

Détection d'arythmies par transformée en ondelettes et diagnostic automatisé

SERROUKH SAAD

Session 2026



Figure 1 — *Diversité des signaux ECG : du rythme sinusal aux arythmies critiques*
<https://www.emjreviews.com/>

Les maladies cardiovasculaires : **32 %** de la mortalité mondiale
selon l'Organisation Mondiale de la Santé (OMS)

Peut-on automatiser la détection des maladies cardiaques ?

Plan de la présentation

1 Acquisition et prétraitement de l'ECG

- Acquisition
- Prétraitement

2 Analyse temps-fréquence par ondelettes

- Généralités
- Principe de la transformée en ondelettes

3 Étude dynamique par boucle d'hystérésis

4 Vers un diagnostic automatisé

5 Bilan

6 Bibliographie

7 Annexes

Qu'est-ce qu'un électrocardiogramme (ECG) ?

Définition

L'électrocardiogramme est la transcription graphique temporelle de l'activité électrique du cœur, captée à la surface de la peau.

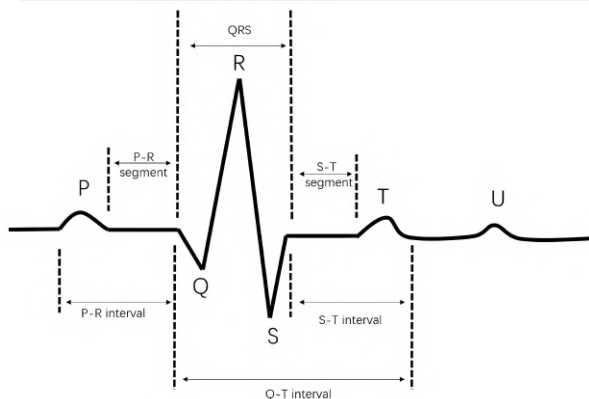


Figure 2 — *Signal ECG typique (sain)*

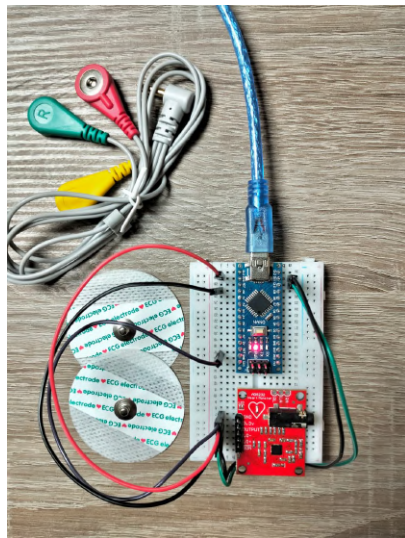
<https://encyclopedia.pub/>

Dispositif matériel.

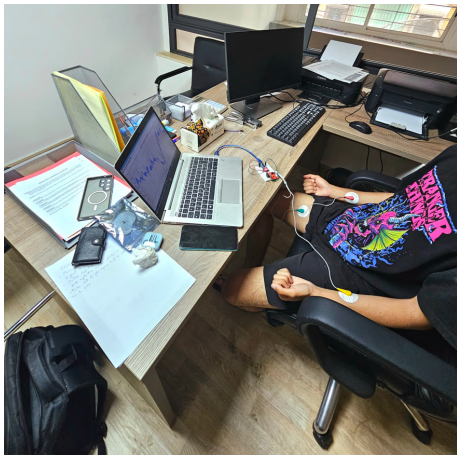
- Capteur ECG AD8232
- Microcontrôleur Arduino Nano
- Électrodes Ag/AgCl
- Breadboard + fils de connexion

Chaîne logicielle.

- Arduino IDE
- Python
- NumPy et Matplotlib



Mise en œuvre du protocole *in situ*



Procédé d'acquisition.

- 1 **Placement des électrodes** en triangle d'EINTHOVEN (bras droit, bras gauche, jambe droite).
- 2 **Repos stationnaire** : calibration de la ligne de base.
- 3 **Phase de capture** : acquisition continue pendant 5 minutes.

Observation du signal brut

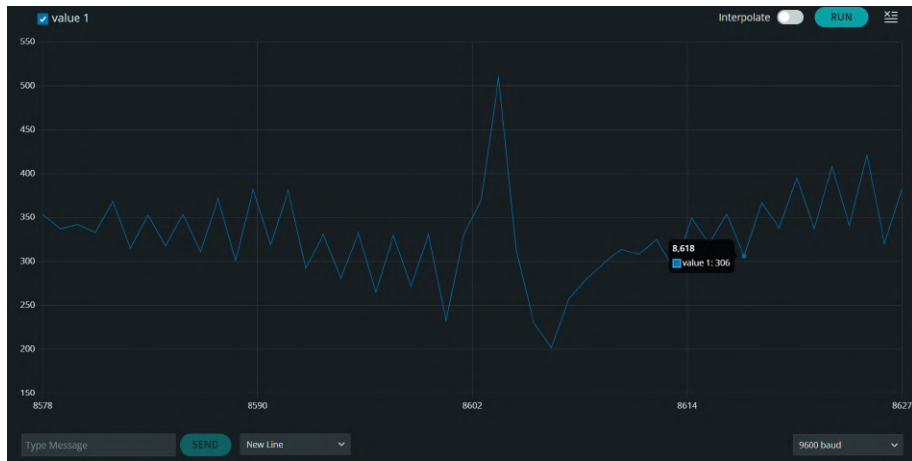


Figure 3 — Screenshot du traceur série Arduino.

Observation du signal brut

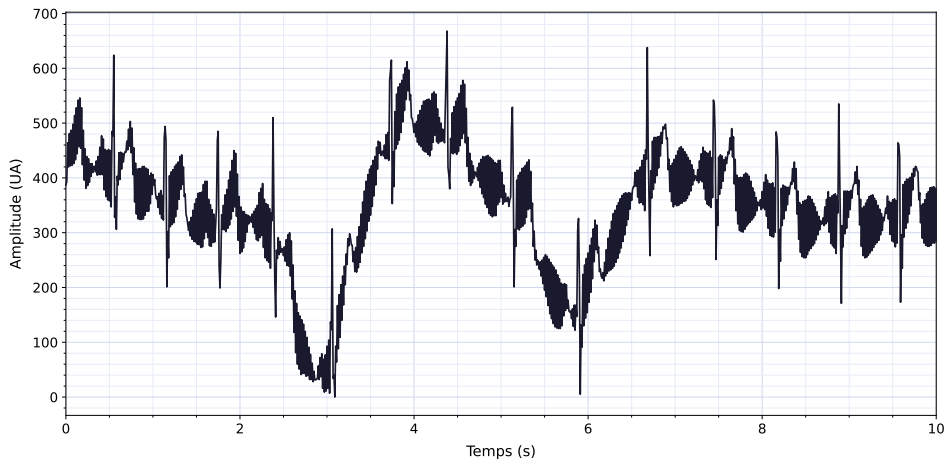


Figure 4 — *Retranscription sous Python de l'acquisition par Arduino.*

Le signal ECG brut est pollué par deux sources majeures :

- **Basse fréquence** ($< 0,5 \text{ Hz}$) : Dérive de la ligne de base (respiration, mouvements).
- **Haute fréquence** (50 Hz / **EMG**) : Interférences du secteur.

Solution ?

L'utilisation d'un filtre passe-bande pour isoler la bande utile du complexe QRS ($10 - 40 \text{ Hz}$).

Analyse multirésolution (AMR)

L'objectif est de localiser précisément les complexes QRS.
Le filtrage par AMR a la particularité d'être **sans déphasage**.

Principe de l'algorithme (1989) :

- Bancs de filtres itératifs.
- Séparation en *approximations* (événements lents) et *détails* (événements rapides).

Reconstruction exacte

$$S(t) = A_N(t) + \sum_{i=1}^N D_i(t)$$

Avec N le niveau de décomposition maximal atteint.



Stéphane MALLAT
<https://fr.wikipedia.org/>

Algorithme de MALLAT

- **Décomposition récursive**

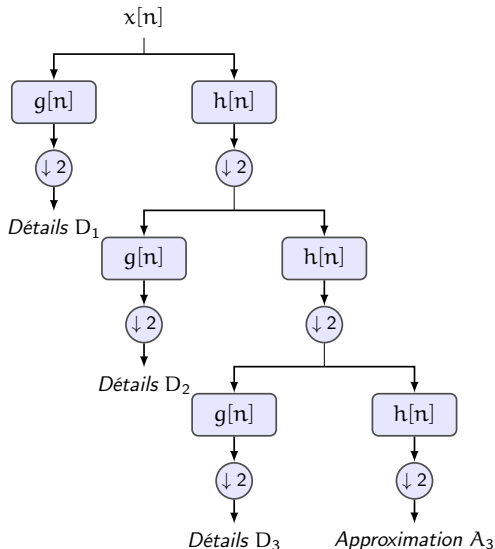
Séparation par banc de filtres miroir conjugués ($h[n]$, $g[n]$).

- **Décimation ($\downarrow 2$)**

Justifiée par le théorème de NYQUIST-SHANNON : on ne garde qu'un échantillon sur deux.

- **Seuillage & Synthèse**

Mise à zéro des coefficients de détails D_j (bruit) puis reconstruction **sans déphasage**.



Décomposition par AMR

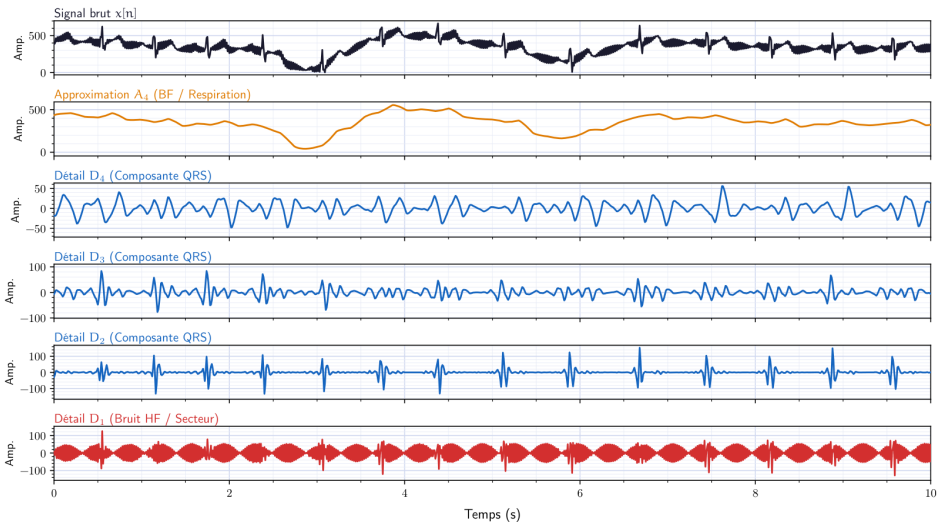


Figure 5 — Décomposition du signal ECG – Approximation A_4 et détails D_1 à D_4 .

Signal filtré par AMR

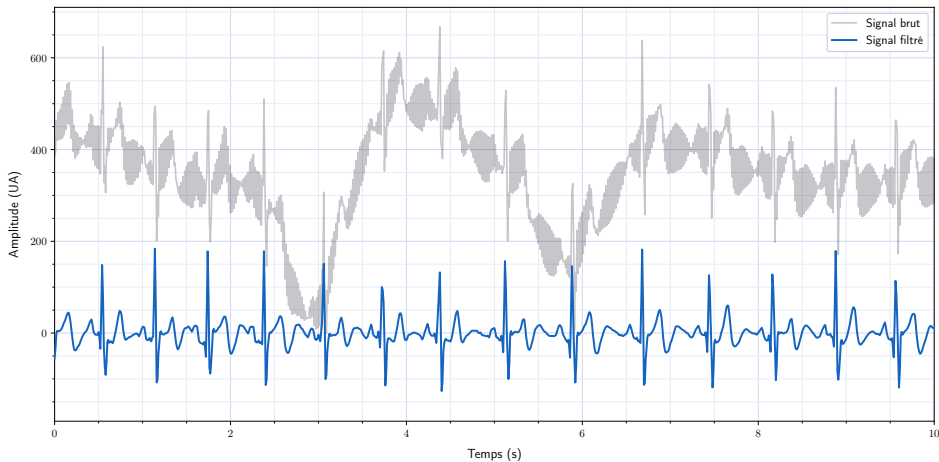


Figure 6 — *Comparaison entre signal brut et filtré par AMR.*

Plan de la présentation

- 1 Acquisition et prétraitement de l'ECG
 - Acquisition
 - Prétraitement
- 2 Analyse temps-fréquence par ondelettes
 - Généralités
 - Principe de la transformée en ondelettes
- 3 Étude dynamique par boucle d'hystérésis
- 4 Vers un diagnostic automatisé
- 5 Bilan
- 6 Bibliographie
- 7 Annexes

Un peu de contexte

Un signal physiologique comme l'ECG est fondamentalement **non-stationnaire** : son contenu spectral évolue brusquement au cours du temps.

Deux approches mathématiques s'imposent :

- La Transformée de FOURIER à Court Terme (**STFT**)
- La Transformée en Ondelettes Continue (**CWT**)

La capacité de la STFT à isoler un motif dépend de sa fenêtre, caractérisée par deux écarts-types (voir annexes) :

- **Étalement temporel** σ_t : incertitude sur la localisation dans le temps.
- **Étalement fréquentiel** σ_ω : incertitude sur la mesure de la fréquence.

Principe d'incertitude (d'HEISENBERG - GABOR)

La théorie du signal impose :

$$\sigma_t \cdot \sigma_\omega \geq \frac{1}{2}$$

Cas d'un signal crêteau

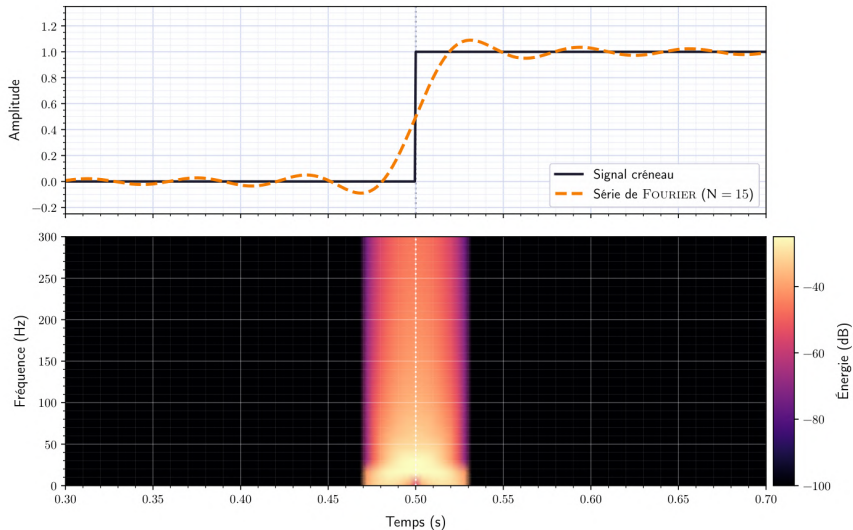


Figure 7 — Spectrogramme d'un signal crêteau.

Conclusion : la STFT est inadéquate pour notre objectif.

- On ne peut dater l'apparition d'un complexe QRS avec précision.
- Les oscillations dues au phénomène de GIBBS peuvent être confondues avec d'autres accidents physiologiques.

Alternative ?

Nous devons passer d'une approche temps-fréquence à une **analyse temps-échelle**.

Définition.

Une ondelette mère est une fonction $\psi \in L^2(\mathbb{R})$, d'énergie finie et bien localisée dans le temps (*i.e.* s'amortit rapidement).

Condition d'admissibilité

Pour être qualifiée d'ondelette, la fonction doit être à **moyenne nulle** :

$$\int_{-\infty}^{+\infty} \psi(t) dt = 0$$

Familles d'ondelettes

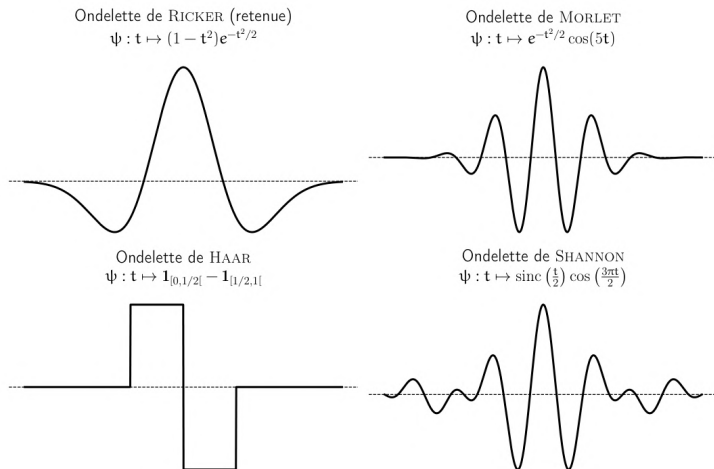


Figure 8 — *Quelques ondelettes classiques.*

Équation de la CWT

À partir de l'ondelette mère $\psi(t)$, on construit une famille infinie de fonctions en faisant varier deux paramètres continus :

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) \quad \text{avec } a > 0 \text{ et } b \in \mathbb{R}$$

Calcul des coefficients d'ondelettes

La Transformée en Ondelettes Continue (CWT) d'un signal s est définie par le produit scalaire dans $L^2(\mathbb{R})$:

$$C_s(a, b) = \langle s, \psi_{a,b} \rangle = \int_{-\infty}^{+\infty} s(t) \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) dt$$

- **Le paramètre b** est dit de translation.
- **Le paramètre a** est dit d'échelle. Il est inversement proportionnel à la fréquence.

Définition.

Le scalogramme $\mathcal{S}$ est le carré du module des coefficients de la CWT. Il représente la **densité d'énergie** du signal :

$$\mathcal{S}(a, b) = |C_s(a, b)|^2 = \left| \int_{-\infty}^{+\infty} s(t) \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) dt \right|^2$$

C'est plus intéressant graphiquement !

On obtient une matrice 2D où :

- L'axe des abscisses représente le **temps** (b).
- L'axe des ordonnées représente l'**échelle** (a).
- La couleur indique l'**intensité** de l'énergie.

Application à l'ECG

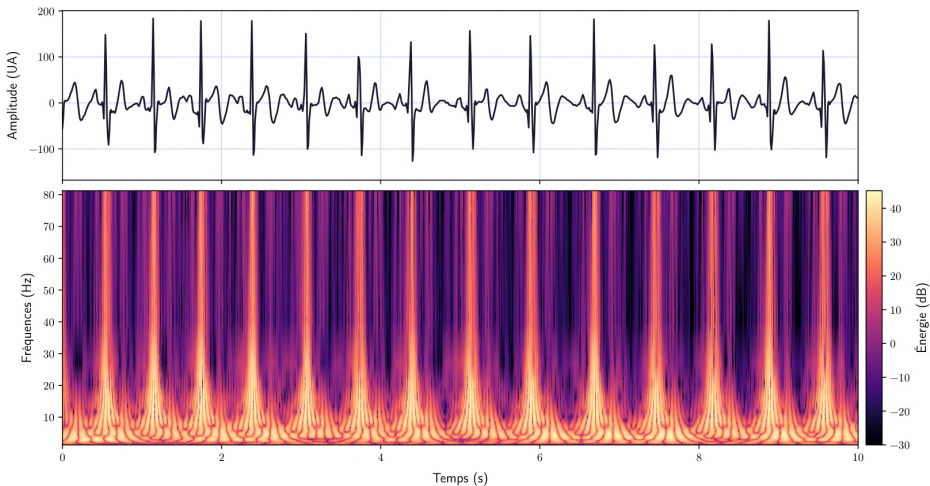


Figure 9 — Scalogramme du signal acquis *in situ* post-filtrage.

Plan de la présentation

- 1 Acquisition et prétraitement de l'ECG
 - Acquisition
 - Prétraitement
- 2 Analyse temps-fréquence par ondelettes
 - Généralités
 - Principe de la transformée en ondelettes
- 3 Étude dynamique par boucle d'hystérésis
- 4 Vers un diagnostic automatisé
- 5 Bilan
- 6 Bibliographie
- 7 Annexes

Qu'est-ce que la capacité aérobie ?

Définition

La capacité cardio-respiratoire ou capacité aérobie, notée $\text{VO}_{2,\text{max}}$, est la quantité maximale d'oxygène que l'organisme peut prélever et consommer par minute.

- C'est une composante majeure de la condition physique.
- Elle se mesure en **millilitres par minute** (mL/min).

<https://sante.lefigaro.fr>

L'évaluation du $\text{VO}_{2,\text{max}}$ nécessite aujourd'hui une infrastructure hospitalière lourde (masques d'échanges gazeux, ergomètres...).

⇒ C'est un examen coûteux et difficilement accessible.

Objectif

Peut-on estimer cette même grandeur physique complexe uniquement à partir d'un simple ECG acquis à l'effort ?

L'asymétrie de la réponse cardiaque

Le temps de récupération après un effort diffère fondamentalement de l'accélération : c'est le phénomène d'hystérésis.



Figure 10 — Jannik SINNER au Masters de Cincinnati, août 2025
<https://www.lequipe.fr>

Conséquence ?

Pour une même puissance mécanique externe $\mathcal{P}$, la fréquence cardiaque FC sera **plus élevée** en phase de descente qu'en phase de montée.

La relation $FC = f(\mathcal{P})$ n'est pas bijective : elle décrit une boucle fermée Γ .

Exemple issu d'une b.d.d PhysioNet

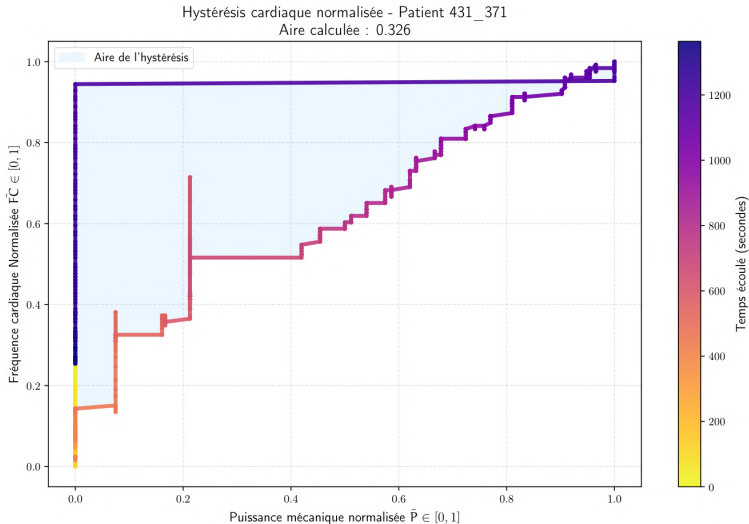


Figure 11 — Aire d'hystérésis d'un patient pris au hasard.

(Source : Cohorte UMA/Málaga)

Approche ? Utiliser une régression linéaire multiple (RLM) pour lier les descripteurs physiques et mathématiques au VO_2 .

$$\widehat{\text{VO}_2}(t) = \beta_0 + \beta_1 \cdot \text{Âge} + \beta_2 \cdot \text{Poids} + \beta_3 \cdot \mathcal{P}(t) + \beta_4 \cdot \text{FC}(t) + \beta_5 \cdot \mathcal{A}_{\text{norm}}$$

avec $\mathcal{A}_{\text{norm}}$ l'aire normalisée de la boucle d'hystérésis.

Validation du modèle

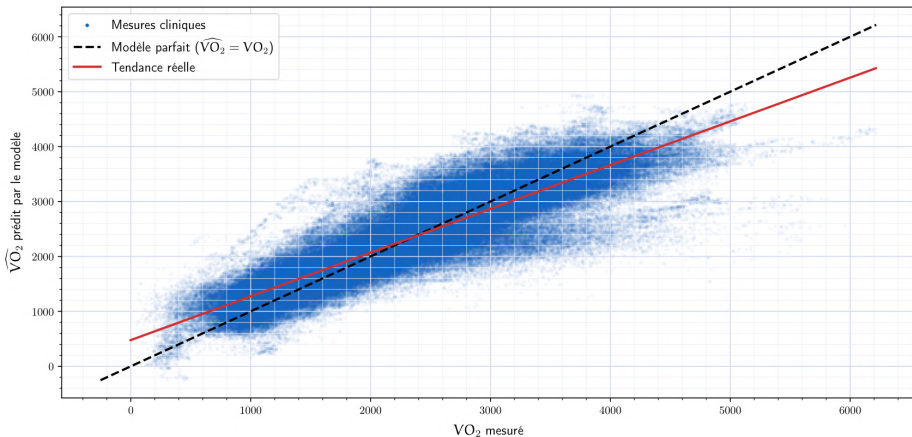


Figure 12 — Résultats de la régression.

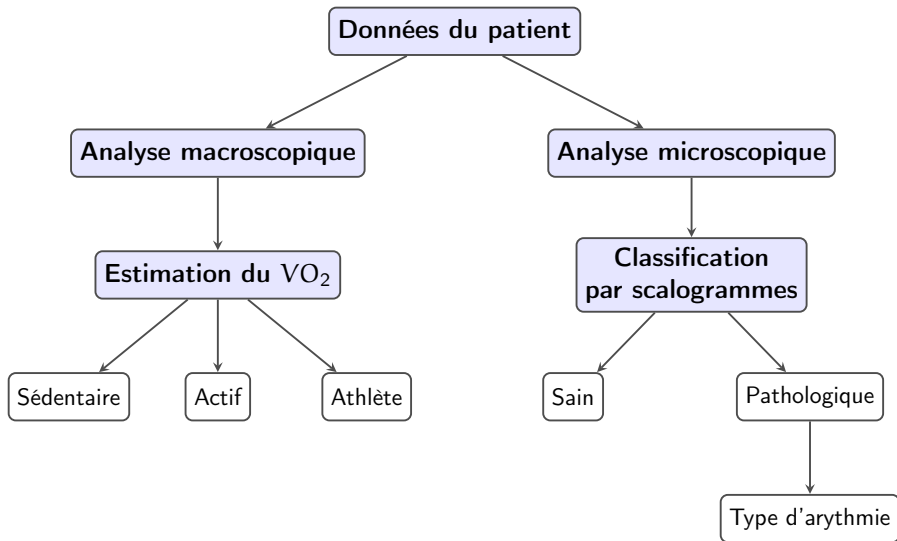
Variable	Coefficient β
$\mathcal{P}(t)$	+520.7954
FC(t)	+427.0530
$\mathcal{A}_{\text{norm}}$	+35.4266
Âge	+54.7501
Poids	+108.2432

Coefficient de détermination : $R^2 = 0.81$

Plan de la présentation

- 1 Acquisition et prétraitement de l'ECG
 - Acquisition
 - Prétraitement
- 2 Analyse temps-fréquence par ondelettes
 - Généralités
 - Principe de la transformée en ondelettes
- 3 Étude dynamique par boucle d'hystérésis
- 4 Vers un diagnostic automatisé
- 5 Bilan
- 6 Bibliographie
- 7 Annexes

Architecture unifiée du système d'analyse



Évaluation des performances sur la base MIT-BIH

Modèle IA	Précision	Sensibilité
KNN ($k = 5$)	73.0 %	34.0 %
MLP	69.3 %	34.2 %

Ces modèles sont bien peu sensibles aux arythmies !

Résultats détaillés en annexe.

Limites des modèles 1D et choix du CNN

L'aplatissement en vecteurs 1D (k-NN, MLP) détruit la topologie temps-fréquence des scalogrammes.

L'architecture convolutive (CNN) est la technologie la mieux adaptée pour traiter directement des images 2D.

Évaluation des performances sur la base MIT-BIH

Résultats du CNN via les bibliothèques Keras/TensorFlow.

Classe	Pathologie	Précision	Sensibilité
N	Rythme Normal	94.4 %	98.7 %
V	Arythmie Ventriculaire	79.4 %	81.4 %
S	Arythmie Supraventriculaire	23.7 %	3.6 %

Précision globale : 92.4 %

Résultats détaillés en annexe.

Plan de la présentation

- 1 Acquisition et prétraitement de l'ECG
 - Acquisition
 - Prétraitement
- 2 Analyse temps-fréquence par ondelettes
 - Généralités
 - Principe de la transformée en ondelettes
- 3 Étude dynamique par boucle d'hystérésis
- 4 Vers un diagnostic automatisé
- 5 **Bilan**
- 6 Bibliographie
- 7 Annexes

L'intégralité du projet est documentée et reproductible.

<https://github.com/ShirosakiWhite/TIPE-Arythmies>



Conclusion

Merci pour votre attention

Plan de la présentation

- 1 Acquisition et prétraitement de l'ECG
 - Acquisition
 - Prétraitement
- 2 Analyse temps-fréquence par ondelettes
 - Généralités
 - Principe de la transformée en ondelettes
- 3 Étude dynamique par boucle d'hystérésis
- 4 Vers un diagnostic automatisé
- 5 Bilan
- 6 Bibliographie**
- 7 Annexes

Références bibliographiques

- [1] J. PAN et W. J. TOMPKINS : *IEEE Transactions on Biomedical Engineering*, "A Real-Time QRS Detection Algorithm" : Vol. 32, no. 3 (1985), pp. 230–236.
- [2] A. CABASSON : *Analyse du rythme cardiaque et détection d'arythmies par méthodes temps-fréquence* : Thèse de doctorat, Université de Rennes 1, France, 2008.
- [3] S. MALLAT : *A Wavelet Tour of Signal Processing : The Sparse Way* : 3rd edition, Academic Press, 2008.
- [4] C. R. COLE et al. : *The New England Journal of Medicine*, "Heart-rate recovery immediately after exercise as a predictor of mortality" : Vol. 341, no. 18 (1999), pp. 1351–1357.
- [5] R. MARGARIA, P. CERRETELLI, P. AGHEMO et G. SASSI : *Journal of Applied Physiology*, "Energy cost of running" : Vol. 18, no. 2 (1963), pp. 367–370.
- [6] G. LIGUORI, Y. FEITO, C. FOUNTAINE et B. A. ROY : *ACSM's Guidelines for Exercise Testing and Prescription* : 11th ed., Lippincott Williams & Wilkins, 2021.
- [7] Université de Technologie de Compiègne : *Théorème d'incertitude d'Heisenberg–Gabor*, cours MT12 – https://moodle.utc.fr/file.php/332/ChapIV_TF/MT12_ChapIV5.pdf
- [8] D. MONGIN, J. GARCÍA ROMERO et J. R. ALVERO CRUZ : *Treadmill Maximal Exercise Tests from the Exercise Physiology and Human Performance Lab of the University of Málaga*, PhysioNet, 2021. <https://doi.org/10.13026/7ezk-j442>

Plan de la présentation

- 1 Acquisition et prétraitement de l'ECG
 - Acquisition
 - Prétraitement
- 2 Analyse temps-fréquence par ondelettes
 - Généralités
 - Principe de la transformée en ondelettes
- 3 Étude dynamique par boucle d'hystérésis
- 4 Vers un diagnostic automatisé
- 5 Bilan
- 6 Bibliographie
- 7 Annexes

Annexe A : Acquisition de signal ECG in situ

```
void setup() {  
    Serial.begin(9600);  
  
    pinMode(10, INPUT);  
    pinMode(11, INPUT);  
}  
  
void loop() {  
    if ((digitalRead(10) == 1) || (digitalRead(11) == 1)) {  
        Serial.println(0);  
    }  
    else {  
        Serial.println(analogRead(A0));  
    }  
  
    delay(10);  
}
```

Annexe B : Script d'enregistrement PC (30s)

```
import serial, time

PORT, BAUD, TEMPS_ACQ = 'COM7', 9600, 30
ser = serial.Serial(PORT, BAUD)
time.sleep(2)
t_end = time.time() + TEMPS_ACQ

try:
    with open('mon_ecg_brut.csv', 'w') as f:
        f.write("Amplitude\n")
        while time.time() < t_end:
            try:
                ligne = ser.readline().decode('utf-8').strip()
                if ligne.isdigit(): f.write(f"{ligne}\n")
            except:
                pass
finally:
    if ser.is_open: ser.close()
```

Annexe C : Tracé du signal ECG brut

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

data = np.loadtxt('mon_ecg_brut.csv', skiprows=1)
temps = np.arange(len(data)) / 100

plt.figure(figsize=(12, 6))
plt.plot(temps, data, color='black', linewidth=1.5)

plt.xlim(0, 10)
plt.xlabel("Temps (s)")
plt.ylabel("Amplitude (UA)")
plt.grid(True)

plt.tight_layout()
plt.show()
```

Annexe D : Algorithme de MALLAT et choix des filtres

- Le filtre passe-haut $g[n]$ est déduit directement du filtre passe-bas $h[n]$ par inversion temporelle et alternance de signe :

$$g[n] = (-1)^n h[L - 1 - n]$$

avec L la longueur de la réponse impulsionnelle.

- L'ondelette db4 (DAUBECHIES) est définie exclusivement par les coefficients de son filtre discret $h[n]$ ($L = 8$ coefficients).

Annexe E : Décomposition de MALLAT et CWT

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

data = np.loadtxt('mon_ecg_brut.csv', skiprows=1)
n, temps = len(data), np.arange(len(data)) / 100

# 1. DÉCOMPOSITION DE MALLAT (AMR)
coeffs = pywt.wavedec(data, 'sym4', level=4)

def extraire(c_all, idx, wave, n_len):
    c_tmp = [np.zeros_like(x) for x in c_all]
    c_tmp[idx] = c_all[idx]
    return pywt.waverec(c_tmp, wave)[:n_len]

# Extraction en boucle (D4, D3, D2, D1)
D = [extraire(coeffs, i, 'sym4', n) for i in range(1, 5)]

# 2. RECONSTRUCTION ET SCALOGRAMME (CWT)
signal_filtre = D[0] + D[1] + D[2] # D4 + D3 + D2
cwt_coefs, freqs = pywt.cwt(signal_filtre, np.arange(1, 65), 'morl', sampling_period=0.01)

plt.plot(temps, signal_filtre)
plt.pcolormesh(temps, freqs, 10 * np.log10(np.abs(cwt_coefs)**2 + 1e-12), cmap='magma')
plt.show()
```

Annexe F : Code de filtrage du signal (AMR)

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

data = np.loadtxt('mon_ecg_brut.csv', skiprows=1)
temps = np.arange(len(data)) / 100

# 1. DÉCOMPOSITION DE MALLAT
coeffs = pywt.wavedec(data, 'sym4', level=4)

# 2. SEUILLAGE : Suppression de la dérive (A4) et du bruit (D1)
coeffs[0] = np.zeros_like(coeffs[0])
coeffs[-1] = np.zeros_like(coeffs[-1])

# 3. RECONSTRUCTION
data_filtree = pywt.waverec(coeffs, 'sym4')[:len(data)]

# Affichage basique de comparaison
plt.plot(temps, data, alpha=0.3, label='Signal brut')
plt.plot(temps, data_filtree, color='blue', label='Signal filtré')

plt.xlim(0, 10); plt.legend(); plt.show()
```

L'espace de HILBERT $L^2(\mathbb{R})$

C'est l'espace des fonctions de carré intégrable (signaux à énergie finie) :

$$L^2(\mathbb{R}) = \left\{ f : \mathbb{R} \rightarrow \mathbb{C} \mid \int_{-\infty}^{+\infty} |f(t)|^2 dt < \infty \right\}$$

Cet espace est muni de :

- **Son produit scalaire usuel :**

$$\langle f, g \rangle = \int_{-\infty}^{+\infty} f(t) \overline{g(t)} dt$$

- **Sa norme associée :**

$$E_f = \|f\|^2 = \langle f, f \rangle = \int_{-\infty}^{+\infty} |f(t)|^2 dt$$

Annexe G.2 : Transformée de Fourier à Court Terme (STFT)

Soit un signal $s \in L^2(\mathbb{R})$ non nul.

Définition de la STFT

On multiplie le signal $s(t)$ par une **fenêtre d'analyse** $h(t)$ que l'on fait glisser le long de l'axe temporel (translation $\tau \in \mathbb{R}$) :

$$\text{STFT}_s(\tau, \omega) = \int_{-\infty}^{+\infty} s(t) h(t - \tau) e^{-j\omega t} dt$$

Notons de plus les centres temporel et fréquentiel de la fenêtre d'énergie unitaire de la façon suivante :

$$m_t(h) = \int_{-\infty}^{+\infty} t |h(t)|^2 dt \quad \text{et} \quad m_\omega(h) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \omega |\hat{h}(\omega)|^2 d\omega$$

Annexe G.3 : Étalements caractéristiques et incertitude

Les étalements caractéristiques (écarts-types) de la fenêtre sont définis par :

Étalement temporel (σ_t)

$$\sigma_t = \sqrt{\int_{-\infty}^{+\infty} (t - m_t(h))^2 |h(t)|^2 dt}$$

Étalement fréquentiel (σ_ω)

$$\sigma_\omega = \sqrt{\frac{1}{2\pi} \int_{-\infty}^{+\infty} (\omega - m_\omega(h))^2 |\hat{h}(\omega)|^2 d\omega}$$

Ces valeurs caractérisent le *pouvoir de résolution*. Minimiser un σ maximise la précision de localisation.

Principe d'incertitude (d'HEISENBERG - GABOR)

Pour toute fenêtre d'analyse $h \in L^2(\mathbb{R})$, la théorie impose :

$$\sigma_t \cdot \sigma_\omega \geq \frac{1}{2}$$

Annexe G.4 : Démonstration (HEISENBERG - GABOR) I

Soit $h \in L^2(\mathbb{R})$ à décroissance rapide avec $\|h\| = 1$.

On introduit $g(t) = h(t + m_t)e^{-jm_\omega t}$ pour recentrer l'étalement :

$$m_t(g) = m_\omega(g) = 0$$

L'étalement conjoint s'écrit alors :

$$\sigma_t \cdot \sigma_\omega = \left(\int_{-\infty}^{+\infty} t^2 |g(t)|^2 dt \right)^{1/2} \left(\int_{-\infty}^{+\infty} \omega^2 \frac{|\hat{g}(\omega)|^2}{2\pi} d\omega \right)^{1/2}$$

Avec $\hat{g}'(\omega) = j\omega \hat{g}(\omega)$, l'égalité de PARSEVAL donne directement :

$$\sigma_t \cdot \sigma_\omega = \|tg\| \cdot \|g'\|$$

Annexe G.4 : Démonstration (HEISENBERG - GABOR) II

L'inégalité de CAUCHY-SCHWARZ, minorée par la partie réelle, donne :

$$\sigma_t \cdot \sigma_\omega \geq |\langle tg, g' \rangle| \geq \left| \operatorname{Re} \int_{-\infty}^{+\infty} t g(t) \overline{g'(t)} dt \right|$$

Or $\operatorname{Re}(g\overline{g'}) = \frac{1}{2}(|g|^2)'$. Une intégration par parties mène à :

$$\sigma_t \cdot \sigma_\omega \geq \left| \left[\frac{t}{2} |g(t)|^2 \right]_{-\infty}^{+\infty} - \frac{1}{2} \int_{-\infty}^{+\infty} |g(t)|^2 dt \right|$$

Or par hypothèse sur h : $\lim_{|t| \rightarrow \infty} t |g(t)|^2 = 0$.

Puisque la norme est invariante par translation, $\|g\|^2 = 1$ et a fortiori :

$$\sigma_t \cdot \sigma_\omega \geq \frac{1}{2}$$

C.Q.F.D.

Annexe H : Spectrogramme et discrétisation

Le spectrogramme représente la densité spectrale d'énergie d'un signal. Il est défini comme le carré du module de la STFT :

$$\text{STFT}_x(\tau, \omega) = \left| \int_{-\infty}^{+\infty} x(t) h(t - \tau) e^{-j\omega t} dt \right|^2$$

Pour passer au calcul numérique, le signal est découpé en blocs de taille

$L = 128$ et utilise la fenêtre de HANN :

$$h[n] = \frac{1}{2} \left(1 - \cos \left(\frac{2\pi n}{L-1} \right) \right)$$

La version évaluée par l'algorithme pour le segment m et l'indice fréquentiel k s'écrit alors :

$$\text{STFT}_x[m, k] = \sum_{n=0}^{L-1} x[n] h[n-m] e^{-j2\pi \frac{kn}{L}}$$

Annexe I : Code de génération du spectrogramme

```
import numpy as np, matplotlib.pyplot as plt
from scipy.signal import spectrogram

# 1. GÉNÉRATION DES SIGNAUX
fs = 2000
t = np.linspace(0, 1, fs, endpoint=False)

# Signal créneau idéal (saut à t = 0.5s)
signal = np.zeros_like(t)
signal[t >= 0.5] = 1.0

# Approximation par série de Fourier (N = 15)
approx = np.ones_like(t) * 0.5
for k in range(1, 16, 2):
    approx -= (2 / (np.pi * k)) * np.sin(2 * np.pi * k * t)

# 2. CALCUL DU SPECTROGRAMME (SciPy)
f, t_spec, Sxx = spectrogram(signal, fs, nperseg=128, noverlap=120, window='hann')

# 3. AFFICHAGE BASIQUE (Superposition et Pcolormesh)
fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True)

ax1.plot(t, signal, label='Créneau')
ax1.plot(t, approx, linestyle='--', label='Approximation STFT')
ax1.legend()

ax2.pcolormesh(t_spec, f, 10 * np.log10(Sxx + 1e-12), cmap='magma', shading='gouraud')
ax2.set_ylim(0, 300)
ax2.set_xlim(0.3, 0.7)

plt.show()
```

Annexe K : Génération du dictionnaire d'ondelettes

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

t = np.linspace(-4, 4, 1000)
t_h = np.linspace(-1, 2, 1000)

# 1. Ricker (Chapeau Mexicain) -> Retenue pour le QRS
y_mex = (1 - t**2) * np.exp(-t**2 / 2)

# 2. Morlet (Partie réelle)
y_mor = np.exp(-t**2 / 2) * np.cos(5 * t)

# 3. Haar
y_har = np.piecewise(t_h, [t_h < 0, (t_h>=0) & (t_h<0.5), (t_h>=0.5) & (t_h<1), t_h>=1], [0, 1, -1, 0])

# 4. Shannon
y_sha = np.sinc(t/2) * np.cos(3 * np.pi * t / 2)

# Affichage optimisé en boucle
fig, axs = plt.subplots(2, 2, figsize=(10, 7))
donnees = zip(axs.flat, [t, t, t_h, t], [y_mex, y_mor, y_har, y_sha],
               ['Ricker', 'Morlet', 'Haar', 'Shannon'])

for ax, x, y, titre in donnees:
    ax.plot(x, y, color='black', linewidth=2)
    ax.set_title(titre, pad=15)
    ax.axhline(0, color='black', linewidth=0.8, linestyle='--')
    ax.axis('off')

plt.tight_layout(); plt.show()
```

Annexe L.1 : Condition d'admissibilité

Pour qu'une CWT soit inversible, l'ondelette mère ψ doit vérifier la **condition d'admissibilité** :

$$C_\psi = \int_{-\infty}^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty$$

Si $\psi \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$, alors sa transformée de FOURIER $\hat{\psi}$ est continue.
Par l'absurde, si $\hat{\psi}(0) = L \neq 0$, alors par continuité en zéro :

$$|\hat{\psi}(\omega)|^2 \xrightarrow{\omega \rightarrow 0} |L|^2 > 0$$

On obtient donc l'équivalent :

$$\frac{|\hat{\psi}(\omega)|^2}{|\omega|} \underset{\omega \rightarrow 0}{\sim} \frac{|L|^2}{|\omega|}$$

Annexe L.2 : Condition d'admissibilité

Or, l'intégrale de RIEMANN $\int_0^\varepsilon \frac{1}{\omega} d\omega$ diverge pour tout $\varepsilon > 0$.

Par théorème d'équivalence des intégrales de fonctions positives, l'intégrale définissant C_ψ diverge en zéro, ce qui contredit l'hypothèse d'admissibilité.

Par conséquent :

$$\hat{\psi}(0) = \int_{-\infty}^{+\infty} \psi(t) dt = 0$$

Annexe M.1 : Justification du filtrage AMR

La fréquence d'échantillonnage vaut $f_e = 100$ Hz, la fréquence de NYQUIST est de 50 Hz.

À chaque niveau j , la bande de fréquence est divisée par deux :

- **Niveau 1** : Détails $D_1 \in [25, 50]$ Hz
- **Niveau 2** : Détails $D_2 \in [12.5, 25]$ Hz
- **Niveau 3** : Détails $D_3 \in [6.25, 12.5]$ Hz
- **Niveau 4** : Détails $D_4 \in [3.12, 6.25]$ Hz et approximation $A_4 \in [0, 3.12]$ Hz

Stratégie de seuillage

Le complexe QRS concentre son énergie entre 5 et 15 Hz. En poussant la décomposition au **niveau 4** et en annulant les coefficients d'approximation A_4 et de détail D_1 , on crée un filtre passe-bande effectif sur l'intervalle $[3.12, 25]$ Hz, idéal pour isoler le rythme cardiaque.

Annexe M.2 : Code complet (filtrage + scalogramme)

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

# 1. CHARGEMENT ET FILTRAGE AMR (Isolant la bande 3-25 Hz)
data = np.loadtxt('mon_ecg_brut.csv', skiprows=1)
temps = np.arange(len(data)) / 100 # fe = 100 Hz

coeffs = pywt.wavedec(data, 'sym4', level=4)
coeffs[0] = np.zeros_like(coeffs[0]) # Suppression A4 (Dérive)
coeffs[-1] = np.zeros_like(coeffs[-1]) # Suppression D1 (Bruit HF)
signal_filtre = pywt.waverec(coeffs, 'sym4')[:len(data)]

# 2. TRANSFORMÉE CONTINUE EN ONDELETTE (CWT)
# Utilisation du "Mexican Hat" (Ricker), morphologiquement adapté au QRS
coefs_cwt, frequences = pywt.cwt(signal_filtre, np.arange(1, 65), 'mexh', sampling_period=0.01)

# CORRECTION : Passage en décibels (dB)
energie_cwt_db = 10 * np.log10(np.abs(coefs_cwt)**2 + 1e-12)

# 3. AFFICHAGE
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 6), sharex=True, gridspec_kw={'height_ratios': [1, 1.5]})

ax1.plot(temps, signal_filtre, color='black', linewidth=1)
ax1.set_ylabel("Amplitude")

# Utilisation de l'énergie en dB et rasterisation
im = ax2.pcolormesh(temps, frequences, energie_cwt_db, cmap='magma', shading='gouraud', rasterized=True)
ax2.set_ylabel("Fréquences (Hz)")
ax2.set_xlabel("Temps (s)")

fig.colorbar(im, ax=ax2, label="Énergie (dB)")
plt.tight_layout(); plt.show()
```

Annexe M.3 : Échelle et pseudo-fréquence

La CWT s'opère sur un paramètre de dilatation abstrait α . Pour lui donner un sens physique sur le scalogramme, on le convertit en **pseudo-fréquence** F_α (en Hz). La relation mathématique est la suivante :

$$F_\alpha = \frac{F_c}{\alpha \cdot T_e}$$

- α : échelle d'analyse.
- T_e : période d'échantillonnage du signal.
- F_c : fréquence centrale de l'ondelette mère.

Énoncé.

Soit $\mathcal{D}$ un compact de $\mathbb{R}^2$ et Γ sa courbe frontière.

On suppose Γ fermée, simple, et orientée positivement (parcourue dans le sens trigonométrique).

Soit un champ de vecteurs $\vec{F}(x, y) = U(x, y)\vec{i} + V(x, y)\vec{j}$ de classe $\mathcal{C}^1$ sur un ouvert contenant $\mathcal{D}$. La circulation du champ sur le bord est égale à l'intégrale double de son rotationnel scalaire sur le domaine :

$$\oint_{\Gamma} (U \, dx + V \, dy) = \iint_{\mathcal{D}} \left(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y} \right) dx dy$$

Annexe N.1b : Application au calcul d'aire

Pour exprimer l'aire du domaine $\mathcal{A} = \iint_{\mathcal{D}} dx dy$, on cherche un champ tel que $\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y} = 1$. Le choix antisymétrique $U(x, y) = -\frac{y}{2}$ et $V(x, y) = \frac{x}{2}$ satisfait cette condition.

Dans le plan de phase de notre essai clinique, l'axe des abscisses est la puissance normalisée $\tilde{\mathcal{P}}$ et l'ordonnée est la fréquence cardiaque normalisée $\tilde{FC}$.

L'aire de la boucle d'hystérésis s'écrit alors :

$$\mathcal{A} = \frac{1}{2} \oint_{\Gamma} \left(\tilde{\mathcal{P}} d\tilde{FC} - \tilde{FC} d\tilde{\mathcal{P}} \right)$$

La formule des lacets (*Shoelace formula*)

Le signal clinique est un ensemble discret de N points $(\tilde{\mathcal{P}}_i, \tilde{F}\tilde{C}_i)$ échantillonnés au cours du temps. La courbe Γ devient un polygone fermé.

On approxime les différentielles $d\tilde{\mathcal{P}}$ et $d\tilde{F}\tilde{C}$ par les accroissements entre deux instants successifs $\tilde{\mathcal{P}}_{i+1} - \tilde{\mathcal{P}}_i$ et $\tilde{F}\tilde{C}_{i+1} - \tilde{F}\tilde{C}_i$. Par télescopage des termes de la somme, l'aire s'approxime par :

$$\mathcal{A} \approx \frac{1}{2} \left| \sum_{i=1}^N (\tilde{\mathcal{P}}_i \cdot \tilde{F}\tilde{C}_{i+1} - \tilde{F}\tilde{C}_i \cdot \tilde{\mathcal{P}}_{i+1}) \right|$$

C'est précisément cette formule qui est utilisée dans le code Python de l'annexe N.2.

Annexe N.2a : Code d'extraction et tracé de l'hystérésis

```
import pandas as pd, numpy as np, matplotlib.pyplot as plt, random, os
from matplotlib.collections import LineCollection

# 1. CALCUL D'AIRE (Green-Riemann discrétisé)
def calcul_aire_lacet(x, y):
    return 0.5 * np.abs(np.dot(x, np.roll(y, 1)) - np.dot(y, np.roll(x, 1)))

# 2. CHARGEMENT DONNÉES
dossier_courant = os.path.dirname(os.path.abspath(__file__))
df_info = pd.read_csv(os.path.join(dossier_courant, 'subject-info.csv'))
df_measure = pd.read_csv(os.path.join(dossier_courant, 'test-measure.csv'))

df = pd.merge(df_measure, df_info[['ID_test', 'Weight']], on='ID_test', how='inner')
df = df.dropna(subset=['HR', 'Speed']).sort_values(by=['ID_test', 'time'])

# 3. TRAITEMENT PATIENT ALÉATOIRE
patients_valides = df['ID_test'].value_counts()[df['ID_test'].value_counts() > 50].index
patient_id = random.choice(patients_valides)
df_patient = df[df['ID_test'] == patient_id].copy()

# Puissance brute et normalisation Min-Max stricte [0, 1]
df_patient['Puissance'] = df_patient['Speed'] * df_patient['Weight']
P_min, P_max = df_patient['Puissance'].min(), df_patient['Puissance'].max()
HR_min, HR_max = df_patient['HR'].min(), df_patient['HR'].max()

df_patient['P_norm'] = (df_patient['Puissance'] - P_min) / (P_max - P_min)
df_patient['HR_norm'] = (df_patient['HR'] - HR_min) / (HR_max - HR_min)

aire = calcul_aire_lacet(df_patient['P_norm'].values, df_patient['HR_norm'].values)
```

Annexe N.2b : Code d'extraction et tracé de l'hystérésis

4. TRACÉ DU GRAPHIQUE

```
plt.figure(figsize=(10, 7))
x, y, t = df_patient['P_norm'].values, df_patient['HR_norm'].values, df_patient['time'].values

plt.fill(x, y, color='#e0f3ff', alpha=0.6, zorder=0, label="Aire de l'hystérésis")

points = np.array([x, y]).T.reshape(-1, 1, 2)
segments = np.concatenate([points[:-1], points[1:]], axis=1)
norm = plt.Normalize(t.min(), t.max())
lc = LineCollection(segments, cmap='plasma_r', norm=norm, linewidth=3.5, alpha=0.9, zorder=2)
lc.set_array(t)

ax = plt.gca()
ligne = ax.add_collection(lc)
plt.scatter(x, y, c=t, cmap='plasma_r', s=15, zorder=3, edgecolors='none')

plt.xlim(-0.05, 1.05); plt.ylim(-0.05, 1.05)
plt.xlabel(r'Puissance mécanique normalisée  $\tilde{P}$  in  $[0,1]$ ', fontsize=12)
plt.ylabel(r'Fréquence cardiaque Normalisée  $\tilde{FC}$  in  $[0,1]$ ', fontsize=12)
plt.title(f'Hystérésis cardiaque normalisée - Patient {patient_id}\nAire calculée : {aire:.3f}',
↵  fontsize=14, fontweight='bold')

cbar = plt.colorbar(ligne, ax=ax)
cbar.set_label('Temps écoulé (secondes)', fontsize=12)

plt.grid(True, linestyle='--', alpha=0.4); plt.legend(loc='upper left', fontsize=11)
plt.tight_layout(); plt.show()
```

Annexe O : Modélisation biomécanique de l'effort

On note E l'énergie métabolique servant au travail interne lors de la course sur tapis : relance de la masse à chaque foulée, élévation du centre de gravité... Expérimentalement, elle est proportionnelle au poids et à la distance l :

$$E = C_r \cdot m \cdot g \cdot l$$

où C_r est le coût de transport (*constante de rendement biomécanique*).
En dérivant par rapport au temps, la puissance métabolique $\mathcal{P}$ s'écrit :

$$\mathcal{P} = \frac{dE}{dt} = C_r \cdot m \cdot g \cdot \frac{dl}{dt} = C_r \cdot m \cdot g \cdot v$$

L'étude de l'hystérésis se fait sur la puissance normalisée $\tilde{\mathcal{P}} \in [0, 1]$.

$$\tilde{\mathcal{P}} = \frac{\mathcal{P} - \mathcal{P}_{\min}}{\mathcal{P}_{\max} - \mathcal{P}_{\min}} = \frac{C_r m g v - C_r m g v_{\min}}{C_r m g v_{\max} - C_r m g v_{\min}} = \frac{mv - mv_{\min}}{mv_{\max} - mv_{\min}}$$

L'utilisation du produit Poids * Vitesse dans le code est donc rigoureuse.

Annexe P.1 : Modélisation RLM (Entraînement)

```
import pandas as pd, numpy as np, matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score

# 1. PRÉPARATION DES DONNÉES
df = pd.read_csv("dataset_malaga_puissance.csv")

colonnes_entrees = ['Age', 'Poids', 'Puissance', 'FC', 'Aire_Hysteresis']
colonnes_totales = colonnes_entrees + ['VO2_Cible']

# Nettoyage strict des données manquantes
df = df.dropna(subset=colonnes_totales)
X = df[colonnes_entrees]
Y = df['VO2_Cible']

# 2. STANDARDISATION (Centrage et Réduction)
# Étape cruciale pour comparer les poids (coefficients bêta) de variables d'unités différentes
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 3. ENTRAÎNEMENT DU MODÈLE RLM
modele = LinearRegression()
modele.fit(X_scaled, Y)

# 4. PRÉDICTION ET ÉVALUATION
Y_pred = modele.predict(X_scaled)
r2 = r2_score(Y, Y_pred)
print(f"Précision globale du modèle ( $R^2$ ) : {r2:.4f}")
```

Annexe P.2 : Modélisation RLM (Visualisation)

5. TRACÉ DU GRAPHIQUE D'ÉVALUATION

```
fig, ax = plt.subplots(figsize=(9, 7))
```

```
ax.grid(True, which='major', color='#d0d8f0', linewidth=0.8)
```

```
ax.grid(True, which='minor', color='#e8ecf8', linewidth=0.4)
```

```
ax.minorticks_on()
```

Nuage de points

```
ax.scatter(Y, Y_pred, alpha=0.05, color='#1565c0', s=5, label="Mesures cliniques")
```

Droite de prédiction idéale ($y = x$)

```
min_val = min(Y.min(), Y_pred.min())
```

```
max_val = max(Y.max(), Y_pred.max())
```

```
ax.plot([min_val, max_val], [min_val, max_val], color='black', linewidth=2, linestyle='--',  
        label=r'Modèle parfait ( $\widehat{V0_2} = V0_2$ )')
```

Droite de régression réelle du nuage (Tendance observée)

```
m, b = np.polyfit(Y, Y_pred, 1)
```

```
ax.plot(Y, m*Y + b, color='#d32f2f', linewidth=2, label=r'Tendance réelle')
```

```
ax.set_xlabel(r"$V0_2$ mesuré", fontsize=13)
```

```
ax.set_ylabel(r"$\widehat{V0_2}$ prédit par le modèle", fontsize=13)
```

```
leg = ax.legend(fontsize=12, loc='upper left', framealpha=0.9)
```

```
for lh in leg.legend_handles:
```

```
    lh.set_alpha(1)
```

```
fig.subplots_adjust(left=0.12, right=0.95, top=0.90, bottom=0.12)
```

```
plt.show()
```

Annexe Q.1a : k-NN et MLP (Initialisation)

```
import numpy as np, pywt, wfdb, gc
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report
from skimage.transform import resize

# 1. SÉPARATION CLINIQUE INTER-PATIENTS STRICTE
DS1_TRAIN = ['101', '106', '108', '109', '112', '114', '115', '116', '118', '119', '122', '124', '201',
↳ '203', '205', '207', '208', '209', '215', '220', '223', '230']
DS2_TEST = ['100', '103', '105', '111', '113', '117', '121', '123', '200', '202', '210', '212', '213',
↳ '214', '219', '221', '222', '228', '231', '232', '233', '234']

AAMI_MAPPING = {'N':0, 'L':0, 'R':0, 'e':0, 'j':0, 'A':1, 'a':1, 'J':1, 'S':1, 'V':2, 'E':2, 'F':3, '/':4,
↳ 'f':4, 'Q':4}

# 2. FILTRAGE AMR (Suppression de la dérive)
def filtrage_amr_mitbih(signal_brut):
    coeffs = pywt.wavedec(signal_brut, 'db4', level=8)
    coeffs[0] = np.zeros_like(coeffs[0])
    return pywt.waverec(coeffs, 'db4')[:len(signal_brut)]
```

Annexe Q.1b : k-NN et MLP (Extraction)

3. EXTRACTION TEMPS-FRÉQUENCE ET APLATISSEMENT

```
def extraire_vecteurs_id_stricts(liste_patients, max_par_classe=400):
    X_aplatis, Y = [], []
    echelles, limites = np.arange(1, 33), {i: max_par_classe for i in range(5)}

    for rec in liste_patients:
        try:
            record = wfdb.rdrecord(rec, pn_dir='mitdb')
            annot = wfdb.rdann(rec, 'atr', pn_dir='mitdb')
        except: continue

        signal_propre = filtrage_amr_mitbih(record.p_signal[:, 0])
        compteur = {0: 0, 1: 0, 2: 0, 3: 0, 4: 0}

        for pos, sym in zip(annot.sample, annot.symbol):
            if sym in AAMI_MAPPING and compteur[AAMI_MAPPING[sym]] < limites[AAMI_MAPPING[sym]]:
                c = AAMI_MAPPING[sym]
                if pos - 130 > 0 and pos + 50 < len(signal_propre):
                    # Calcul de la CWT locale
                    coefs, _ = pywt.cwt(signal_propre[pos-130 : pos+50], echelles, 'mexh')

                    # Destruction de la matrice 2D en vecteur 1D
                    scalo_reduit = resize(np.log1p(np.abs(coefs)**2), (32, 32), anti_aliasing=True)
                    X_aplatis.append(scalo_reduit.flatten())
                    Y.append(c)
                    compteur[c] += 1
        gc.collect()

    return np.array(X_aplatis, dtype=np.float32), np.array(Y, dtype=np.int32)
```

Annexe Q.2 : Évaluation des modèles k-NN et MLP

```
if __name__ == "__main__":

    X_train, Y_train = extraire_vecteurs_id_stricts(DS1_TRAIN, max_par_classe=400)
    X_test, Y_test = extraire_vecteurs_id_stricts(DS2_TEST, max_par_classe=400)

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    del X_train, X_test
    gc.collect()

    # MODÈLE 1 : K-PLUS PROCHES VOISINS (k-NN)
    knn = KNeighborsClassifier(n_neighbors=5, n_jobs=-1)
    knn.fit(X_train_scaled, Y_train)
    Y_pred_knn = knn.predict(X_test_scaled)

    classes_knn = np.unique(np.concatenate((Y_test, Y_pred_knn)))
    print("RÉSULTATS K-NN :")
    print(classification_report(Y_test, Y_pred_knn, labels=classes_knn, digits=3, zero_division=0))

    # MODÈLE 2 : PERCEPTRON MULTICOUCHE (MLP)
    mlp = MLPClassifier(hidden_layer_sizes=(128,), max_iter=300, random_state=42)
    mlp.fit(X_train_scaled, Y_train)
    Y_pred_mlp = mlp.predict(X_test_scaled)

    classes_mlp = np.unique(np.concatenate((Y_test, Y_pred_mlp)))
    print("RÉSULTATS MLP :")
    print(classification_report(Y_test, Y_pred_mlp, labels=classes_mlp, digits=3, zero_division=0))
```

Annexe R.1 : CNN (1/3)

```
import numpy as np, pywt, wfdb, gc, tensorflow as tf
from tensorflow.keras import layers, models, callbacks
from sklearn.metrics import classification_report

# 1. SÉPARATION CLINIQUE INTER-PATIENTS STRICTE
DS1_TRAIN = ['101', '106', '108', '109', '112', '114', '115', '116', '118', '119', '122', '124', '201',
↳ '203', '205', '207', '208', '209', '215', '220', '223', '230']
DS2_TEST = ['100', '103', '105', '111', '113', '117', '121', '123', '200', '202', '210', '212', '213',
↳ '214', '219', '221', '222', '228', '231', '232', '233', '234']

AAMI_MAPPING = {'N':0, 'L':0, 'R':0, 'e':0, 'j':0, 'A':1, 'a':1, 'J':1, 'S':1, 'V':2, 'E':2, 'F':3, '/':4,
↳ 'f':4, 'Q':4}
NOMS_CLASSES = ['N (Normal)', 'S (Supra)', 'V (Ventri)', 'F (Fusion)', 'Q (Inconnu)']

# 2. FILTRAGE AMR
def filtrage_amr_mitbih(signal_brut):
    coeffs = pywt.wavedec(signal_brut, 'db4', level=8)
    coeffs[0] = np.zeros_like(coeffs[0])
    return pywt.waverec(coeffs, 'db4')[:len(signal_brut)]
```

Annexe R.2 : CNN (2/3)

3. EXTRACTION DU DATASET ÉQUILIBRÉ

```
def extraire_dataset_equilibre(liste_patients, max_par_classe=600):
    X, Y = [], []
    echelles = np.arange(1, 33)
    limites = {0: max_par_classe*2, 1: max_par_classe, 2: max_par_classe, 3: max_par_classe, 4:
    ↪ max_par_classe}

    for rec in liste_patients:
        try:
            record, annot = wfdb.rdrecord(rec, pn_dir='mitdb'), wfdb.rdann(rec, 'atr', pn_dir='mitdb')
        except: continue

        signal_propre = filtrage_amr_mitbih(record.p_signal[:, 0])
        compteur = {0: 0, 1: 0, 2: 0, 3: 0, 4: 0}

        for pos, sym in zip(annot.sample, annot.symbol):
            if sym in AAMI_MAPPING and compteur[AAMI_MAPPING[sym]] < limites[AAMI_MAPPING[sym]]:
                c = AAMI_MAPPING[sym]
                if pos - 130 > 0 and pos + 50 < len(signal_propre):
                    coefs, _ = pywt.cwt(signal_propre[pos-130 : pos+50], echelles, 'mexh')
                    X.append(np.log1p(np.abs(coefs)**2).astype(np.float32))
                    Y.append(c)
                    compteur[c] += 1
        gc.collect()

    return np.array(X, dtype=np.float32)[..., np.newaxis], np.array(Y, dtype=np.int32)
```

Annexe R.3 : CNN (3/3)

4. EXÉCUTION PRINCIPALE

```
if __name__ == "__main__":
    X_train, Y_train = extraire_dataset_equilibre(DS1_TRAIN, max_par_classe=600)
    X_test, Y_test = extraire_dataset_equilibre(DS2_TEST, max_par_classe=600)

    m, s = np.mean(X_train), np.std(X_train)
    X_train = (X_train - m) / (s + 1e-7)
    X_test = (X_test - m) / (s + 1e-7)

    modele = models.Sequential([
        layers.Conv2D(32, (3,3), activation='relu', input_shape=(X_train.shape[1], X_train.shape[2], 1),
        ↪ padding='same'),
        layers.MaxPooling2D((2,2)), layers.Dropout(0.2),
        layers.Conv2D(64, (3,3), activation='relu', padding='same'),
        layers.MaxPooling2D((2,2)), layers.Dropout(0.3),
        layers.Flatten(),
        layers.Dense(128, activation='relu'), layers.Dropout(0.5),
        layers.Dense(5, activation='softmax')
    ])

    modele.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    ↪ loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    early_stop = callbacks.EarlyStopping(monitor='val_loss', patience=4, restore_best_weights=True)

    modele.fit(X_train, Y_train, epochs=30, batch_size=64, validation_split=0.15, callbacks=[early_stop])

    Y_pred = np.argmax(modele.predict(X_test), axis=1)
    cls_p = np.unique(np.concatenate((Y_test, Y_pred)))
    print(classification_report(Y_test, Y_pred, labels=cls_p, target_names=[NOMS_CLASSES[i] for i in
    ↪ cls_p], digits=3, zero_division=0))
```

Annexe S : Bilan des incertitudes d'acquisition

L'Arduino Nano numérise le signal analogique de la carte AD8232 via un convertisseur ADC de 10 bits. La plage de tension de 3.3V est divisée en $2^{10} = 1024$ niveaux discrets.

Le quantum (plus petit pas de tension mesurable) vaut :

$$q = \frac{3.3}{1024} \approx 3.22 \text{ mV}$$

En supposant une distribution uniforme de l'erreur d'arrondi, l'incertitude-type liée à la numérisation est :

$$u(V) = \frac{q}{\sqrt{12}} \approx 0.93 \text{ mV}$$

L'exécution de la boucle `loop()` et de la fonction `analogRead()` induit des micro-variations temporelles. Le pas d'échantillonnage Δt n'est pas rigoureusement constant, ce qui crée un "jitter" (bruit de phase).

Annexe T : Rapport k-NN

Classe	Précision	Rappel	F1-Score	Support
N (Normal)	0.897	0.796	0.843	8798
S (Supra)	0.107	0.137	0.120	855
V (Ventri)	0.651	0.820	0.726	2364
F (Fusion)	0.009	0.013	0.011	388
Q (Inconnu)	0.000	0.000	0.000	7
Accuracy			0.730	12412
Moyenne macro	0.333	0.353	0.340	12412
Moyenne pondérée	0.767	0.730	0.744	12412

Annexe U : Rapport MLP

Classe	Précision	Rappel	F1-Score	Support
N (Normal)	0.905	0.716	0.799	8798
S (Supra)	0.128	0.235	0.166	855
V (Ventri)	0.622	0.889	0.732	2364
F (Fusion)	0.010	0.013	0.011	388
Q (Inconnu)	0.000	0.000	0.000	7
Accuracy			0.693	12412
Moyenne macro	0.333	0.370	0.342	12412
Moyenne pondérée	0.769	0.693	0.718	12412

Annexe V : Rapport CNN

Classe	Précision	Rappel	F1-Score	Support
N (Normal)	0.944	0.987	0.965	25598
S (Supra)	0.237	0.036	0.063	1055
V (Ventri)	0.794	0.814	0.804	2764
F (Fusion)	0.026	0.003	0.005	388
Q (Inconnu)	0.000	0.000	0.000	7
Accuracy			0.924	29812
Moyenne macro	0.400	0.368	0.367	29812
Moyenne pondérée	0.893	0.924	0.905	29812

Annexe W : Classification clinique du VO_2 relatif

D'après les recommandations officielles de l'ACSM [7], l'état de santé cardiorespiratoire d'un adulte standard s'évalue à partir de la valeur de son $\text{VO}_{2\text{ max}}$ relatif.

État de santé	Valeur (mL/kg/min)
Sédentaire / Faible	< 35
Sain / Bon	35 – 45
Athlétique / Excellent	> 45

Rappel :

$$\text{VO}_2 \text{ relatif} = \frac{\text{VO}_2 \text{ absolu} \times 1000}{\text{Poids (kg)}}$$

Annexe X : Classification AAMI des arythmies

Classe	Nom	Symboles MIT-BIH
N	Rythme Normal	N, L, R, e, j
S	Arythmie Supraventriculaire	A, a, J, S
V	Arythmie Ventriculaire	V, E
F	Battement de Fusion	F
Q	Battement Inconnu	/, f, Q

Table 1 – Regroupement AAMI des classes d'arythmies.

La norme **AAMI EC57** définit ce regroupement standard pour l'évaluation des algorithmes de détection automatique des arythmies à partir de la base **MIT-BIH Arrhythmia Database**.

1. Cohorte UMA / Málaga

- *Exercise Physiology and Human Performance Lab.*
- Appliquée au calcul de l'aire d'hystérésis à l'effort.

<https://doi.org/10.13026/7ezk-j442>

2. MIT-BIH Arrhythmia Database

- *Beth Israel Hospital & MIT.*
- Appliquée à l'entraînement et la validation du réseau de neurones convolutif.

<https://doi.org/10.13026/C2F305>

Caractéristique de la cohorte	Valeur [Min - Max]
Nombre de tests	992
Participants uniques	857
Âge (années)	[10.8 - 63]
Poids (kg)	[41 - 135]
Taille (cm)	[150 - 230]
Proportion de femmes	15.0 %

Table 2 – Profil physiopathologique des patients lors des tests.

Annexe Z.2 : Profil de la base MIT-BIH Arrhythmia

Caractéristique de la cohorte	Valeur
Sujets uniques	47
Enregistrements ECG	48 (extraits de 30 minutes)
Origine clinique	≈ 60 % hospitalisés, ≈ 40 % externes
Fréquence d'échantillonnage	360 Hz
Résolution	11 bits (plage de 10 mV)
Volume d'annotations expertes	$\approx 110\,000$ battements

Table 3 – Caractéristiques du dataset pour la classification IA

Note : Base équilibrée entre 23 enregistrements aléatoires et 25 enregistrements ciblés sur des arythmies cliniquement significatives.