere di -3dB, cioè il rumore deve essere prevalente rispetto al segnale ECG. Il segnale in uscita sarà salvato nella sottocartella 'Wrk' con il nome di *ecg_noise*.*npy*.

Possiamo verificare che la funziona ha svolto il proprio ruolo, andando a visualizzare l'andamento nel tempo del segnale ECG perturbato e il suo spettro di potenza, con una finestra a piacere.

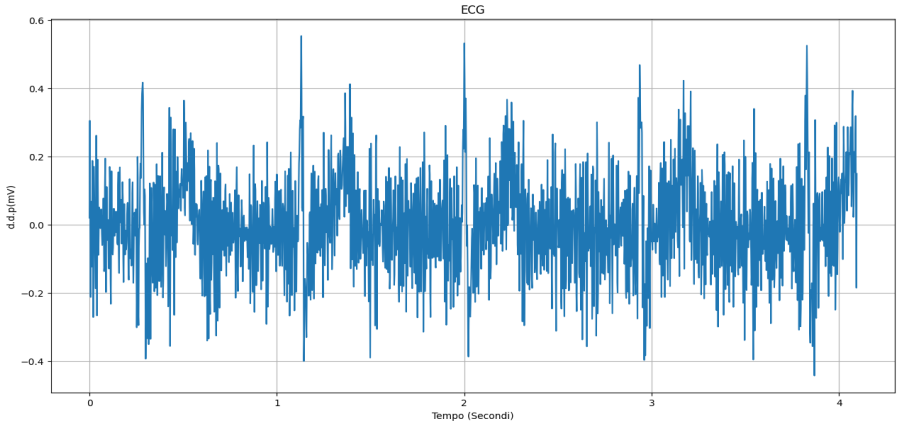

Figura 8: Visualizzazione di 2048 campioni del segnale ECG nel tempo a cui è stato aggiunto rumore bianco nelle frequenze [70,150] con SNR pari a -3 dB

```
>>plot_ecg('ecg_noise')
```

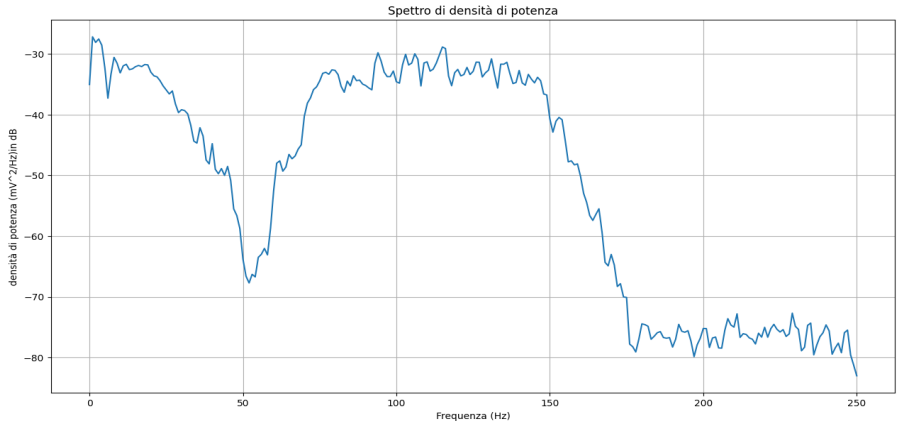

Figura 9: Visualizzazione dello spettro del segnale ECG di 2048 campioni, a cui è stato aggiunto rumore bianco nelle frequenze [70,150] con SNR pari a -3 dB, finestrato con Hann

```
>>graf_spettro('ecg_noise','hann')
```

Come verifica del fatto che possiamo modulare il rapporto segnale-rumore del segnale ECG disturbato in uscita, richiamiamo le stesse funzioni, solo che al posto di - 3 dB,

mettiamo 20 dB. Ci aspettiamo che il segnale ECG abbia un'ampiezza maggiore rispetto al rumore bianco introdotto.

```
>>ecg_noise(2,2048,70,150,20)
```

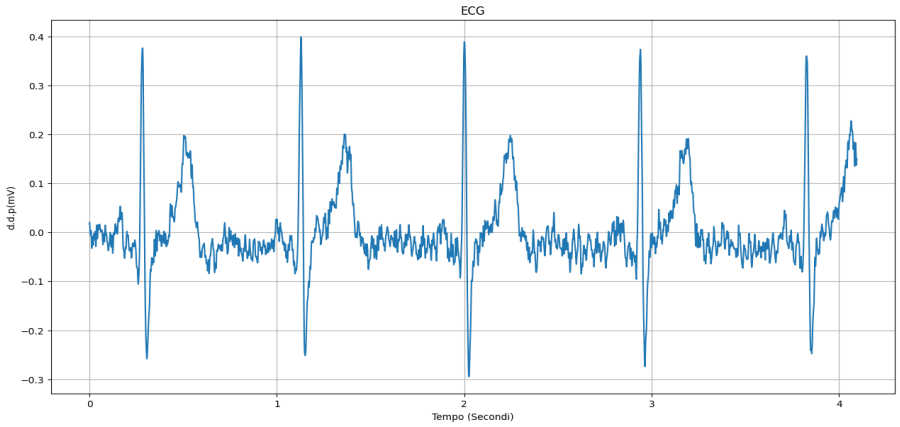

Figura 10: Visualizzazione di 2048 campioni del segnale ECG nel tempo a cui è stato aggiunto rumore bianco nelle frequenze [70,150] con SNR pari a 20 dB

```
>>plot_ecg('ecg_noise')
```

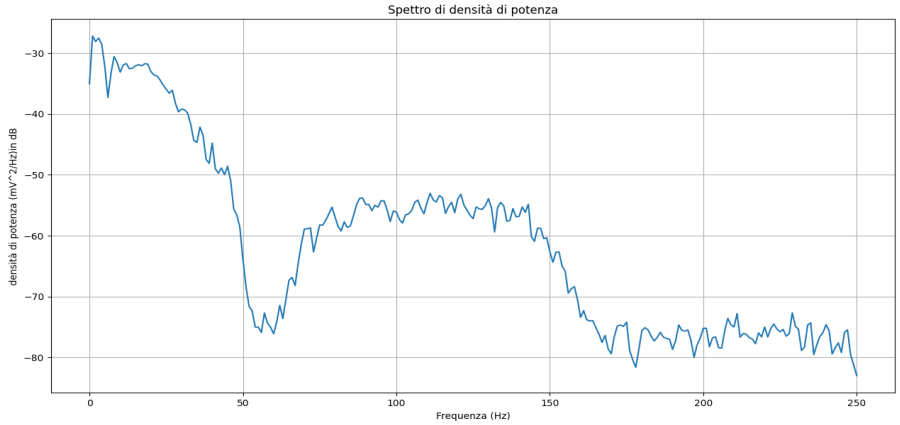

Figura 11: Visualizzazione dello spettro del segnale ECG di 2048 campioni, a cui è stato aggiunto rumore bianco nelle frequenze [70,150] con SNR pari a 20 dB, finestrato con Hann

```
>>graf_spettro('ecg_noise','hann')
```

4.3 Filtraggio

La libreria BioSPPy mette a disposizione una funzione che permette di filtrare un segnale in ingresso. Si tratta della funzione *biosppy.signals.tools.filter_signal*, la quale effettua un filtraggio del segnale in base ai parametri definiti in ingresso: dobbiamo definire in ingresso il segnale da filtrare, il tipo di filtro da utilizzare, che tipo di operazione vogliamo fare al segnale (far passare alte frequenze/basse frequenze, etc), l'ordine del filtro, la frequenza di taglio e la frequenza di campionamento del segnale in ingresso. Tra tutti i tipi di filtri, preferiamo utilizzare il filtro Butterworth, in quanto è quello che presenta il roll-off con pendenza più alta, senza che vi sia ripple nella banda passante (che noi non vogliamo). Proprio per sfruttare questa funzione, abbiamo a disposizione la registrazione *rec*0.*txt*, che presenta i campioni del segnale ECG, a cui è aggiunto il disturbo dall'alimentazione di rete a 50 Hz. Andiamo a verificare che il segnale sia effettivamente disturbato dalla frequenza di rete, andando a visualizzare il suo andamento nel tempo e il suo spettro. Le funzioni che dobbiamo richiamare sono le seguenti. Innanzitutto, dobbiamo prelevare i campioni della registrazione *rec*0.*txt* presente nella sottocartella 'Dati', utilizzando la funzione *load_ecg*. A questo punto i campioni prelevati vengono salvati nel file *ecg.npy* della sottocartella 'Wrk'.

```
>>load_ecg(0,2048)
```

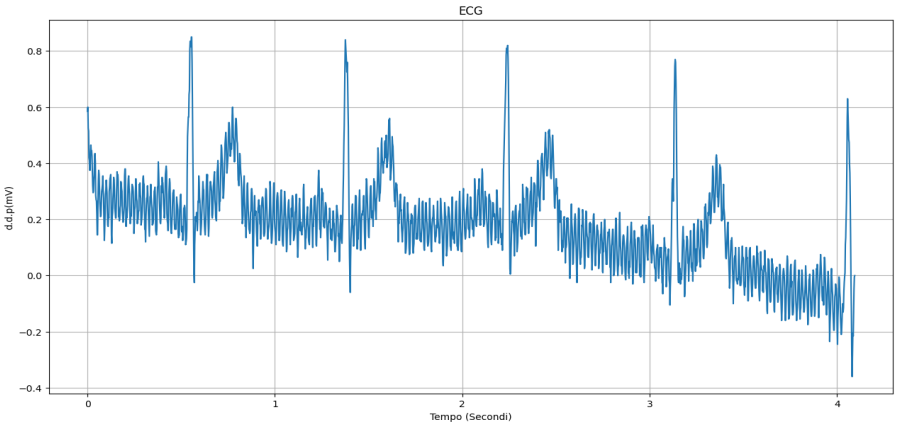

Figura 12: Visualizzazione di 2048 campioni del segnale ECG nel tempo a cui è stato aggiunto disturbo di rete a 50 Hz

A questo punto andiamo a stampare i file che sono stati salvati nell'ultimo file tramite la funzione *plot_ecg*

```
>>plot_ecg('ecg')
```

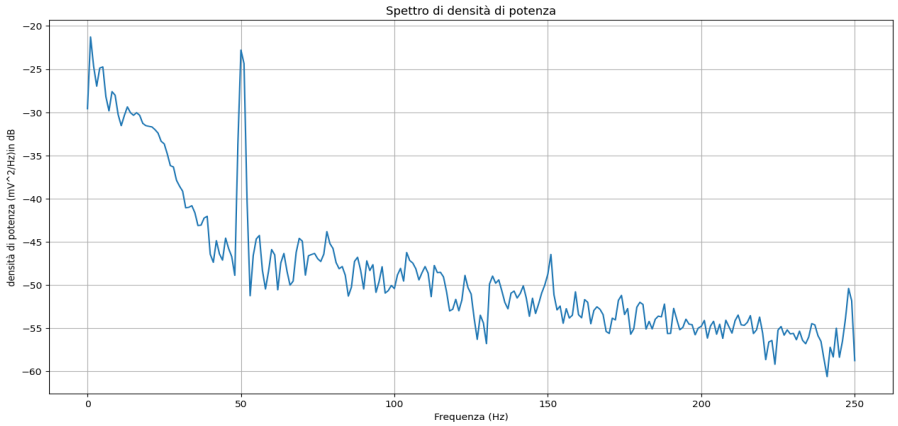

Figura 13: Visualizzazione dello spettro del segnale ECG di 2048 campioni, a cui è presente il disturbo di rete a 50 Hz, finestrato con Hann

Possiamo notare che lo spettro di potenza presenta effettivamente un picco a 50 Hz.

```
>>graf_spettro('ecg','hann')
```

Andiamo quindi ad utilizzare la funzione $biosppy.signals.tools.filter\_signal$. Prima di ciò, siccome dobbiamo mettere in ingresso i campioni del segnale ECG perturbato, dobbiamo introdurli nel nostro ambiente di lavoro. Per fare questo, dobbiamo utilizzare la funzione *np.load*, e la assegnamo a una variabile generica x. Questa funzione riceve in ingresso un certo percorso file. Siccome noi vogliamo prelevare i campioni salvati nella sottocartella 'Wrk' con il nome di *ecg.npy*, dobbiamo introdurre in ingresso la directory di questo file.

```
>>x= np.load('percorso file ecg.npy')
```

Avendo definito la variabile che presenta in ingresso i campioni del segnale da filtrare, dobbiamo decidere gli altri parametri per il nostro filtro. Abbiamo detto che utilizziamo il filtro Butterworth, e in particolare un filtro Notch. Siccome vogliamo eliminare la frequenza a 50 Hz, in ingresso alla funzione mettiamo le frequenze [49,51] Hz. L'ordine, cioè la complessità del filtro, lo impostiamo a 4, e come frequenza di campionamento abbiamo 500 Hz. La funzione $biosppy.signals.tools.filter\_signal$, in uscita, ci da: i campioni del

segnale filtrato, la frequenza di campionamento e parametri del filtro. Di tutti questi parametri, in questo ambito, ci interessano solo i campioni del segnale filtrato, e decidiamo di assegnarli a una certa variabile.

```
>>y,k,l=biosppy.signals.tools.filter_signal(signal=x, ftype='butter',order=4,
band='bandstop',frequency=[49,51], sampling_rate=500)
```

Per comodità decidiamo di salvare questi campioni filtrati nella sottocartella 'Wrk' nel file *ecg.npy*. Per fare questo utilizziamo la funzione *np.save*. Questa funzione permette di salvare l'array nel file definito, e per fare questo, dobbiamo inserito la directory del file e anche la variabile che vogliamo salvare:

```
>>np.save('percorso file ecg.npy',y)
```

A questo punto stampiamo il segnale nel tempo e lo spettro, per vedere come sono cambiati rispetto al caso precedente al filtraggio.

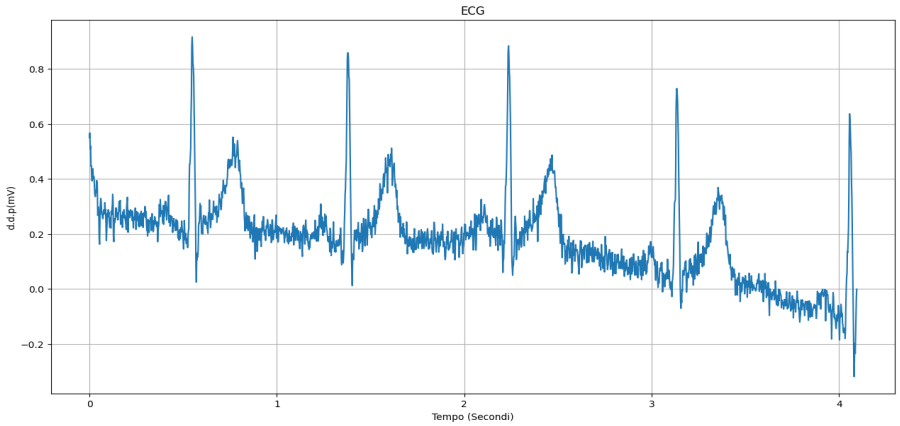

Figura 14: Visualizzazione di 2048 campioni del segnale ECG nel tempo, post filtraggio

```
>>plot_ecg('ecg')
```

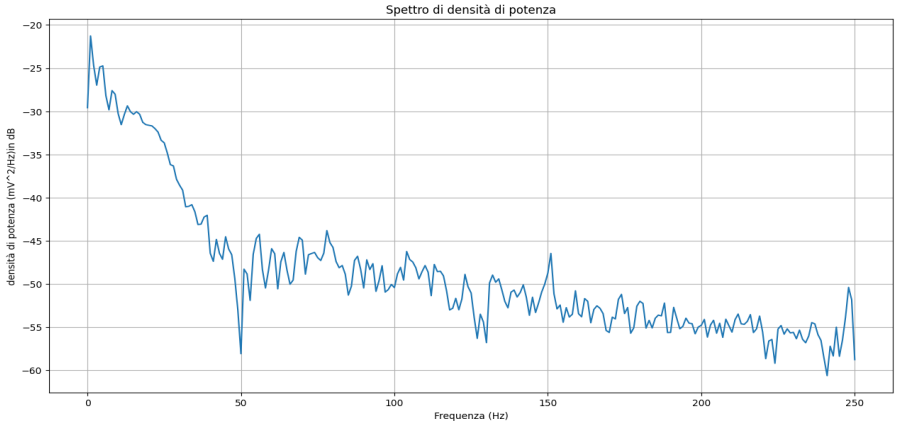

Figura 15: Visualizzazione dello spettro di densità di potenza di 2048 campioni del segnale ECG, post filtraggio

```
>>graf_spettro('ecg','hann')
```

Mentre nel tempo, potrebbe essere difficile l'interpretazione della presenza o meno del disturbo a 50 Hz, nello spettro, possiamo notare che il picco non è più presente.

4.4 Applicazione randomica dei disturbi

La funzione *ecg_disturba* applica uno o piu' disturbi casuali al segnale ECG. Prima di utilizzare questa funzione, verifichiamo che il segnale che noi andiamo a prelevare non sia affetto dai disturbi che la funzione ammette in maniera randomica al segnale ECG selezionato. Andiamo a visualizzare l'andamento nel tempo e lo spettro del segnale ECG presente nel file *rec*8.*txt* della sottocartella 'Dati'. Pertanto andiamo a richiamare le funzioni che permettono di selezionare il file in esame, i campioni e ci permettono di plottare lo spettro e l'andamento nel tempo:

```
>>load_ecg(8,10000)
```

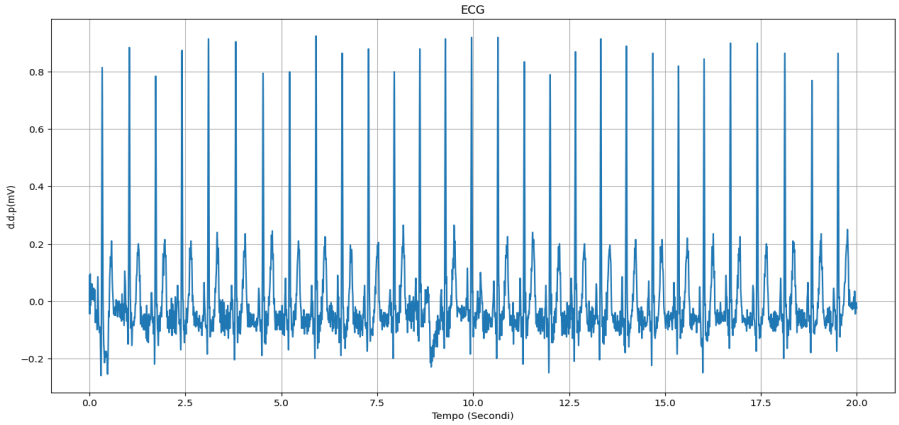

Figura 16: Visualizzazione del segnale ECG nel tempo

```
>>plot_ecg('ecg')
```

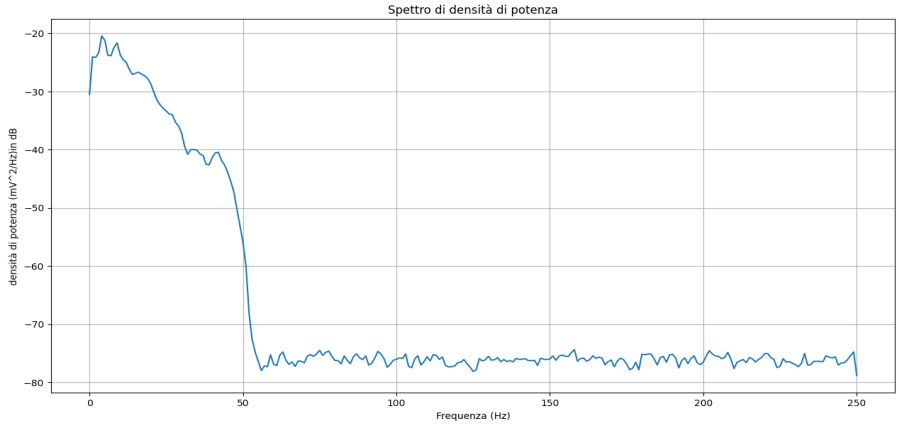

Figura 17: Visualizzazione dello spettro di densità di potenza del segnale ECG

```
>>graf_spettro('ecg','hann')
```

A questo punto applichiamo la funzione *ecg_disturba*. Essa genera e somma al segnale ECG presente nel file scelto tramite il primo parametro, n_file, dei disturbi, che hanno la stessa probabilità di essere applicati. Successivamente salva il segnale disturbato nel file *ecg_disturba.npy*, all'interno della sottocartella 'Wrk'. Il secondo parametro k permette di resettare il generatore di numeri random, consentendo, in caso, di applicare una sequenza

di disturbi precisa ogni volta, per k diverso da zero. Per conoscere i disturbi ed i relativi parametri generati, inoltre la funzione stampa, ogni volta che viene chiamata, i tipi di disturbi che ha inserito. Quindi se noi digitiamo:

```
>>ecg_disturba(8,0)
```

stiamo applicando i disturbi alla registrazione numero 8 del segnale ECG, e stiamo aggiungendo dei disturbi in maniera casuale, essendo k=0. Quello che viene visualizzato dalla funzione, nel mio caso, è:

La funzione ha azzerato il segnale ECG tra to=7 e t1=15.

La funzione ha aggiunto un disturbo sinusoidale di ampiezza a=0.039301946925442005 e frequenza f=60 Hz al segnale ECG.

La funzione ha aggiunto un rumore passa banda tra la frequenza fo=123 Hz e la frequenza f1=247 Hz al segnale ECG. Il rapporto segnale-rumore in dB vale 18.

A questo punto andiamo a visualizzare l'andamento nel tempo del segnale e il suo spettro, per vedere se i disturbi aggiunti sono veritieri. Quello che dobbiamo fare è utilizzare le funzioni *ecg_plot* e *graf_spettro*, solo che questa volta il file che dobbiamo prelevare dalla sottocartella 'Wrk' è *ecg_disturba.npy*

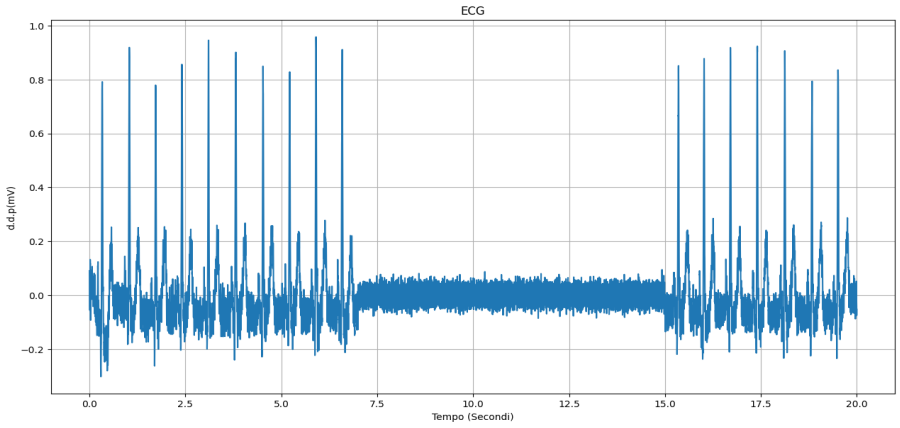

Figura 18: Visualizzazione nel tempo del segnale ECG perturbato da rumore random

```
>>plot_ecg('ecg_disturba')
```

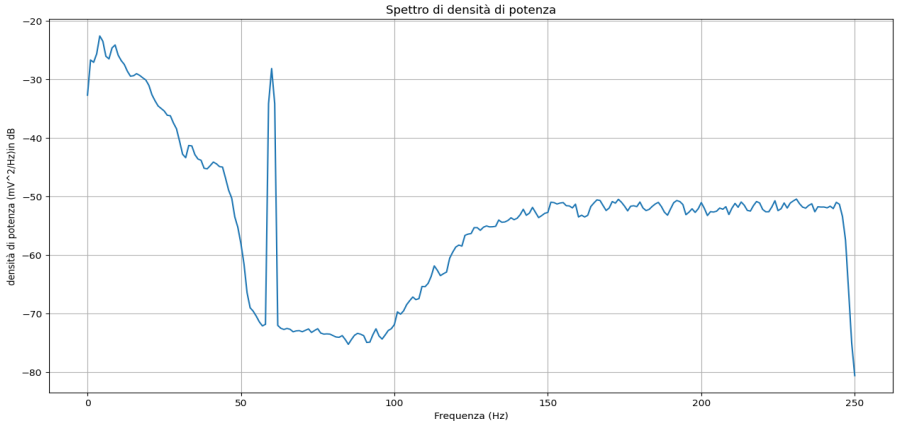

Figura 19: Visualizzazione dello spettro di densità di potenza del segnale ECG perturbato da rumore random

```
>>graf_spettro('ecg_disturba,'hann')
```

Possiamo notare che dalla visualizzazione di questi due grafici, e confrontandoli con i precedenti, tutto torna.