NUMERO 32376

FILTRAGE DU SIGNAL ELECTROCARDIOGRAMME PAR DECOMPOSITION MODALE EMPIRIQUE DANS LE CADRE D'UN SUIVI MEDICAL A DISTANCE

8e24d3c12f_98112_01-intro-1474.jpg (1820×1137) (futura-sciences.com)
heartbeat-icon-ecg-pathology-trace-vector-27113725.jpg (238×250) (vectorstock.com)

PRESENTATION

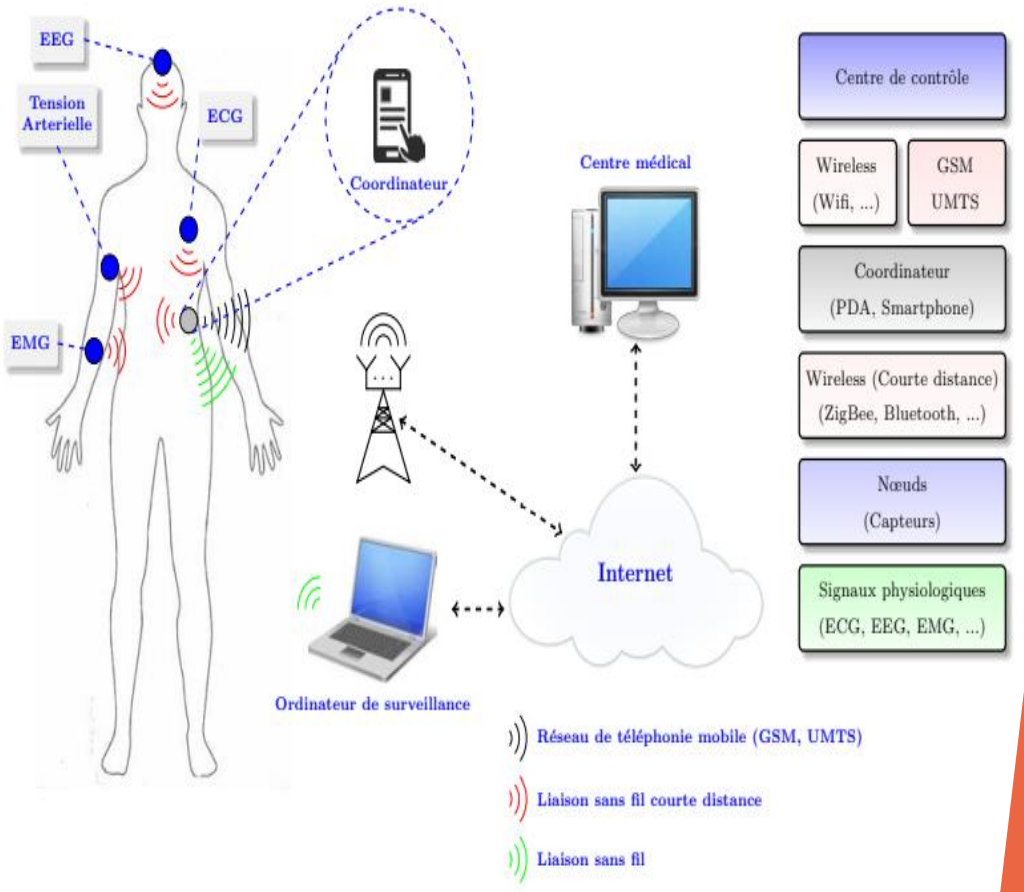
PRESENTATION DU RESEAU DU SUIVI MEDICAL A DISTANCE, “WIRELESS BODY AREA NETWORK” (WBAN)

Fonction globale

- Surveillance en temps direct
- Surveillance en temps différé

Architecture du système

- Le Body Area Network (BAN)
- Le Personal Area Network (PAN)
- Le coordinateur (nœud principal)
- Les réseaux étendus
- Le patient



Source: Figurer 1.1, Approche déterministe de l’acquisition comprimée et la reconstruction des signaux issus de capteurs intelligents distribués, HAL Id: tel-01283555

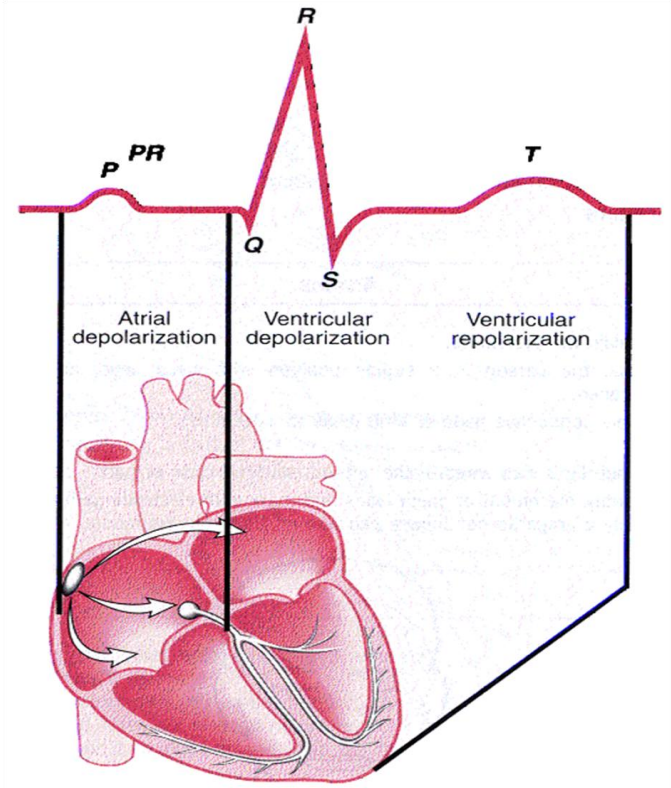
ENJEUX DU SYSTEME

- Mobilité plus importante des patients
- Meilleur diagnostic à distance
- Meilleure intervention des secours
- Meilleure prévention lors des pandémies

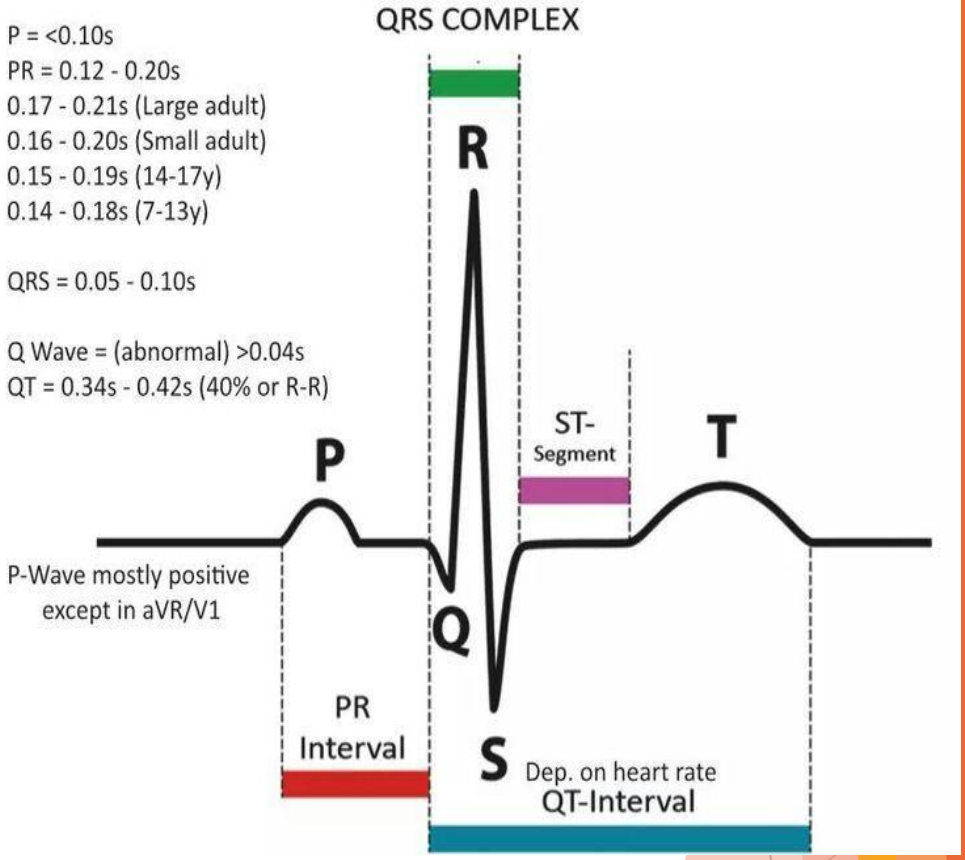
INTERET DU TIPE

- Surveillance à distance du rythme cardiaque des individus atteints d’arythmie cardiaque

ACTIVITE DU CŒUR ET SIGNAL ELECTROCARDIOGRAMME (ECG)



Signal électrique du cœur en fonction de la propagation de l'impulsion électrique



Signal ECG d'un individu sain

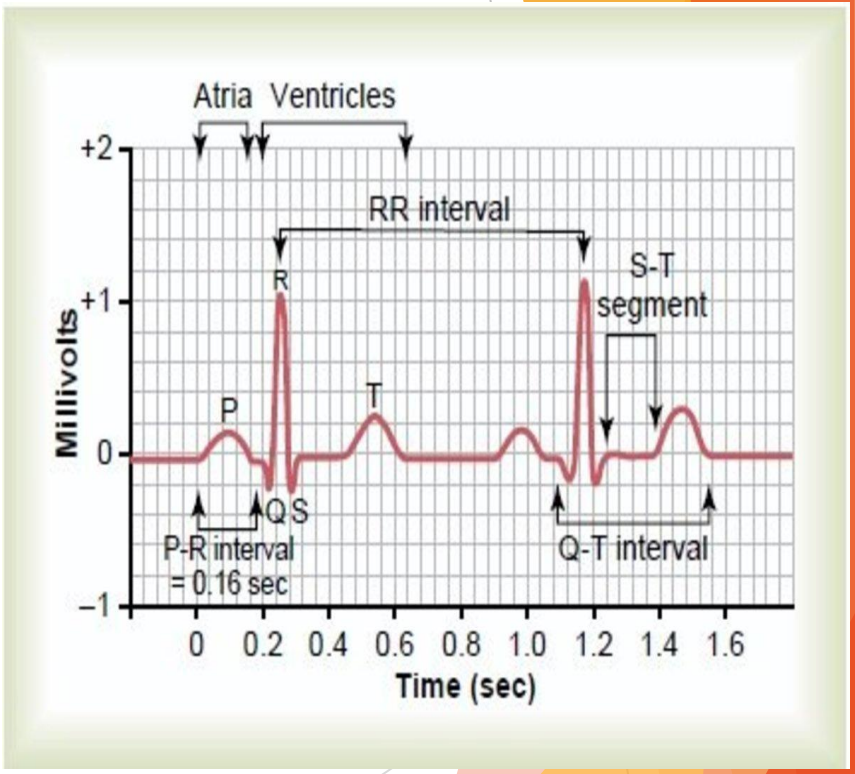
Le signal ECG traduit l'activité mécanique du cœur. Toute anomalie cardiaque entraîne une modification du signal ECG

*Source:*97a8309523ac123e441a40abe9db9555.png (940×952) (pinimg.com)
afc49bfd0b8bd2e8e91712bc7e46817a--qrs-complex.jpg (736×592) (pinimg.com)

ANOMALIE CARDIAQUE : CAS DE L'ARYTHMIE CARDIAQUE

- Arythmie cardiaque : trouble du rythme cardiaque

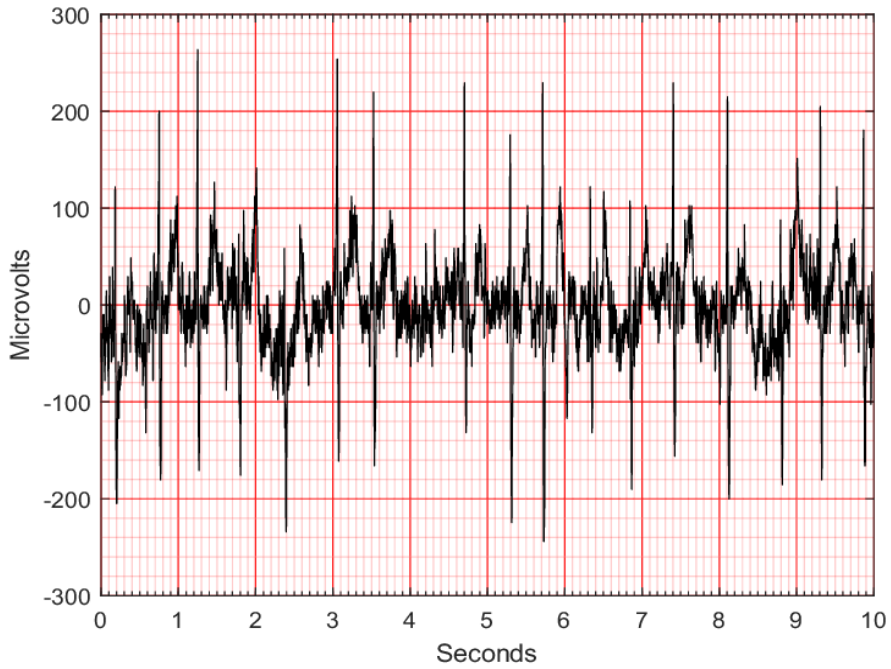
Exemples d'ECG présentant une arythmie cardiaque



ECG d'un individu sain

Source:1002198-Arythmie.jpg (600×235) (larousse.fr)
how-to-read-a-normal-ecgelectrocardiogram.jpg (1200×875) (saymedia-content.com)

NECESSITE D'UN FILTRAGE



Signal en sortie du capteur très bruité → il faut filtrer le signal afin de le rendre exploitable et de permettre une détection plus efficace du complexe QRS

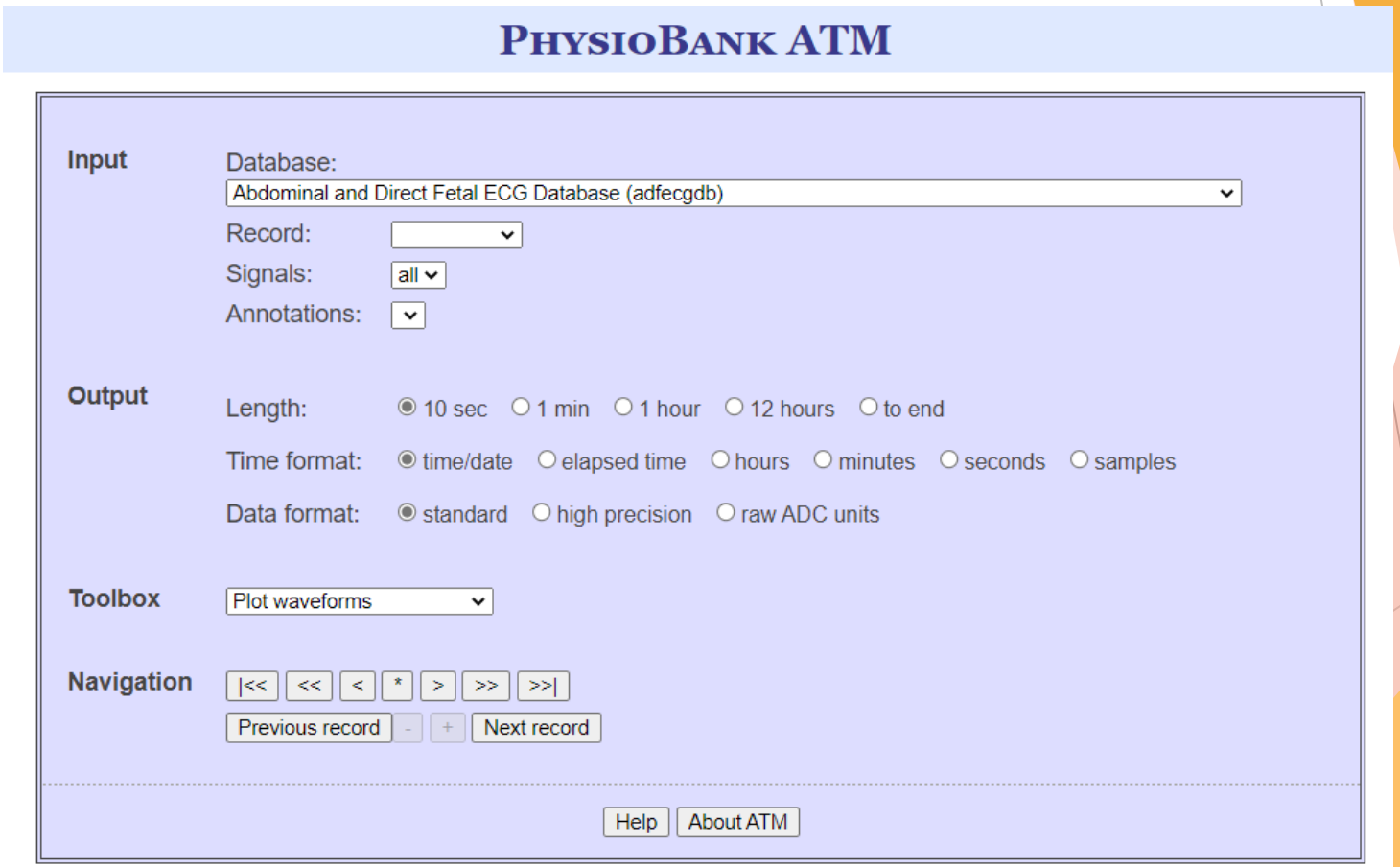
DEUX GRANDS TYPES DE BRUITS AFFECTENT UN SIGNAL ECG :

- **LES BRUITS HAUTE FRÉQUENCE (H-F) (SOURCE D'ALIMENTATION)**
- **LES ONDULATIONS DE LA LIGNE DE BASE (RESPIRATION DU PATIENT) DE FRÉQUENCE F=5 HZ EN GÉNÉRAL**

https://www.researchgate.net/publication/339208642/figure/download/fig1/AS:857749478981632@1581514820025/An-ECG-containing-both-low-and-high-frequency-noise.png

BASE MIT-BIH

- Base de données libre service mise en place par l'université américaine Massachusetts Institute of Technology (MIT)
- Elle contient des enregistrements de signaux ECG d'individus atteints d'arythmie cardiaque numérotés de 100 à 234



PhysioBank ATM (physionet.org)

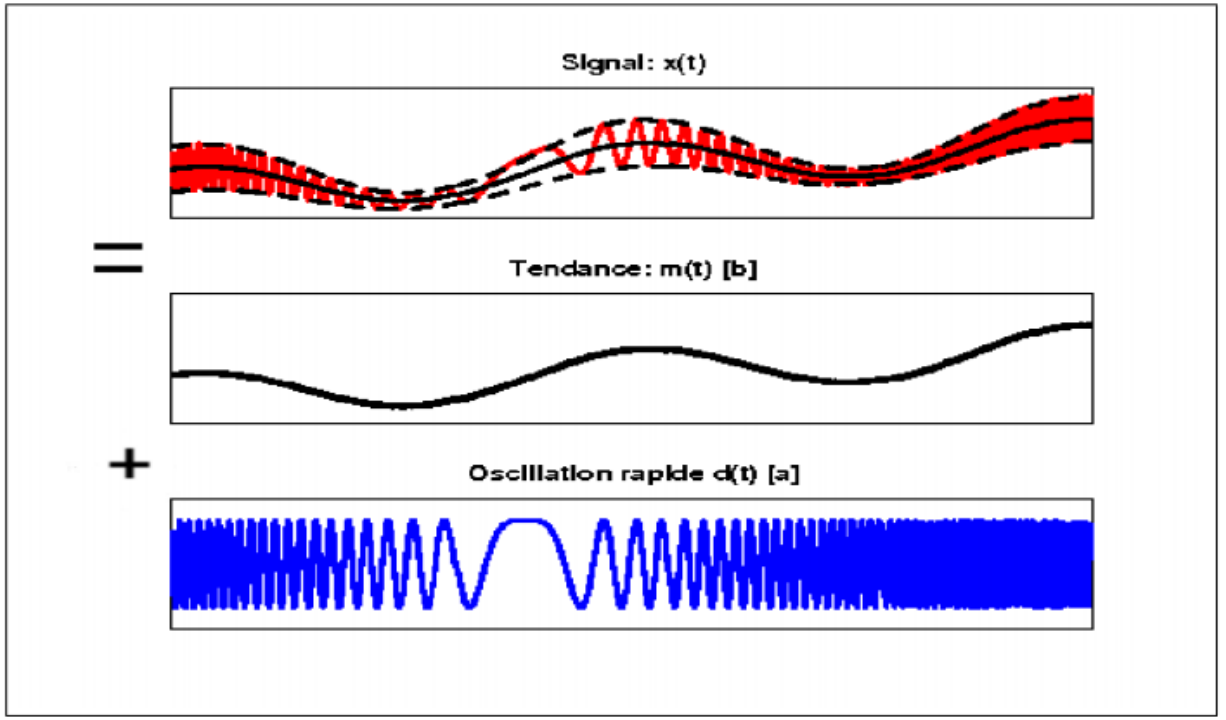
IDEE GENERALE DE LA DECOMPOSITION MODALE EMPIRIQUE (EMD)

Décomposition locale du signal en différentes composantes sinusoïdales haute fréquence et basse fréquence.

$$x(t) = m(t) + d(t)$$

m(t) : tendance locale (basse fréquence)

d(t): détail local (haute fréquence)



Source: Analyse des signaux non-stationnaires par transformation de Huang, Opérateur deTeager-Kaiser, et Transformation de Huang-Teager (THT) , HAL Id: tel-00012141

Plus généralement, on peut décomposer un signal de la façon suivante :

$$x(t) = \sum_{i=1}^{n} s_i(t) + r(t)$$

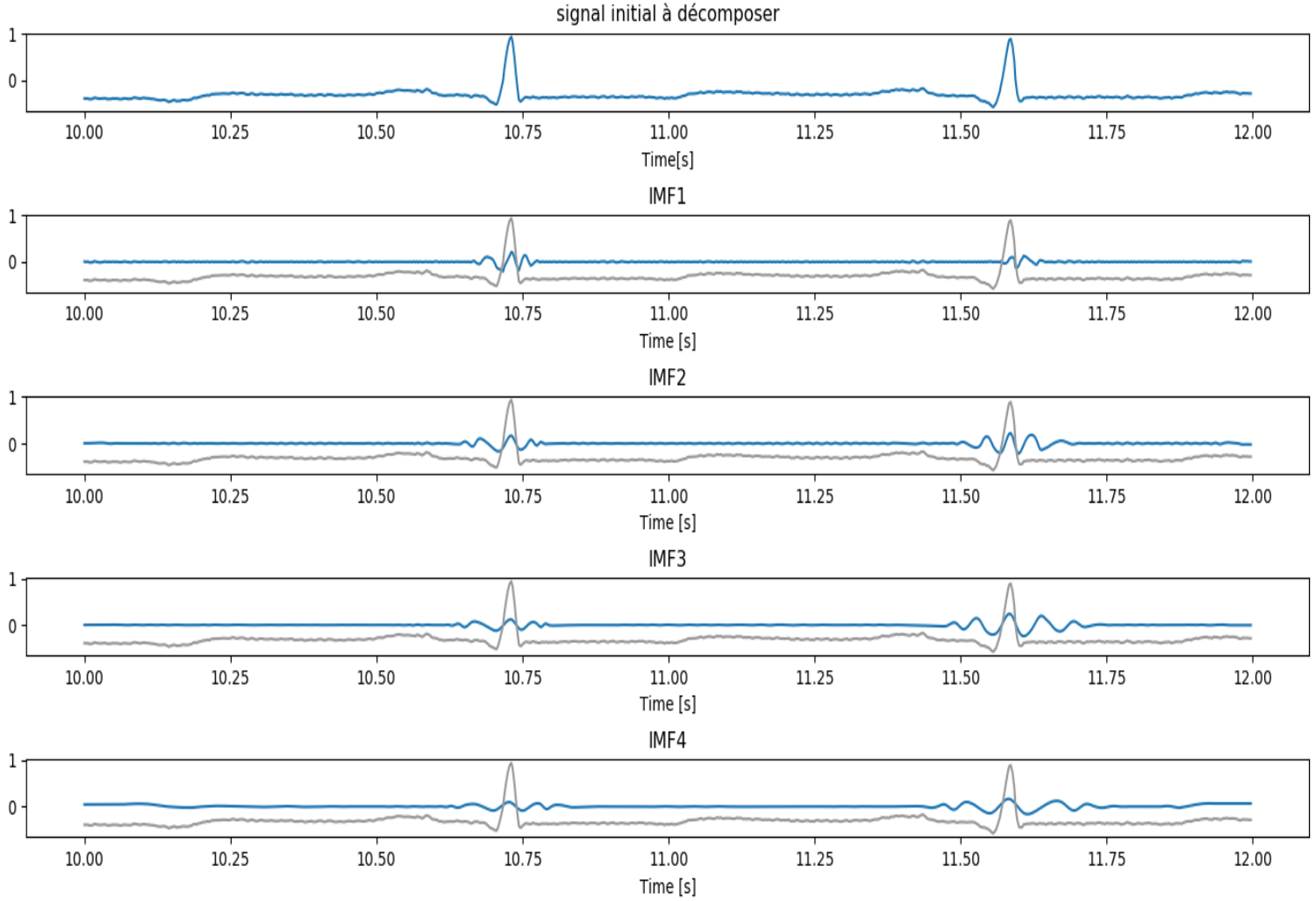
tel que $s_i(t)$ est une fonction oscillante de moyenne nulle et entre deux extrema successifs, la fonction possède un zéro et $r(t)$ est le reste

Les fonctions s_i sont appelées « Intrisic Mode Function »

IMF (Fonction Modale Intrinsèque)

Ces IMF peuvent être déterminées grâce à un algorithme de tamisage

EXEMPLE DE DECOMPOSITION D'UN SIGNAL EN IMF



PROBLEMATIQUE

Dans le souci d'un suivi médical à distance des patients atteints d’arythmie cardiaque, la méthode de filtrage par EMD permet-elle d'obtenir un signal suffisamment pertinent?

?

OBJECTIF DU TIPE :

1- MONTRER L'EFFICACITE DE FILTRAGE H-F ET DE LA BASE DE LIGNE PAR EMD
2-EXPOSER LES AVANTAGES DU FILTRAGE PAR RAPPORT A D'AUTRES FILTRES UTILISES EN ELECTROCARDIOGRAPHIE

Pour atteindre l'objectif 1:

- Ecrire l'algorithme de tamisage
- Ecrire l'algorithme de filtrage H-F
- Evaluer les performances de l'algorithme de filtrage H-F en le comparant à d'autres filtres
- Ecrire l'algorithme du filtrage de la ligne de base
- Evaluer les performances de l'algorithme de filtrage de la ligne de base en le comparant à d'autres filtres

Pour atteindre l'objectif 2:

- Comparaison à la méthode de filtrage par décomposition en ondelettes

Conclure quant aux attentes de ce TIPE

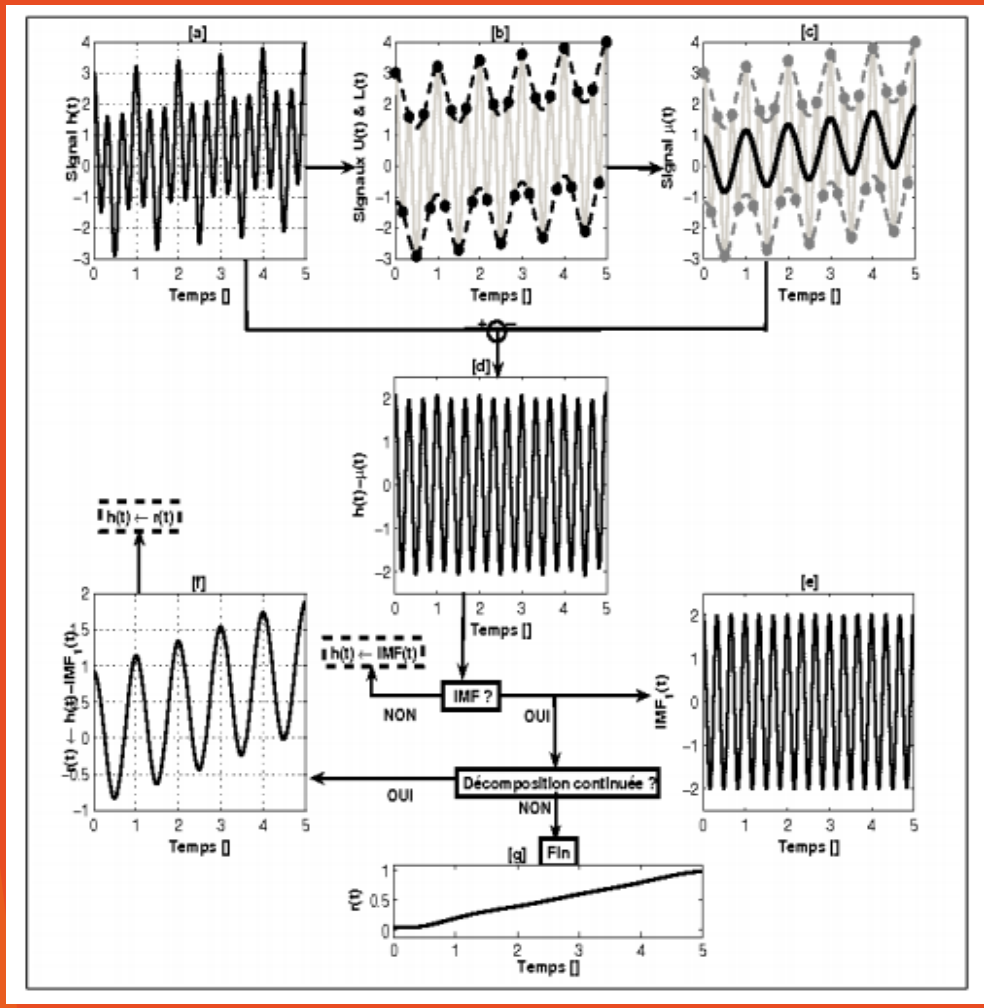
ETAPE 1: ECRIRE L'ALGORITHME DE TAMISAGE

ALGORITHME DE TAMISAGE

IL PERMET DE DÉCOMPOSER LE SIGNAL EN SOMME D'IMF ET D'UN RESTE

ALGORITHME DEFINI DEUX BOUCLES INBRIQUEES :

- *BOUCLE EXTERIEURE:* PERMET DE DECOMPOSER LE SIGNAL EN SOMME D'IMF
- *BOUCLE INTERIEURE:* PERMET DE TROUVER RECURSIVEMENT LES IMF



Explication des etapes de la méthode: c.f. annexe 1

Algorithme en python c.f. annexe 4

Source: Analyse des signaux non-stationnaires par transformation de Huang, Opérateur deTeager-Kaiser, et Transformation de Huang-Teager (THT) , HAL Id: tel-00012141

ETAPE 2: ECRIRE L'ALGORITHME DE FILTRAGE H-F

ALGORITHME DE FILTRAGE H-F

Méthode:

Détermination de l'ordre de bruit

↓

La localisation temporelle des complexes QRS

↓

Fenêtrage du complexe QRS

↓

Définition d'un fenêtrage complémentaire

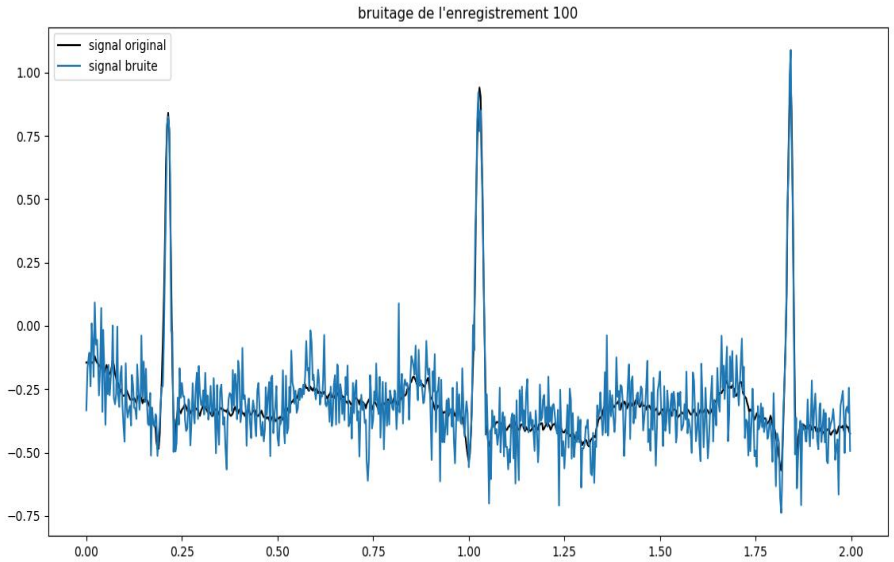
↓

Application du filtrage au signal partiellement filtré

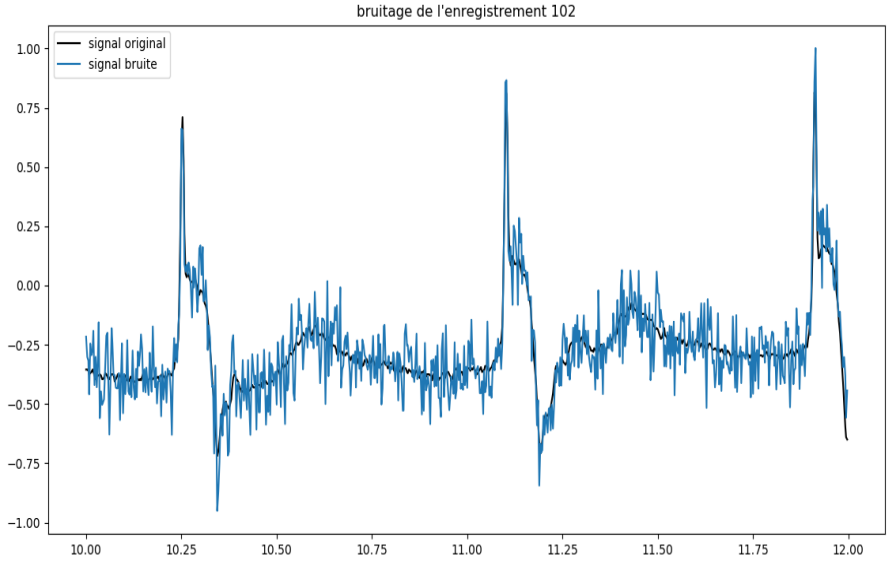
Explication de la méthode c.f.: annexe 6

Algorithme en python c.f.: annexe 8

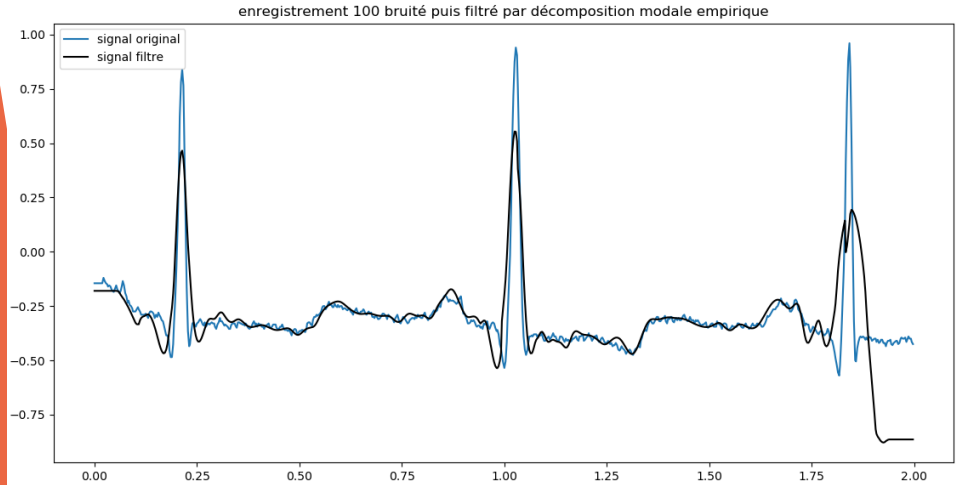
Résultats pour les enregistrements 100 et 102



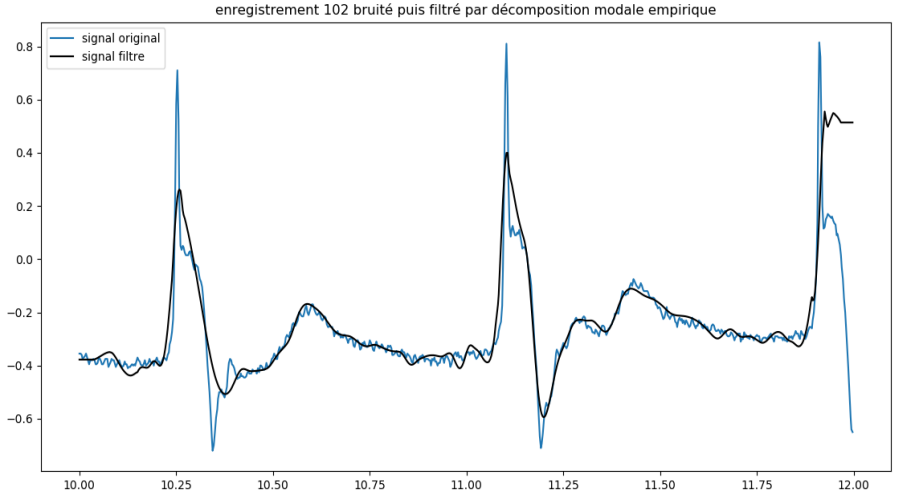
Enregistrement 100 bruité



Enregistrement 102 bruité



Enregistrement 100 bruite puis filtré

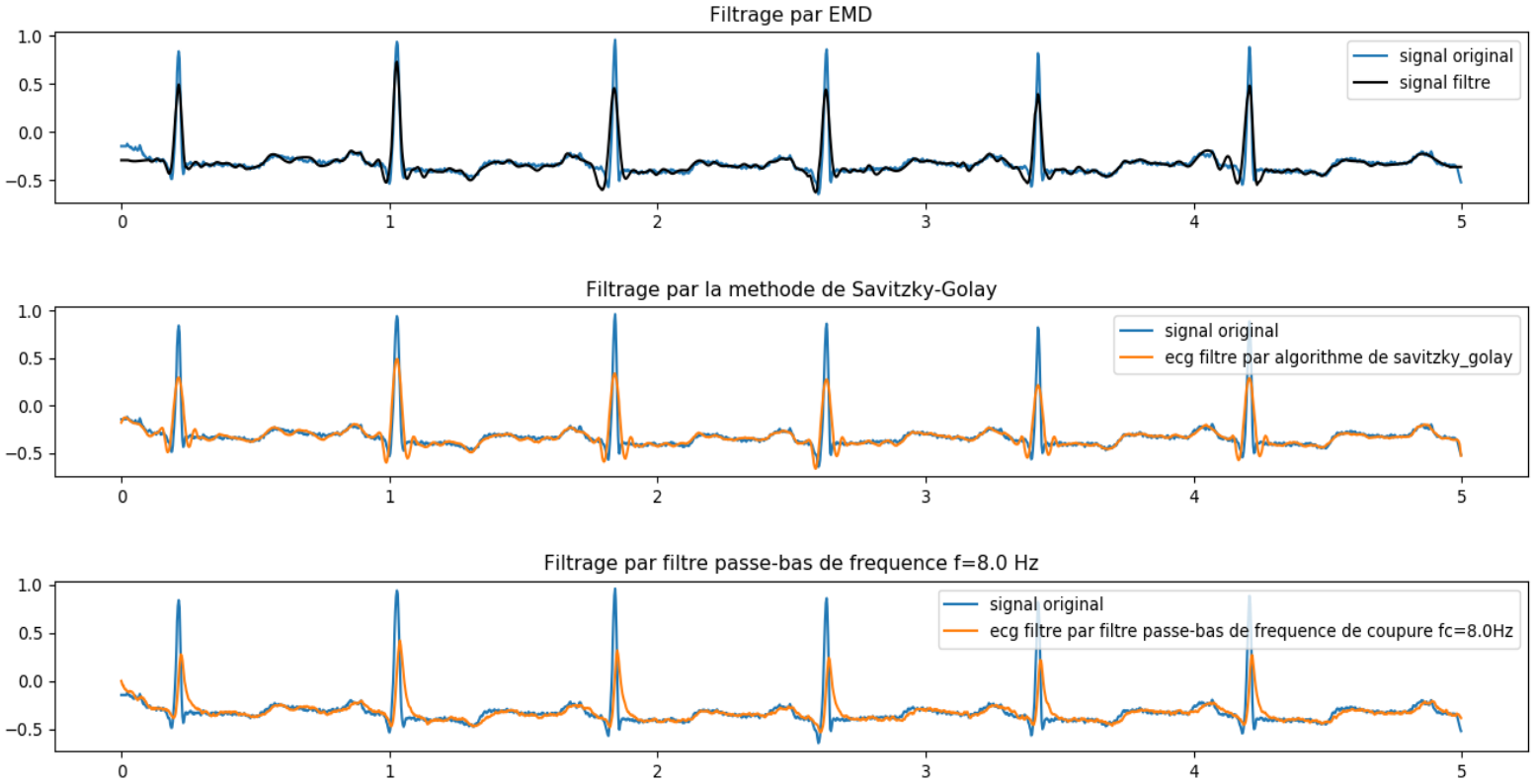


Enregistrement 102 bruite puis filtré

ETAPE 3: COMPARAISON DE L'ALGORITHME DE FILTRAGE H-F PAR EMD AU FILTRAGE DE SAVITZKY-GOLAY ET AU FILTRAGE PASSE-BAS

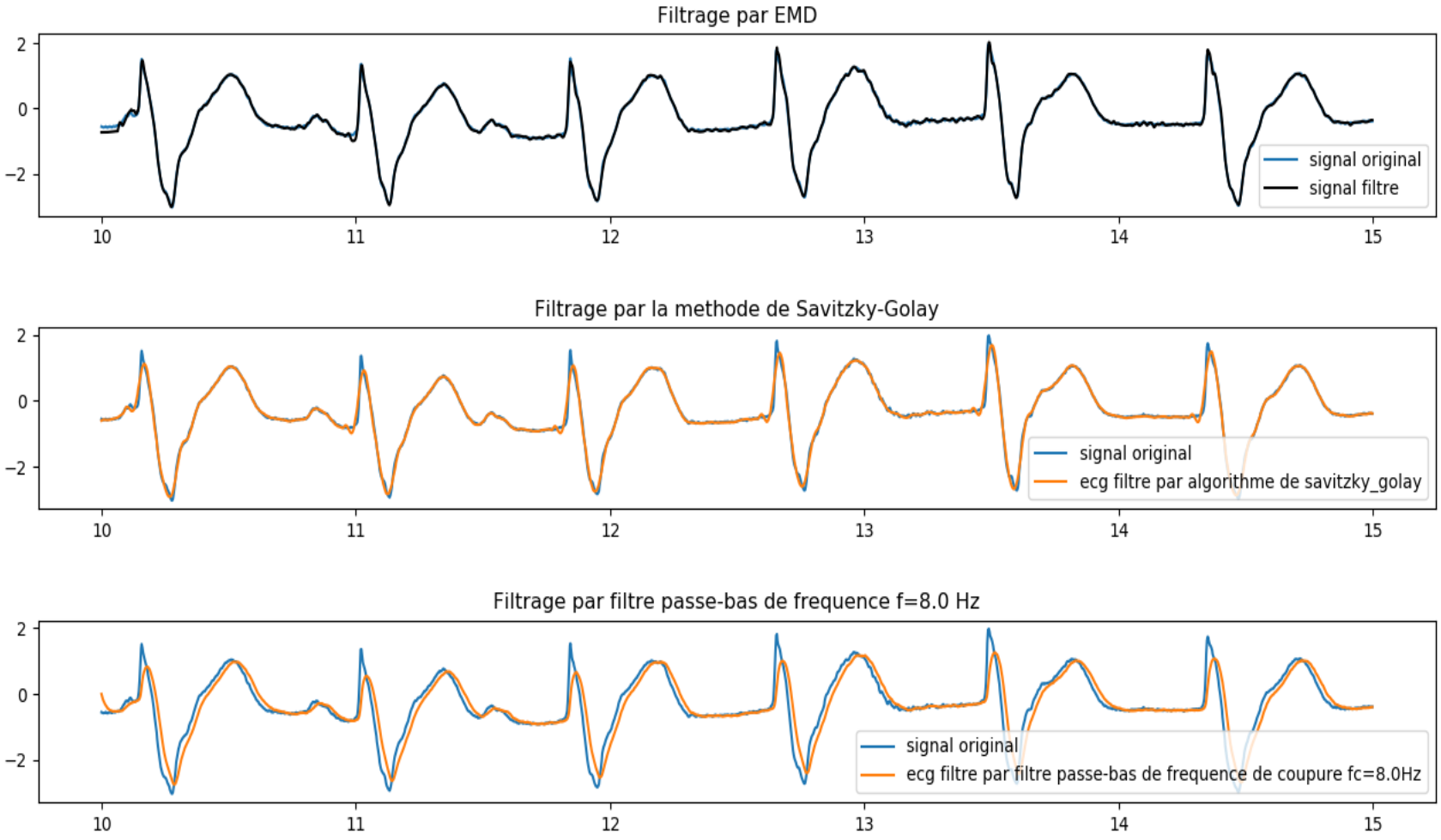
Explication filtrage Savitzky-Golay *c.f. annexe 9*
Explication filtrage passe-bas *c.f. annexe 10*

EXEMPLE DE FILTRAGE DE L'ENREGISTREMENT 100 BRUITE PAR DES BRUITS H-F PAR TROIS MÉTHODES DIFFERENTES



Algorithme python *c.f. annexe 11*

EXEMPLE DE FILTRAGE DE L'ENREGISTREMENT 107 BRUITE PAR DES BRUITS H-F PAR TROIS MÉTHODES DIFFERENTES



COMPARAISON DES ERREURS ENTRE LE SIGNAL ORIGINAL ET LE SIGNAL FILTRE

ERREUR QUADRATIQUE MOYENNE (MSE)

$$MSE(s, s_{filtré}) = \frac{1}{N}\sum_{i=1}^{N}(s(t_i) - s_{filtré}(t_i))^2$$

EVALUATION DU MSE POUR CHAQUE METHODE DE FILTRAGE

	100	*101*	*102*	*103*	*104*	*105*	*107*	*108*	*109*
EMD	1,11%	1,07%	0,44%	2,12%	0,74%	0,64%	0,15%	0,32%	1,13%
SG	0,77%	0,78%	0,37%	1,76%	0,49%	0,15%	1,01%	0,08%	0,10%
PB	1,80%	2,16%	1,14%	6,02%	1,65%	2,99%	14,1%	0,58%	3,58%

On observe que le filtrage par EMD présente le MSE assez faible en général

ETAPE 4: ECRIRE L'ALGORITHME DU FILTRAGE DE LA LIGNE DE BASE

ALGORITHME DE FILTRAGE LIGNE DE BASE

Méthode:

Décomposition du signal par EMD

↓

Détection des complexes QRS

↓

Détection des points d'inactivité du cœur

↓

Détermination de la ligne de base par interpolation des points d'inactivité

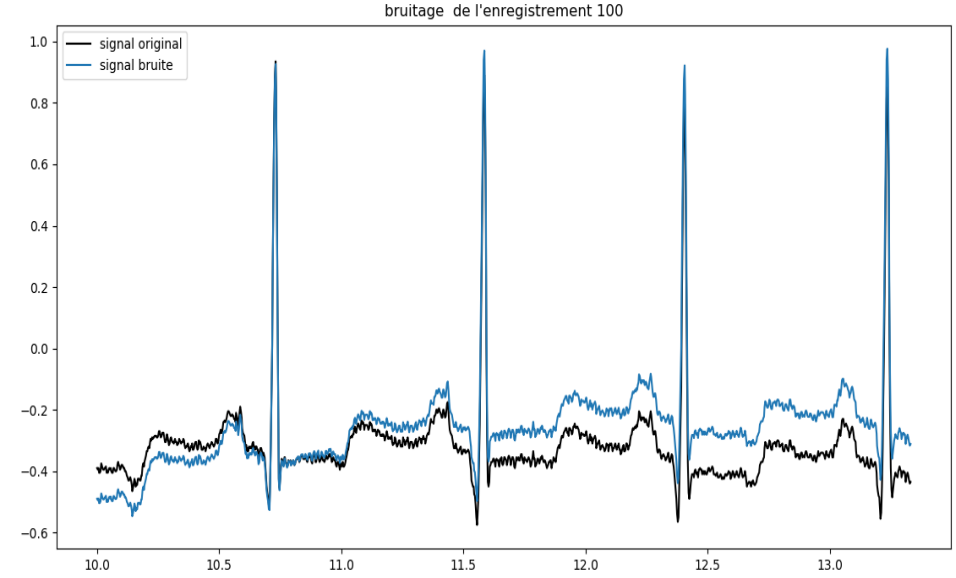
↓

Elimination de la ligne de base du signal ECG initial

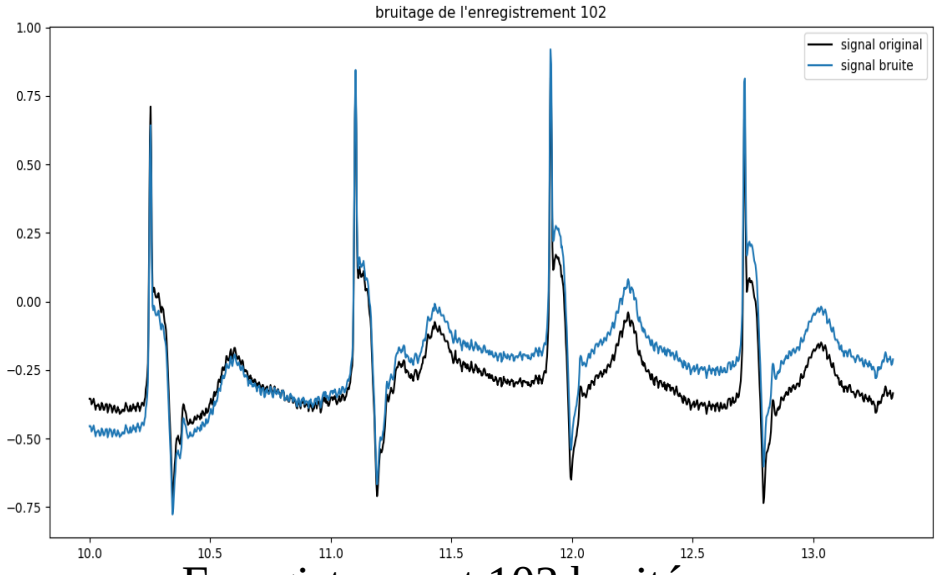
Explication de la méthode c.f.: annexe 12

Algorithme en python c.f.: annexe 13

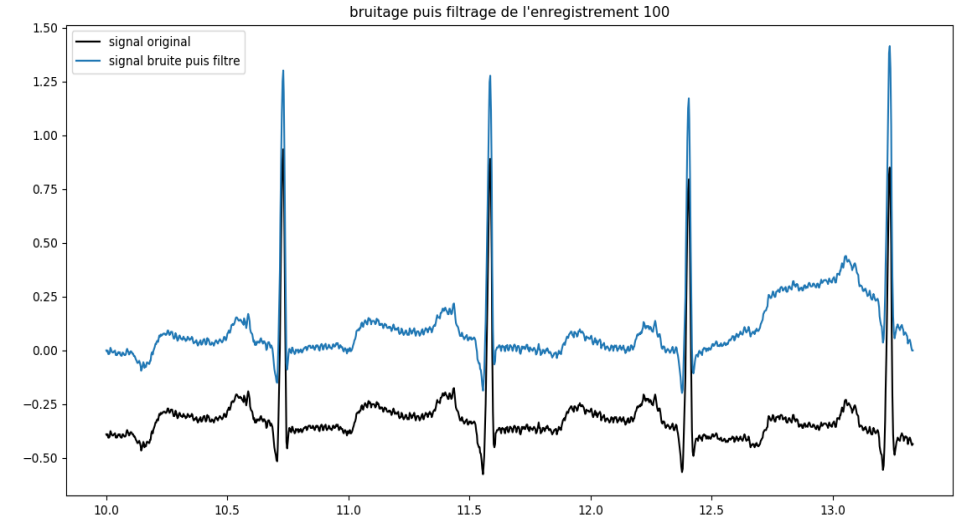
Résultats pour les enregistrements 100 et 102



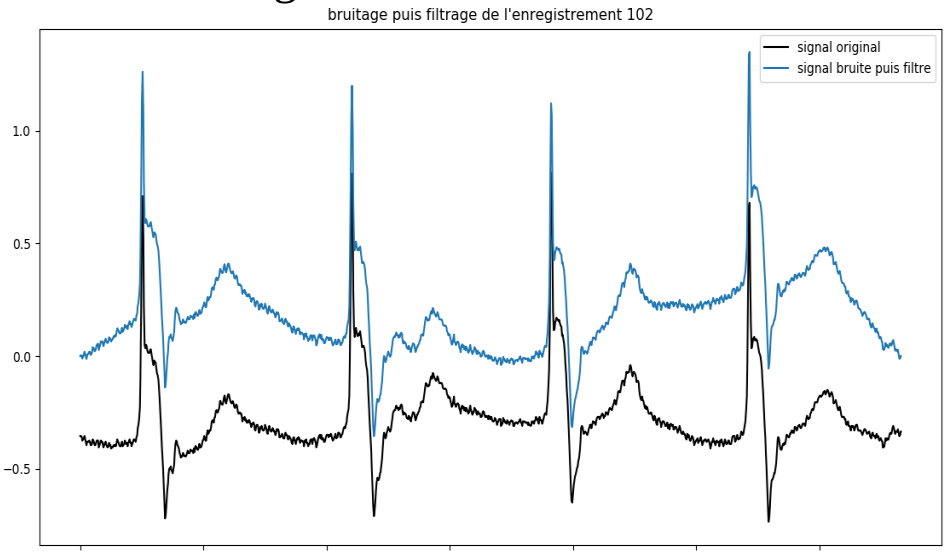
Enregistrement 100 bruité



Enregistrement 102 bruité



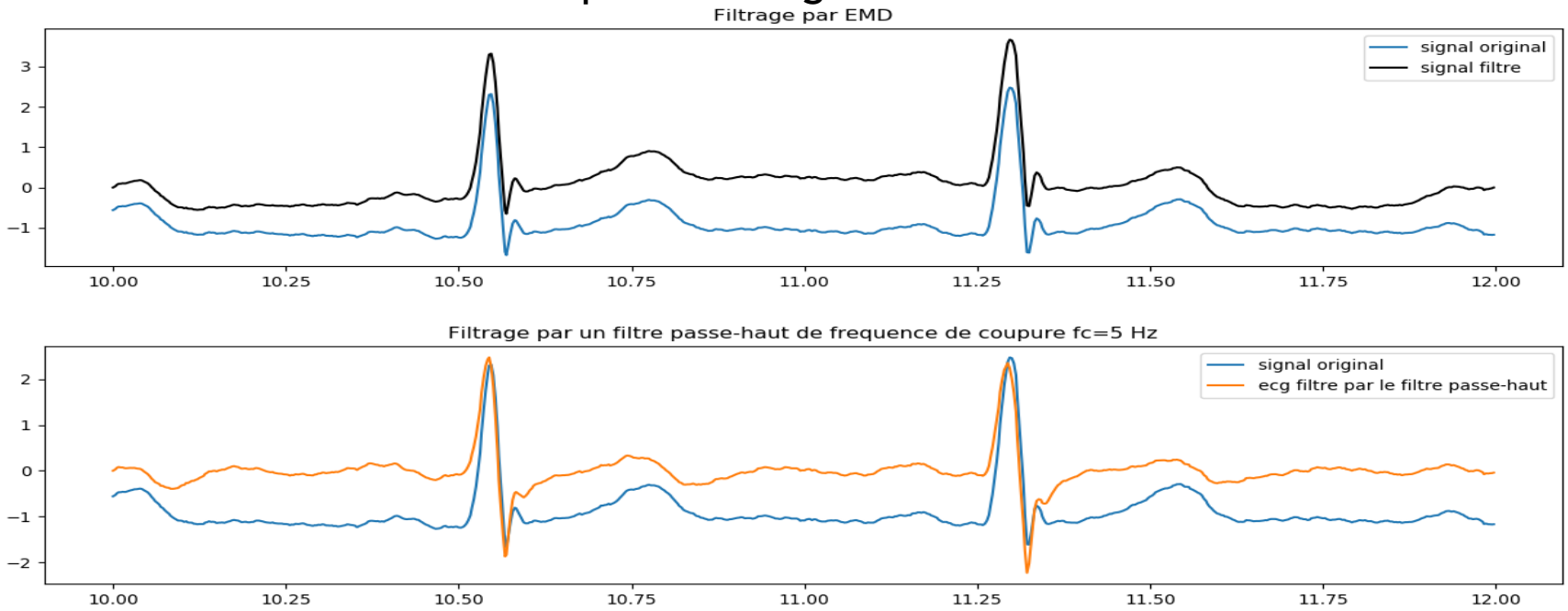
Enregistrement 100 bruite puis filtré



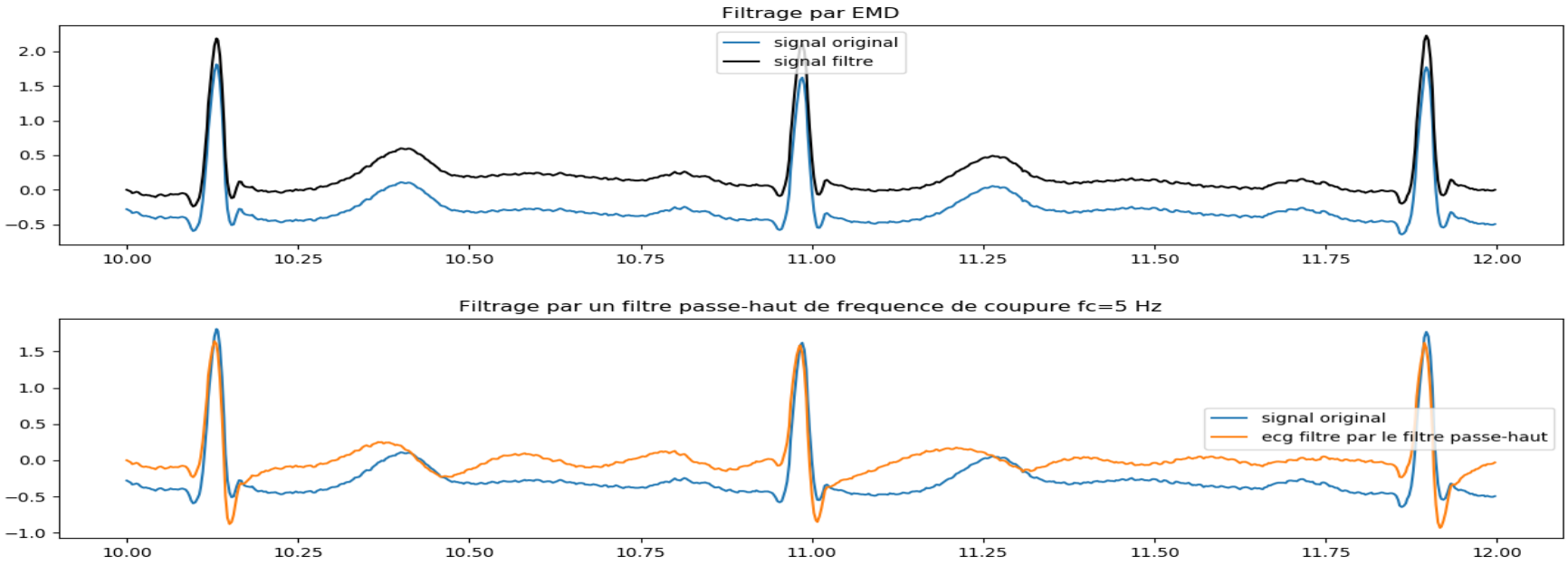
Enregistrement 102 bruite puis filtré

ETAPE 5: COMPARAISON DE L'ALGORITHME DE FILTRAGE DE LA LIGNE DE BASE PAR EMD AU FILTRAGE PASSE-HAUT

Résultat pour l'enregistrement 116



Résultat pour l'enregistrement 103



COMPARAISON DES ERREURS ENTRE LE SIGNAL ORIGINAL ET LE SIGNAL FILTRE

On évalue à nouveau le MSE :

- Dans le cas d'un filtre passe-haut de fréquence de coupure fc=5Hz
- Dans le cas du filtrage par EMD

	100	***101***	***103***	***104***	***108***	***114***	***115***	***116***
EMD	0,06%	0,11%	0,28%	1,52%	2,63%	6,76%	3,95%	8,53%
PH	1,24%	4,32%	8,25%	3,70%	2,65%	1,05%	5,53%	15,85%

Explication filtrage passe-haut *c.f. annexe 14*

ETAPE 6: COMPARAISON AVEC LE FILTRAGE PAR DECOMPOSITION EN ONDELETTES

FILTRAGE PAR TRANSFORMATION EN ONDELETTES

Transformation en ondelettes = définir une famille d'ondelettes à partir d'une sinusoïde de moyenne nulle qui est dilatée et translatée selon la relation suivante :

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)$$

a = facteur d'échelle

b = paramètre de translation

Avantages = elle permet d'effectuer plusieurs étapes du traitement du signal conjointement (filtrage, détection, …)

Inconvénient majeur = elle nécessite la définition de la fonction qu'il faut choisir de façon suffisamment pertinente afin d'obtenir une analyse cohérente.

C'est pourquoi en 1992, Huang et al. a développé la méthode de décomposition modale empirique

CONCLUSION

AVANTAGES ET INCONVENIENTS DE L'ALGORITHME DE TAMISAGE

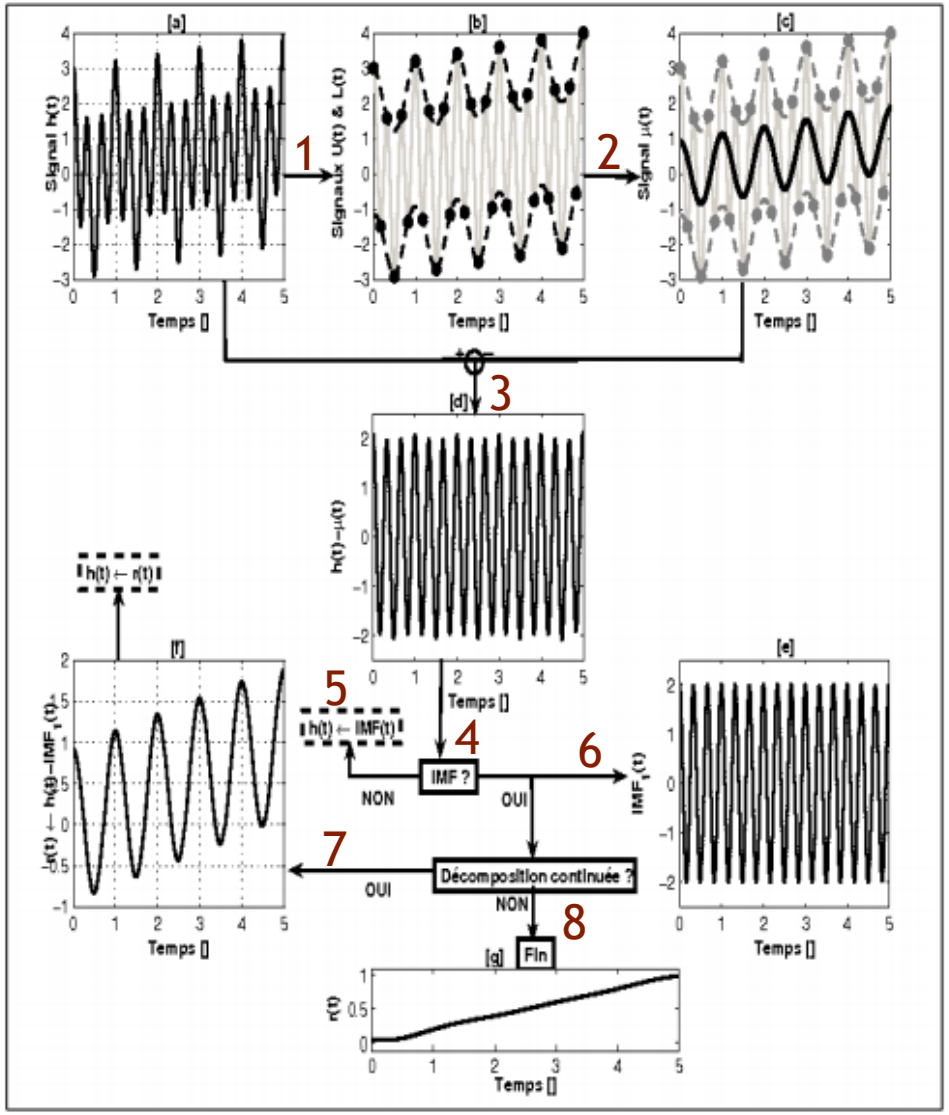
1. Avantages :
   - Ne nécessite pas de fonctions préalablement définies ce qui assure une certaine fiabilité de la décomposition à tout signal
   - Permet une détection efficace des complexes QRS ce qui peut être utile pour le traitement du signal
2. Inconvénients :
   - Nombreux degrés de liberté doivent être défini par l'utilisateur : choix de l'interpolation, critère d'arrêt, choix du fenêtrage,…

REPONSE A LA PROBLEMATIQUE :

Oui, c'est un algorithme qui est à privilégier dans le cadre d'un suivi médical à distance sous réserve de définir correctement les paramètres qui doivent être fixés par l'utilisateur

ANNEXES

ANNEXE 1: ALGORITHME DE TAMISAGE



On considère un signal d'origine à partir duquel on déduit la j-ième IMF

1) On extrait les extrema et on définit des enveloppes par interpolation (*spline cubique cf annexe 2*)

2) Calcul de la moyenne des deux enveloppes

3) Définition du signal issu de la difference du signal d'origine et de la moyenne

4) On verifie le critere d'arret (*cf annexe 3*)

5) Si le signal ne le verifie, on retourne à l'étape 1avec le signal obtenu

6) Si le citere est verifie alors on a une IMF

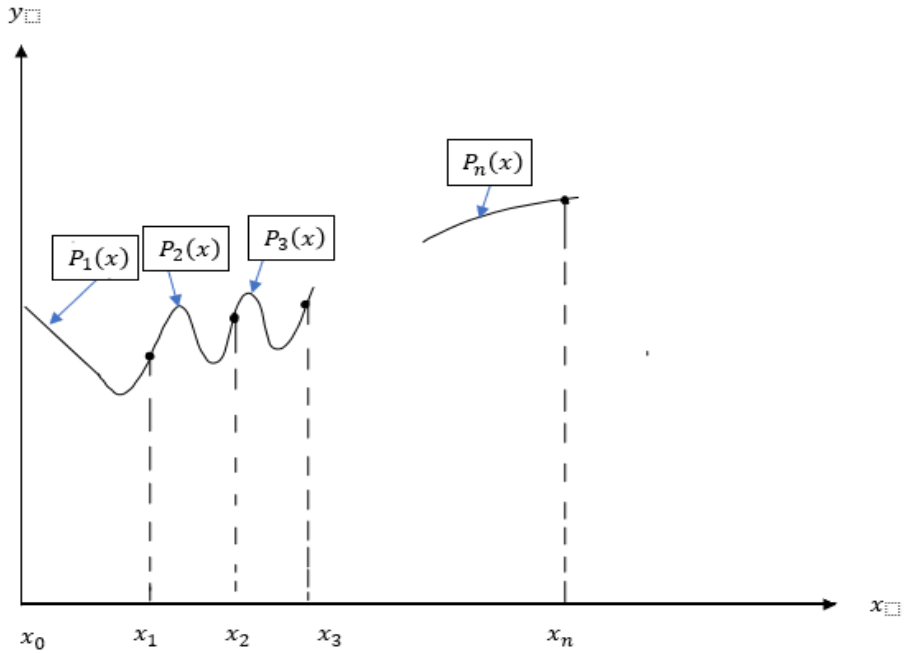
7) On determine la j+1 ieme IMF en prenant comme signal initial le reste du signal d'origine par la j-ieme IMF et on retourne à l'étape 1

8) On continue jusqu'à avoir un reste constant

ANNEXE 2: INTERPOLATION PAR SPLINE CUBIQUE

Pour chaque intervalle $[x_{i-1}, x_i]$, on peut définir une fonction polynômiale P_i telle que :

$$P_i(x) = a_i(x - x_{i-1})^3 + b_i(x - x_{i-1})^2 + c_i(x - x_{i-1}) + d_i$$



Propriétés que doivent valider les fonctions polynômiales sur chaque intervalle afin de vérifier l'interpolation :

- $P_1(x_0) = \mathrm{f}(x_0)$
- $P_n(x_n) = \mathrm{f}(x_n)$
- $pour\ i = 1, \dots, (n-1)\ P_i(x_i) = \mathrm{f}(x_i)$
- $pour\ i = 1, \dots, (n-1)\ P_{i+1}(x_i) = \mathrm{f}(x_i)$
- $pour\ i = 1, \dots, (n-1) P'_{i+1}(x_i) = P_i'(x_i)$
- $pour\ i = 1, \dots, (n-1) P''_{i+1}(x_i) = P_i''(x_i)$

ANNEXE 3: CRITERE D'ARRET PERMETTANT DE DETERMINER L'IMF

$$SD(i) = \sum_{t=0}^{T} \frac{|h_{j,i-1}(t) - h_{j,i}(t)|^2}{(h_{j,i-1}(t))^2}$$

Où i correspond au nombre de passage dans la boucle de tamisage et T le nombre d'échantillons du signal

ANNEXE 4

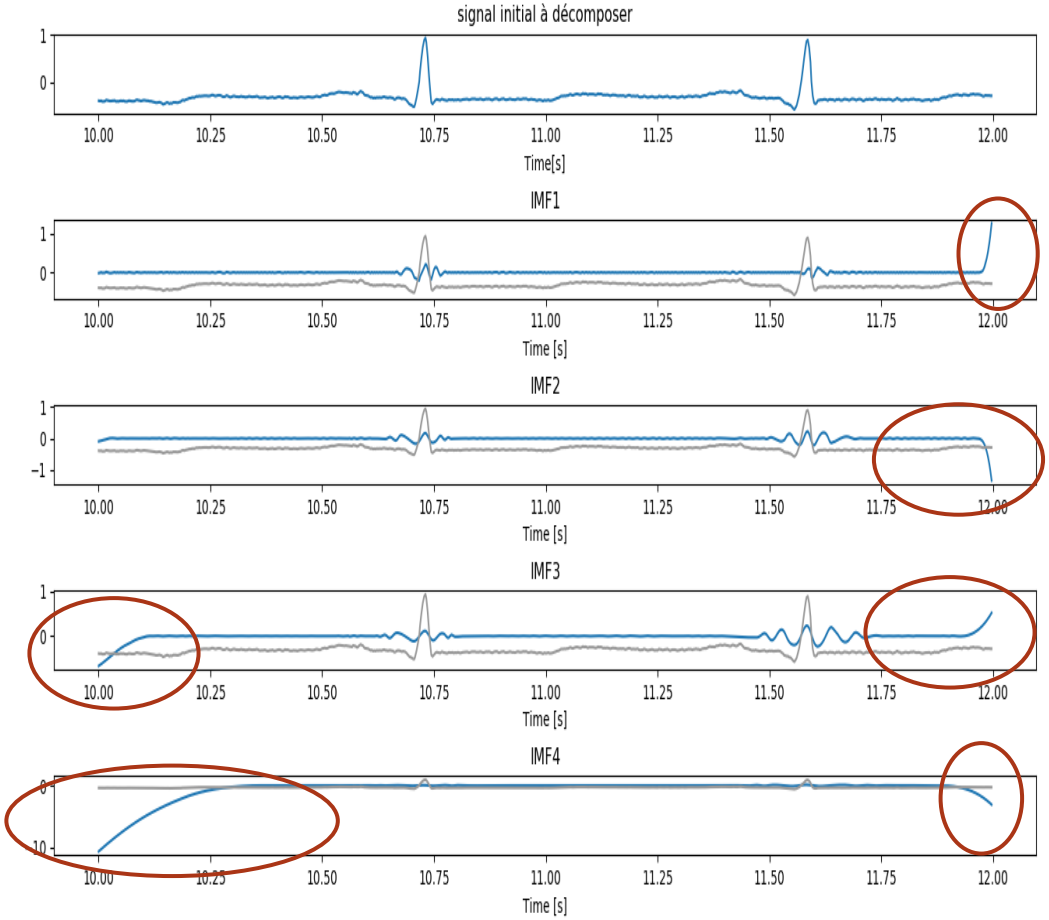
ALGORITHME DE TAMISAGE

```
def detection_extrema(s,t):
    # on determine les indices des pics
    i_t_maxi, _ = find_peaks(s)
    i_t_mini, _ = find_peaks(-s)

    #print(np.max(i_t_mini))
    #print(np.max(i_t_maxi))
    #print(sublist_end)

    #on determine la date d'apparition des pics
    t_maxi = t[i_t_maxi]
    t_mini = t[i_t_mini]
    #on determine la valeur de la fonction a ces dates
    maxi = s[i_t_maxi]
    mini = s[i_t_mini]


    return maxi,t_maxi,mini,t_mini,i_t_maxi,i_t_mini
```

```
def nombre_extrema(s):
    _, t_maxi, _, t_mini, _, _ =detection_extrema(s,t)
    return np.size(t_maxi)+np.size(t_mini)-6

def spline_cubique(t,tl,yl):
    tck = interpolate.splrep(tl, yl,k=3)
    return interpolate.splev(t, tck)

def spline_lineaire(t,tl,yl):
    tck = interpolate.splrep(tl, yl,k=1)
    return interpolate.splev(t, tck)

def enveloppe_sup(t,y):
    y_maxi, t_maxi, _, _, _, _ = detection_extrema(y,t)
    #t_maxi=detection_extrema(y,t)[1]
    return spline_cubique(t,t_maxi,y_maxi)

def enveloppe_inf(t,y):
    _, _, y_mini, t_mini, _, _ = detection_extrema(y,t)
    #t_mini=detection_extrema(y,t)[3]
    return spline_cubique(t,t_mini,y_mini)

def critere_arret(h_j_i, h_j_i_1,epsilon):
    n=np.size(h_j_i)
    SD_i=0
    for i in range(n):
        if h_j_i_1[i]== 0:
            SD_i+=0
        else:
            SD_i+=(abs(h_j_i_1[i]-h_j_i[i])**2)/((h_j_i_1[i])**2)
    return SD_i<epsilon

def sublist_mean(mean, i_t_maxi, i_t_mini):
    sublist_start = np.max((np.min(i_t_maxi),np.min(i_t_mini)))
    sublist_end = np.min((np.max(i_t_maxi),np.max(i_t_mini))) + 1
    mean[:sublist_start] = mean[sublist_start]
    mean[sublist_end:] = mean[sublist_end]
    return mean
```

```
'''on suppose le signal deja lu et s correspond a la lite des echanitllons du signal
i.e. amplitude en fonction du temps'''
def tamissage(t,s, epsilon):
    # definition des parametres
    IMF=[]
    s=np.array(s)
    #on commence pour j=1
    #r_(j-1) correspond au (j-1)_ieme residu. pour j=1, le residu est le signal entier
    r_j_1=s
    h=[]
    r_j=[]
    '''ici, il s'agit du critere d'arret de la boucle exterieure.
    Elle permet de finir l'extraction des IMF du signal'''
    while nombre_extrema(r_j_1)>=2:
        #extraction de la j-ieme IMF
        '''On souhaite ici extraire la i-ieme IMF. Sachant que la i-ieme IMF est obtenue de maniere iterative et doit remplir certains criteres,
        on doit a nouveau definir un critere d'arret permettant de terminer l'extraction de la i-ieme IMF'''
        #initialisation
        #on commence pour i=1
        h_j_i_1=r_j_1
        x=True
        while x==True:
            maxi_h_j_i_1, t_maxi_h_j_i_1, mini_h_j_i_1, t_mini_h_j_i_1, i_t_maxi,i_t_mini = detection_extrema(h_j_i_1,t)
            if len(maxi_h_j_i_1)<4 or len(mini_h_j_i_1)<4:
                break
            #on construit ensuite l'enveloppe superieure et l'enveloppe inferieure
            e_sup=np.array(enveloppe_sup(t,h_j_i_1))
            e_inf=np.array(enveloppe_inf(t,h_j_i_1))
            #on le tableau moy qui contient la moyenne des enveloppes et qui est local
            moy_j_i_1=(e_sup+e_inf)/2
            moy_j_i_1 = sublist_mean(moy_j_i_1, i_t_maxi, i_t_mini)
            #on soustrait au signal la moyenne du signal
            h_j_i=h_j_i_1-moy_j_i_1

          if critere_arret(h_j_i, h_j_i_1,epsilon)==True:
              x=False
              break
          else:
              #on affecte a la variable h_j_(i_1) la valeur h_j_i afin de retourner dans la boucle avec cette fois-ci h_j_i
              #c'est equivalent a dire que i prend la valeur i+1
              h_j_i_1=h_j_i
      #on a trouve, la j-ieme IMF et on l'ajoute dans le tableau des IMF
      IMF.append(list(h_j_i))

      if [h_j_i[i]>10.**(-8) for i in range(np.size(h_j_i))]==[False]*np.size(h_j_i):
              r_j_1=r_j_1-h_j_i
              IMF.append(list(r_j_1))
              return IMF
      #de la meme facon j prend la valeur j+1 et r_(j_1) prend la valeur r_(j_1)-h_j_i
      r_j_1=r_j_1-h_j_i
      h_j_i_1=r_j_1
      r_j.append(list(r_j_1))
  IMF.append(list(h_j_i_1))
  return IMF
```

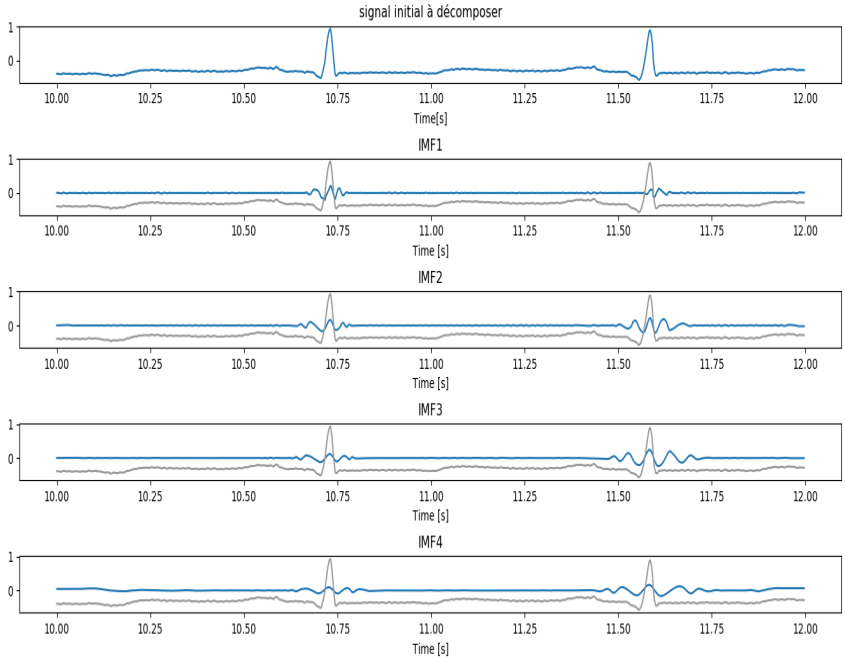
ANNEXE 5: CORRECTION DE LA DIVERGENCE LIEE AU SPLINE CUBIQUE



Signal avant correction

```python
def sublist_mean(mean, i_t_maxi, i_t_mini):
    sublist_start = np.max((np.min(i_t_maxi),np.min(i_t_mini)))
    sublist_end = np.min((np.max(i_t_maxi),np.max(i_t_mini))) + 1
    mean[:sublist_start] = mean[sublist_start]
    mean[sublist_end:] = mean[sublist_end]
    return mean
```

Code python



Signal après correction

ANNEXE 6 EXPLICATION DE LA METHODE

- Détermination de l'ordre du bruit :

  Ordre du bruit = nombre d'IMF modifié pour les bruits.
  On réalise un test sur la moyenne des IMF : si la composante n'est pas de moyenne nulle alors elle est bruitée.

$$O_{bruit} = \min(N, 5)$$

- Détermination du complexe QRS

  Cf annexe 7

- Fenêtrage du complexe QRS

  Fenêtre de Tukey Ψ :

  - Elle doit être plate

  -Elle vaut 1 sur un intervalle τ_1

  - Elle doit être doit converger doucement vers 0
  afin de déformer au minimum le signal sur l'intervalle τ_2

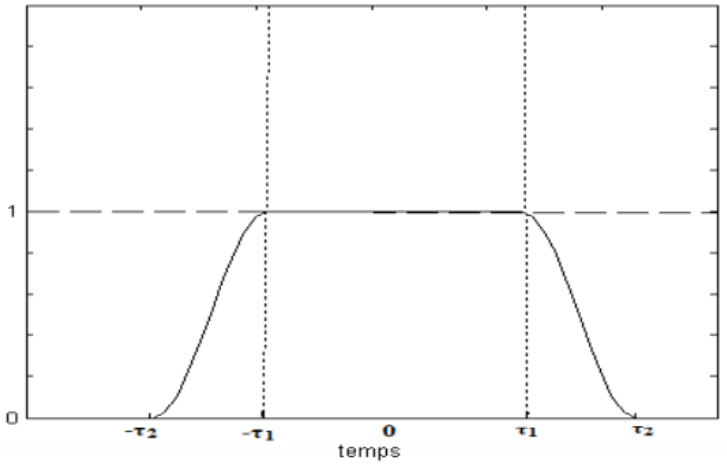


Figure IV.22 : La fenêtre de Tukey.

- Fenêtrage complémentaire : $\overline{\Psi}(t) = \alpha(1 - \Psi(t))$ où $0.1 < \alpha < 0.3$

ANNEXE 7 DETCTION DES COMPLEXES QRS

- Sommation des trois premiers IMFs

$$\widetilde{IMF}(t) = \sum_{i=1}^{3} IMF_i(t)$$

- La localisation temporelle les pics de la fonction :

$$\overline{IMF}(t) = \left(\widetilde{IMF}(t)\right)^2$$

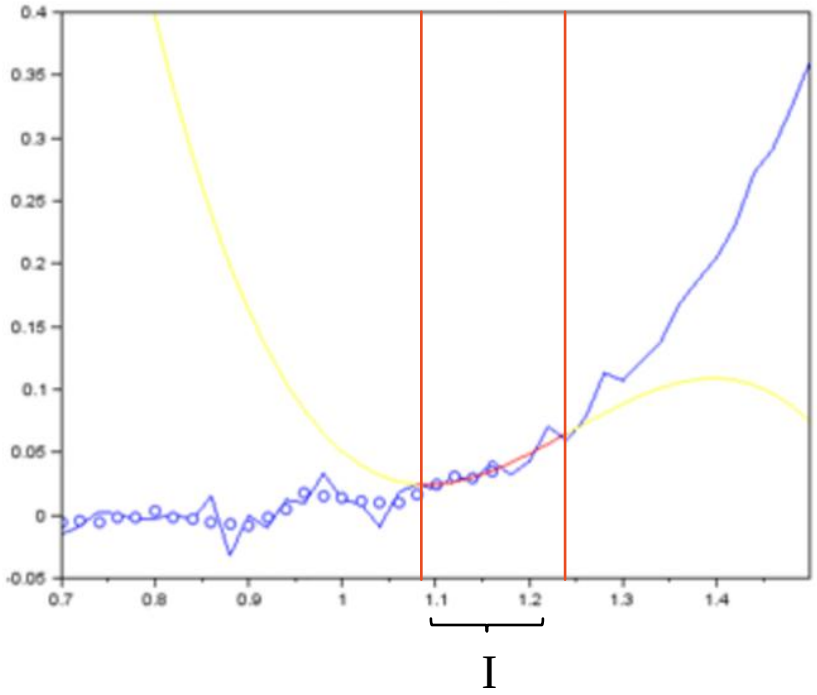
Ils correspondent alors aux possibles pics R

- Etablissement seuil en amplitude et temporel afin de distinguer les pics R de ceux qui ne le sont pas
- Définition des extrémités des bords de l'IMF : ils correspondent au deuxième zéro à gauche et à droite de chaque pic R de la fonction $\widetilde{IMF}(t)$

ANNEXE 8 ALGORITHME DE FILTRAGE H-F

```
def ordre_bruit(t,s,alpha):
    IMF=tamissage(t,s, epsilon=0.3)
    #ordre_bruit va a la fois de determiner l'ordre du bruit et servir de compteur
    ordre_bruit=0
    IMF_average=np.array(IMF[0])
    while (np.mean(IMF_average)<alpha and ordre_bruit<(len(IMF)-2)):
        IMF_average+=np.array(IMF[ordre_bruit+1])
        if np.mean(IMF_average)<alpha:
            ordre_bruit+=1
        else:
            return min(5,ordre_bruit)
    return min(5,ordre_bruit)

def turkey_window(i_zg,i_zd):
    return tukey(i_zd-i_zg)

def phi(t,i_zg,i_zd):
    #on initialise la fonction de Turkey liee a la i-eme IMF en creant la fonction constante nulle
    fonction_Turkey=[0]*(np.size(t))
    for i in range(len(i_zg)):
        fonction_Turkey[i_zg[i]:i_zd[i]]=turkey_window(i_zg[i],i_zd[i])
    return np.array(fonction_Turkey)

def phi_complementaire(t,i_zg,i_zd):
    fct_phi=phi(t,i_zg,i_zd)
    phi_comp=0.1*(np.array([5])-fct_phi)
    return phi_comp

def ecg_filtre(t,s):
    ecg_filtre=np.zeros(np.size(s))
    IMF,indice_bruit=tamissage(t,s, epsilon=0.3),ordre_bruit(t,s,alpha=0.05)
    _,_,i_zg,i_zd=detect_QRS_intervals(t, s)
    #print(i_zg)
    #print(i_zd)
    for i in range(indice_bruit):
        fct_phi=phi(t,i_zg,i_zd)
        fct_phi_comp=phi_complementaire(t,i_zg,i_zd)
        ecg_filtre=np.multiply(np.array(IMF[i]),fct_phi)+np.multiply(np.array(IMF[i]),fct_phi_comp)
    for i in range(indice_bruit,len(IMF)):
        ecg_filtre+=np.array(IMF[i])
    return ecg_filtre
```

ANNEXE 9 METHODE DE FILTRAGE SAVITZKY-GOLAY



Soit I= $[i - \ell; i + \ell]$ une fenêtre glissante L'objectif est de lisser le signal dans cette fenêtre en considérant un polynôme de degré d, où $d < 2\ell + 1$ telle que la courbe soit la plus lisse possible dans l'intervalle

Sur chaque intervalle I, on peut alors déterminer la valeur lissée en fonction de ses voisins :

$$y_{i,lissée} = \sum_{-l}^{+l} b_{k+l} y_{i+k}$$

Où b_{k+l} sont les coefficients appelés coefficients de convolution

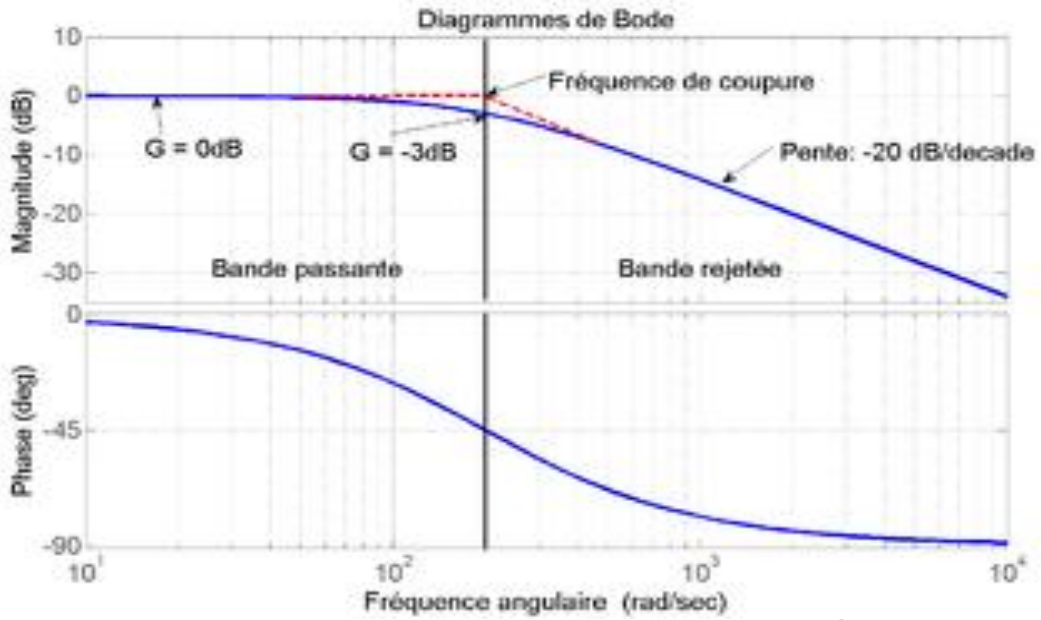
https://fr.wikipedia.org/wiki/Fichier:Lissage_sg3_anim.gif

ANNEXE 10 FILTRAGE PASSE-BAS DU PREMIER ORDRE

- Le filtre passe-bas du premier ordre est de la forme :

$$\tau \times \frac{ds}{dt}(t) + s(t) = e(t)$$

où $\frac{1}{\tau}$ correspond à la pulsation de coupure du filtre



https://upload.wikimedia.org/wikipedia/commons/thumb/0/06/FPBP1.png/360px-FPBP1.png

ANNEXE 11 ALGORITHME PYTHON DE COMPARAISON DES 3 METHODES DE FILTRAGE

```
def add_noise(s):
    noise = np.random.normal(0, 0.05, s.shape)
    return s + noise

def ecg_filtre(t,s):
    ecg_filtre=np.zeros(np.size(s))
    IMF,indice_bruit=tamissage(t,s, epsilon=0.3),ordre_bruit(t,s,alpha=0.05)
    _,_,i_zg,i_zd=detect_QRS_intervals(t, s)
    #print(i_zg)
    #print(i_zd)
    for i in range(indice_bruit):
        fct_phi=phi(t,i_zg,i_zd)
        fct_phi_comp=phi_complementaire(t,i_zg,i_zd)
        ecg_filtre=np.multiply(np.array(IMF[i]),fct_phi)+np.multiply(np.array(IMF[i]),fct_phi_comp)
    for i in range(indice_bruit,len(IMF)):
        ecg_filtre+=np.array(IMF[i])
    return ecg_filtre

def savitzky_golay(y, window_size=51, order=7):
    return savgol_filter(y, window_size, order)

def filtre_passe_bas(t,e,fc):
    tau = 1/(2*np.pi*fc)
    # Préparation de la liste de sortie
    s = []
    s.append(0)
    # Application du filtre
    for i in range(1, len(e)):
        s.append(s[i-1]+(t[i]-t[i-1])/tau*(e[i-1]-s[i-1]))
    return s

def MSE(s1,s2):
    return np.sum(((s1-s2)**2)/np.size(s1))
```

```
_,t,s,_ =read_file()
t=np.array(t)
s=np.array(s)
noisy_signal=add_noise(s)

fc=8.0
noisy_signal_filtered=ecg_filtre(t,noisy_signal)
noisy_signal_filtered_SG=savitzky_golay(noisy_signal, window_size=51, order=7)
noisy_signal_filtered_PB=np.array(filtre_passe_bas(t,noisy_signal,fc))

print("MSE EMD", MSE(noisy_signal,noisy_signal_filtered))
print("MSE SG", MSE(noisy_signal,noisy_signal_filtered_SG))
print("MSE PB", MSE(noisy_signal,noisy_signal_filtered_PB))

plt.figure()

plt.subplot(311)
plt.plot(t,s,label="signal original")
plt.plot(t,noisy_signal_filtered,'k',label="signal filtre")
plt.legend()
plt.title("Filtrage par EMD")

plt.subplot(312)
plt.plot(t,s,label="signal original")
plt.plot(t,noisy_signal_filtered_SG,label="ecg filtre par algorithme de savitzky_golay")
plt.legend()
plt.title("Filtrage par la methode de Savitzky-Golay")

plt.subplot(313)
plt.plot(t,s,label="signal original")
plt.plot(t,noisy_signal_filtered_PB,label="ecg filtre par filtre passe-bas de frequence de coupure fc="+str(fc)+"Hz")
plt.legend()
plt.title("Filtrage par filtre passe-bas de frequence f=8.0 Hz")

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

ANNEXE 12 DETCTION DES POINTS DE NON ACTIVITE

Les points de non-activité sont des points de potentiel électrique nul appartenant au segment ST

On définit les points de non activité du cœur de la façon suivante :

$$t_{na} = \overline{t_d} + T$$

$\overline{t_d}$ = extrémité droite d'un complexe
T= valeur imposée empiriquement afin que t_{na} appartienne tout le temps au segment ST

Expérimentalement, on détermine que le valeur de T vaut:

T= 20*0,0027=0,054s

ANNEXE 13 ALGORITHME DE FILTRAGE DE LA LIGNE DE BASE

```
def IMF_filtrage_BL(t,s):
    L=tamissage(t,s, epsilon=0.3)
    IMF_filtrage=np.array(L[0])+np.array(L[1])+np.array(L[2])
    IMF_carre=IMF_filtrage**2
    return IMF_carre, IMF_filtrage

def seuillage(t,s):
    IMF2, IMF = IMF_filtrage_BL(t,s)
    indexes = peakutils.indexes(IMF2, thres=0.2, min_dist=10)
    t_maxi = t[indexes]
    maxi = s[indexes]
    return maxi,t_maxi,indexes, IMF
```

```
def point_de_non_activite(t,s,T):
    _,_,_,i_zd=detect_QRS_intervals(t, s)
    i_zd=np.array(i_zd)
    #on definit la date d'aaprition du point de non activite
    i_t0=i_zd+np.array([T]*np.size(i_zd))
    if i_t0[-1]>np.size(t):
        i_t1=i_t0[:-1]
        return t[i_t1],s[i_t1]
       # print(np.size(i_t1))
    #print(np.size(i_t0))
    return t[i_t0],s[i_t0]

def ligne_de_base(t,s,T):
    t1,y1=point_de_non_activite(t,s,T)
    t1=np.append(t1,t[-1])
    t1=np.insert(t1,0,t[0])
    y1=np.append(y1,s[-1])
    y1=np.insert(y1,0,s[0])
    if len(t1)<4:
        return spline_lineaire(t,t1,y1)
    return spline_cubique(t,t1,y1)

def ecg_partiellement_filtre(t,s,T):
    ligne_base=ligne_de_base(t,s,T)
    ecg_partiellement_filtre=s-ligne_base
    return ecg_partiellement_filtre
```

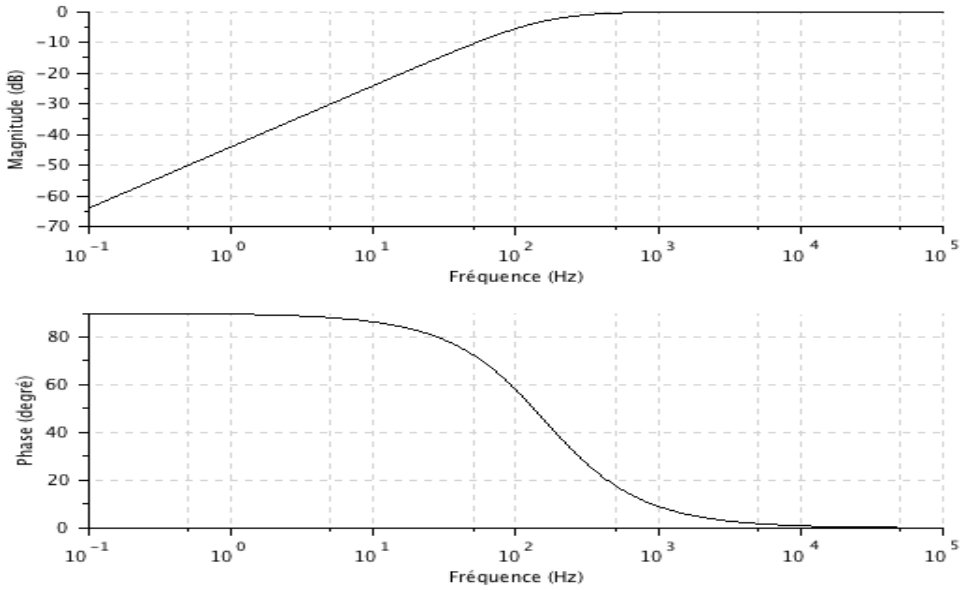
```
def zero_crossing_ind(array):
    return np.where(np.diff(np.sign(array)))[0]

def detect_QRS_intervals(t, s):
    #on recupere les indices des pics R et le signal a filtrer
    _, _, R_ind, imf_filtered = seuillage(t, s)
    #on recupere les indices des poits d'intersection du signal avec l'axe des abscisses
    zero_ind = zero_crossing_ind(imf_filtered)
    #on cree les listes permettant de recuperer les indices des extremites gauches et droites du signal
    zg_ind_list, zd_ind_list = [], []
    for r in R_ind:
        if len(zero_ind[zero_ind<r]) < 2 or len(zero_ind[zero_ind>r]) < 2:
            #print("QRS pas entièrement contenu dans le signal pour r = "+str(r))
            # le complexe QRS n'est pas entièrement contenu dans le signal
            continue
        #on recupere l'avant dernier indice des points d'intersection avant le pic R et le deuxieme indice apres le pic R
        zg_ind_list.append(zero_ind[zero_ind<r][-2])
        zd_ind_list.append(zero_ind[zero_ind>r][1])
    return t[zg_ind_list], t[zd_ind_list], zg_ind_list, zd_ind_list
```

ANNEXE 14 FILTRAGE PASSE-HAUT DU PREMIER ORDRE

Le filtre passe-haut du premier ordre est de la forme :

$$\tau \times \frac{ds}{dt}(t) + s(t) = \frac{de}{dt}(t)$$

où $\frac{1}{\tau}$ correspond à la pulsation de coupure du filtre



Bode_RC_HP.png (610×460) (unisciel.fr)