Caratterizzazione di un segnale ECG di un paziente con fibrillazione atriale

1 Introduzione

In questa esperienza di laboratorio studiamo la capacità di DFT, STFT e CWT di estrarre la **frequenza dominante** (frequenza dei picchi) da segnali ECG relativi a un paziente con fibrillazione atriale.

I dati sono in un file `cardio100.mat` che contiene tre tracce: `heart100`, `heart101`, `heart102`, ciascuna di 2048 campioni, campionate a

$$f_s = 1000 \text{ Hz} \quad (\Delta t = 1 \text{ ms}).$$

La sequenza di attività è:

1. analisi FFT diretta (DFT) delle tre tracce;
2. analisi STFT (spettrogramma);
3. analisi CWT (wavelet continua) con analisi scala–frequenza.

Tutte le attività vanno svolte in **Python** dentro un Jupyter Notebook, completando i frammenti di codice contrassegnati come `# TODO`.

2 Preparazione ambiente e caricamento dati

Passo 0: installazione librerie (fuori dal notebook)

```
pip install numpy matplotlib scipy pywavelets
```

Passo 1: prima cella del notebook

```
import numpy as np
import matplotlib.pyplot as plt

from scipy.io import loadmat
from scipy.signal import spectrogram, get_window
import pywt

# Per grafici interattivi 2D/3D (opzionale):
```

```
# %matplotlib notebook
%matplotlib inline

fs = 1000.0  # Hz (frequenza di campionamento)
```

Passo 2: caricamento di `cardio100.mat`

```
# TODO: controlla che cardio100.mat sia nella stessa cartella del
    notebook
data = loadmat("cardio100.mat")

# TODO: estrai heart100, heart101, heart102 come array 1D
heart100 = np.asarray(data["heart100"]).squeeze()
heart101 = np.asarray(data["heart101"]).squeeze()  # TODO: verifica il
    nome
heart102 = np.asarray(data["heart102"]).squeeze()  # TODO: verifica il
    nome

N  = len(heart100)
t  = np.arange(N) / fs

print("Numero di campioni N =", N)
```

Passo 3: normalizzazione e visualizzazione nel tempo

```
def normalizza(x):
    """
    TODO: normalizza x in modo che max(|x|) = 1.
    Suggerimento: usa np.max(np.abs(x)).
    """
    # return x / ...
    return ...

x  = normalizza(heart100)
x1 = normalizza(heart101)
x2 = normalizza(heart102)

plt.figure(figsize=(10,4))
plt.plot(t, x,  label="heart100")
plt.plot(t, x1, label="heart101")
plt.plot(t, x2, label="heart102")
plt.xlabel("Tempo [s]")
plt.ylabel("Ampiezza normalizzata")
plt.title("Tracce ECG (normalizzate)")
plt.grid(True)
plt.legend()
plt.tight_layout()
```

```
plt.show()
```

Richiesta: per tutte e tre le serie `heart100`, `heart101`, `heart102`:

- determinare **il periodo medio dei picchi** (spikes) nel tempo;
- a partire dal periodo medio, determinare per ognuna il valore di k dove ci si aspetta il picco nella FFT da **1000 punti** e da **2048 punti**.

Suggerimento: il periodo medio T (in secondi) si può stimare contando il numero medio di campioni tra picchi e dividendo per f_s; la frequenza dominante è $f_0 = 1/T$. Per una DFT di lunghezza N_{FFT}:

$$k \approx \frac{f_0}{f_s} \cdot N_{\text{FFT}}.$$

Esercizio: annotare i valori stimati di T, f_0 e k (per 1000 e 2048) per ciascuna traccia.

3 1. Analisi FFT diretta: `heart100`

Si richiede:

- FFT a 1000 punti, senza e con finestra di Hamming;
- FFT a 2048 punti, senza e con finestra di Hamming;
- elevare al quadrato il segnale e ripetere le due FFT;
- riassumere brevemente i risultati.

Passo 4: funzione FFT semplificata (solo frequenze positive)

```
def fft_pos(x, fs, Nfft=None, use_hamming=False):
    """
    Calcola la FFT di x con lunghezza Nfft (se None usa len(x)).
    Restituisce (f_pos, |X(f)|) per f >= 0.
    """
    x = np.asarray(x)

    if Nfft is None:
        Nfft = len(x)

    # TODO: seleziona i campioni da usare (ad es. primi Nfft)
    x_seg = x[:Nfft]

    if use_hamming:
        # TODO: applica finestra di Hamming: w = np.hamming(len(x_seg))
        w = ...
        x_seg = x_seg * w

    X_full = np.fft.fft(x_seg, n=Nfft)
```

```
    f_full = np.fft.fftfreq(Nfft, d=1.0/fs)

    # Considera solo le frequenze positive
    mask   = (f_full >= 0)
    f_pos = f_full[mask]
    X_pos = np.abs(X_full[mask])

    return f_pos, X_pos
```

Passo 5: 1000 punti FFT, con/senza Hamming

```
Nfft = 1000

# (1.a) senza finestra
f_1000_nowin, X_1000_nowin = fft_pos(x, fs, Nfft=Nfft, use_hamming=False
    )

plt.figure(figsize=(8,4))
plt.plot(f_1000_nowin, X_1000_nowin)
plt.xlim(0, 100)
plt.xlabel("Frequenza [Hz]")
plt.ylabel("|X(f)|")
plt.title("heart100 - FFT 1000 pt (senza finestra)")
plt.grid(True)
plt.tight_layout()
plt.show()

# (1.b) con finestra di Hamming
# TODO: imposta use_hamming=True e rifai il grafico
f_1000_ham, X_1000_ham = fft_pos(x, fs, Nfft=Nfft, use_hamming=True)

plt.figure(figsize=(8,4))
plt.plot(f_1000_ham, X_1000_ham)
plt.xlim(0, 100)
plt.xlabel("Frequenza [Hz]")
plt.ylabel("|X(f)|")
plt.title("heart100 - FFT 1000 pt (Hamming)")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Passo 6: 2048 punti FFT, con/senza Hamming

```
Nfft = 2048

# TODO: ripeti come al passo 5 ma con Nfft = 2048
```

```
f_2048_nowin, X_2048_nowin = fft_pos(x, fs, Nfft=Nfft, use_hamming=False)
f_2048_ham,   X_2048_ham   = fft_pos(x, fs, Nfft=Nfft, use_hamming=True)

# TODO: scegli se plottare in due figure separate o nella stessa (con legenda)
```

Passo 7: segnale al quadrato (1000 e 2048 punti)

```
# Segnale al quadrato
x_sq = x**2

# 1000 pt FFT su x_sq (con/senza Hamming)
f_1000_sq_nowin, X_1000_sq_nowin = fft_pos(x_sq, fs, Nfft=1000, use_hamming=False)
f_1000_sq_ham,   X_1000_sq_ham   = fft_pos(x_sq, fs, Nfft=1000, use_hamming=True)

# 2048 pt FFT su x_sq (con/senza Hamming)
f_2048_sq_nowin, X_2048_sq_nowin = fft_pos(x_sq, fs, Nfft=2048, use_hamming=False)
f_2048_sq_ham,   X_2048_sq_ham   = fft_pos(x_sq, fs, Nfft=2048, use_hamming=True)

# TODO: plottare e confrontare gli spettri ottenuti.
```

Passo 8: commento finale su heart100

Richiesta: scrivere un breve paragrafo (a parole) in cui si riassume:

- se il picco atteso si vede chiaramente;
- differenze tra 1000 e 2048 punti;
- effetto della finestra di Hamming;
- effetto dell'elevazione al quadrato del segnale.

4 2. Analisi FFT di `heart101` e `heart102`

Si richiede di ripetere la stessa analisi eseguita per `heart100` anche sulle tracce `heart101` e `heart102`.

Passo 9: ripetizione per `heart101`, `heart102`

```
# TODO: normalizza heart101 e heart102 se non lo hai gia' fatto
x1 = normalizza(heart101)
x2 = normalizza(heart102)

# TODO: ripeti gli stessi passi 5-7 per x1 e x2
# (1000 e 2048 punti, con/senza Hamming, segnale al quadrato)
```

Esercizio: confronta visivamente gli spettri di `heart100`, `heart101`, `heart102` e commenta:

- le frequenze dominanti sono simili o diverse?
- la definizione del picco è migliore per qualche traccia?

5 3. Analisi tramite STFT

Si vuole ora eseguire un'analisi simile usando la STFT (Short-Time Fourier Transform), scegliendo opportunamente le finestre, e trarre alcune considerazioni.

Passo 10: funzione STFT semplificata

```
def stft_spettrogramma(x, fs, window_len, overlap_frac):
    """
    Calcola STFT tramite scipy.signal.spectrogram.
    Restituisce (t_stft, f_stft, Sxx).
    """
    # TODO: crea una finestra di Hamming di lunghezza window_len
    win = get_window("hamming", window_len)

    # TODO: calcola il numero di campioni di overlap
    noverlap = int(overlap_frac * window_len)

    f_stft, t_stft, Sxx = spectrogram(
        x,
        fs=fs,
        window=win,
        nperseg=window_len,
        noverlap=noverlap,
        mode="complex"
    )
    return t_stft, f_stft, Sxx
```

Passo 11: spettrogramma 2D (modulo lineare)

```
# Esempio: STFT su heart100
window_len   = 128      # TODO: prova 64, 256, ...
overlap_frac = 0.5      # 50% di sovrapposizione
```

```
4
5 t_stft, f_stft, Sxx = stft_spettrogramma(x, fs, window_len, overlap_frac
      )
6
7 plt.figure(figsize=(8,4))
8 plt.pcolormesh(t_stft, f_stft, np.abs(Sxx), shading="gouraud")
9 plt.colorbar(label="|S(t,f)|")
10 plt.xlabel("Tempo [s]")
11 plt.ylabel("Frequenza [Hz]")
12 plt.ylim(0, 100)
13 plt.title(f"Spettrogramma STFT (heart100, win={window_len})")
14 plt.tight_layout()
15 plt.show()
```

Esercizi:

- ripetere l'analisi STFT per `heart101`, `heart102`;
- provare diverse lunghezze di finestra (ad es. 64, 128, 256) e discutere il compromesso tra risoluzione temporale e frequenziale.

6 4. Analisi tramite CWT (wavelet continua)

Si vuole ora usare la CWT (Continuous Wavelet Transform) per lo stesso obiettivo, passando per:

- analisi scala–frequenza;
- test su sinusoide a 50 Hz;
- applicazione all'ECG `heart100`.

Passo 12: relazione scala–frequenza in PyWavelets

In PyWavelets, per una wavelet continua (ad es. `"mexh"`) la funzione `pywt.scale2frequency` fornisce il rapporto (cicli per campione), che va moltiplicato per f_s per ottenere la frequenza in Hz.

```
1 # Wavelet Mexican hat
2 wavelet_mexh = pywt.ContinuousWavelet("mexh")
3
4 # TODO: scegli alcune scale da provare
5 scales_test = [1, 5, 10]    # ad es.
6
7 print("Scala -> frequenza (Hz) per 'mexh':")
8 for s in scales_test:
9     f_cicli = pywt.scale2frequency(wavelet_mexh, s)  # cicli per
          campione
10     f_hz    = f_cicli * fs
11     print(f"  scala = {s:3d}  ->  f \approx {f_hz:7.2f} Hz")
```

TODO: ripetere il calcolo per altre wavelet continue disponibili in PyWavelets (ad esempio `"morl"`) e confrontare le frequenze corrispondenti alle diverse scale.

Passo 13: sinusoide a 50 Hz

Consideriamo una sinusoide:

$$x(t) = \sin(2\pi \cdot 50\, t)$$

campionata a 1000 Hz.

```
f0   = 50.0
Ns   = 1000            # TODO: puoi anche provare con Ns=100
t_sin = np.arange(Ns) / fs
x_sin = np.sin(2 * np.pi * f0 * t_sin)

plt.figure(figsize=(8,3))
plt.plot(t_sin, x_sin)
plt.xlabel("Tempo [s]")
plt.ylabel("x_sin(t)")
plt.title("Sinusoide a 50 Hz")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Passo 14: CWT della sinusoide (scale 1–32, wavelet `mexh`)

```
scales = np.arange(1, 33)

coeffs_sin, freqs_sin = pywt.cwt(
    x_sin,
    scales,
    "mexh",
    sampling_period=1.0/fs
)

plt.figure(figsize=(8,4))
plt.contour(
    np.abs(coeffs_sin),
    levels=8
)
plt.colorbar(label="|C(a,b)|")
plt.title("CWT (mexh) della sinusoide a 50 Hz - contour a 8 livelli")
plt.tight_layout()
plt.show()

# TODO: prova anche contourf(...) per un riempimento pieno
```

Domanda: si osservano picchi dei coefficienti alla scala attesa (cioè la scala che corrisponde a 50 Hz)?

Passo 15: ripetere con un'altra wavelet

Si vuole ripetere l'esperimento con un'altra wavelet continua (ad es. `"morl"`) e confrontare gli scalogrammi.

```
# TODO: sostituisci "mexh" con un'altra wavelet continua (es. "morl")
coeffs_sin_2, freqs_sin_2 = pywt.cwt(
    x_sin,
    scales,
    "mexh",         # TODO: cambia qui, ad es. "morl"
    sampling_period=1.0/fs
)

# TODO: confronta gli scalogrammi ottenuti.
```

Passo 16: CWT di `heart100`

Si applica ora la CWT al segnale ECG `heart100` normalizzato.

```
scales_ecg = np.arange(1, 101)    # TODO: prova anche scale piu' grandi

coeffs_ecg, freqs_ecg = pywt.cwt(
    x,    # ECG heart100 normalizzato
    scales_ecg,
    "mexh",
    sampling_period=1.0/fs
)

plt.figure(figsize=(8,4))
plt.imshow(
    np.abs(coeffs_ecg),
    extent=[t[0], t[-1], scales_ecg[-1], scales_ecg[0]],
    aspect="auto",
    origin="upper",
    cmap="viridis"
)
plt.xlabel("Tempo [s]")
plt.ylabel("Scala")
plt.title("Scalogramma CWT (mexh) - heart100")
plt.colorbar(label="|C(a,b)|")
plt.tight_layout()
plt.show()
```

Esercizio: conoscendo la relazione scala–frequenza per `"mexh"`:

- individuare la scala (o l'intervallo di scale) con coefficiente medio più elevato;

- convertire tale scala in frequenza (Hz) usando `freqs_ecg`;
- confrontare la frequenza così ottenuta con la frequenza dominante ricavata con la FFT.

Passo 17: coefficienti alla scala "giusta"

Si può estrarre una linea di coefficienti alla scala che interessa e plottarla nel tempo.

```
# TODO: scegli una scala target, ad es. scala_tipo = 10
scala_tipo = 10

# Trova l'indice in scales_ecg piu' vicino a scala_tipo
idx_scala = np.argmin(np.abs(scales_ecg - scala_tipo))

coeff_line = coeffs_ecg[idx_scala, :]

plt.figure(figsize=(8,3))
plt.plot(t, np.abs(coeff_line)**2)
plt.xlabel("Tempo [s]")
plt.ylabel("|C(a,b)|^2")
plt.title(f"Coefficiente CWT alla scala circa {scala_tipo}")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Esercizi:

- cercare a occhio la scala che massimizza l'energia dei coefficienti;
- verificare se corrisponde alla scala attesa dalla frequenza dominante;
- sperimentare scale molto piccole (1, 2, 3) per evidenziare eventuali oscillazioni ad alta frequenza e valutare l'idea di fare poi la FFT di tali coefficienti.