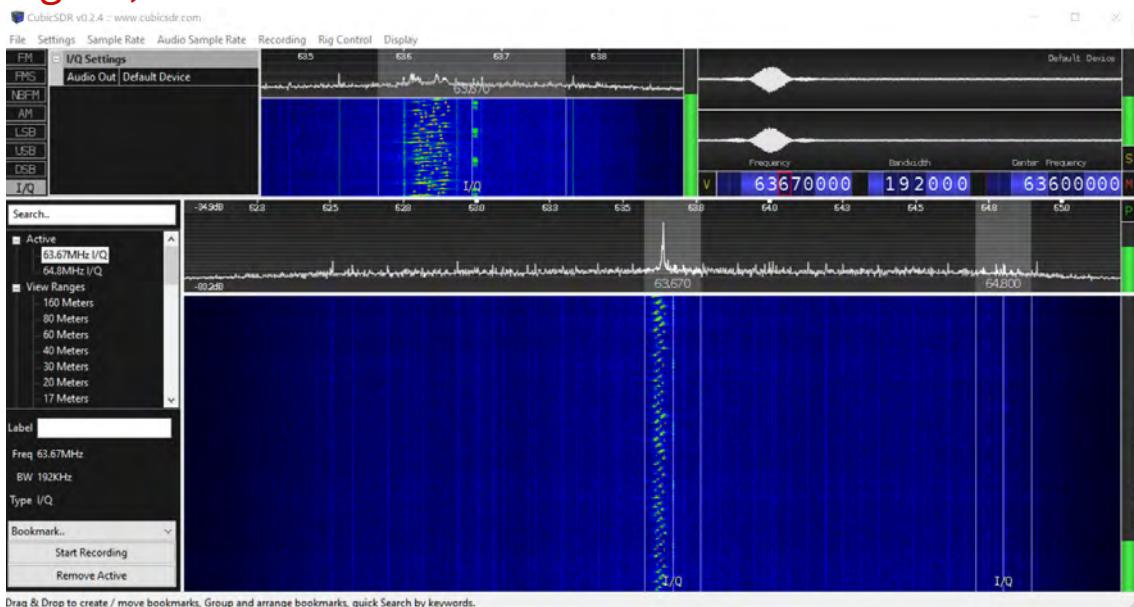


# SDR4MR script.nb

Lecture et analyse fichier WAV créé par une radio logicielle et un logiciel, CubicSDR ici.



HSJ 04/07/2024

V2 11/07/2024 : quelques corrections,  
améliorations.

d'après ReadWav SDR

1. fichier

2. lecture => data

3. Visualisation partielle

4. Analyse fréquentielle

permet une bonne estimation de la fréquence  
de démodulation à appliquer en 5.

## 5. Visualisation détaillée (R/I, phase, module) filtrage passe-bas et démodulation complémentaire.

## 6. Graphique R, I, module

In[200]:=

```
nnb = Last@FileNameSplit@NotebookFileName[];
version = "1.1";
verbeux = False;
```

In[203]:=

```
SetOptions[$FrontEndSession, DockedCells →
  Cell[BoxData[ToBoxes[nnb <> " " <> version <> " sélection du fichier .WAV, analyses
  temporelles et fréquentielles, démodulation numérique",
  StandardForm]], "DockedCells", ShowStringCharacters → False,
  CellMargins → {{0, 0}, {0, 0}}, Background → Orange]
  ]
```

### 1. Sélection du fichier .wav ici fichier exemple :

In[121]:=

```
fichier = "C:\\\\Users\\\\HSJ\\\\Documents\\\\EN_COURS\\\\Publis\\\\RF
  Monitor\\\\MAGMA\\\\Documents pour Magma\\\\63.640_2024-12-02_15-16-43.wav"
  ]
```

Out[121]=

```
C:\\Users\\HSJ\\Documents\\EN_COURS\\Publis\\RF
  Monitor\\MAGMA\\Documents pour Magma\\63.640_2024-12-02_15-16-43.wav
```

### 2. Lecture fichier => data

Variables :

tEch : période d'échantillonnage en secondes

nbPoints : nombre de points (paire de points)

date : date heure d'enregistrement du fichier

Avec la radio logicielle RTL SDR & CubicSDR : fichier Wav avec 2 octets par point.

In[122]:=

```

data = Import[fichier, "Data"];
    [importe
nbPoints = Dimensions[data][[2]];
    [dimensions
Print["Nombre de paires de points ; " <> ToString[nbPoints]]
[imprime                                [en chaîne de caractères
tEch = 1 / Import[fichier, "SampleRate"];
    [importe                            [taux d'échantillonnage
Print["Période d'échantillonnage : " <> ToString[N[tEch 10^6]] <> " microsecondes"]
[imprime                                [en chaîne... valeur numérique
Print["Durée enregistrée : " <> ToString[N[tEch nbPoints]] <> " secondes"]
[imprime                                [en chaîne... valeur numérique
date = StringTake[fichier, -23]
    [extrais chaîne de caractères
Print["Valeur maximum (si 1 => saturation) : " <> ToString[Max[data]]]
[imprime                                [en chaîne... maximum
```

Nombre de paires de points ; 1657729

Période d'échantillonnage : 5.20833 microsecondes

Durée enregistrée : 8.63401 secondes

Out[128]=

2024-12-02\_15-16-43.wav

Valeur maximum (si 1 => saturation) : 0.926145

### 3. Visualisation partielle

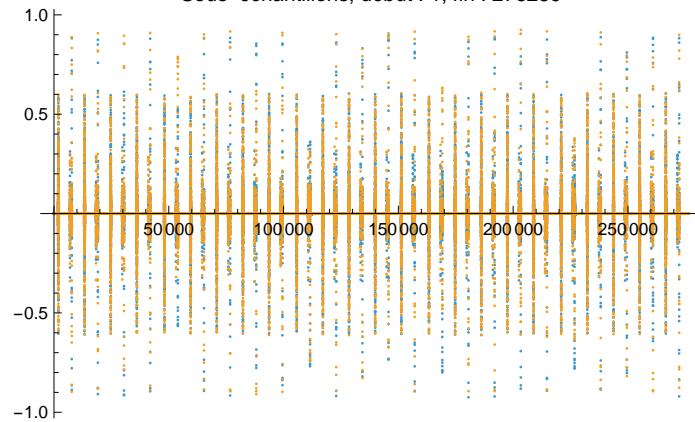
Affichage sous-échantillonné de la totalité du fichier  
(maximum maxAff 3000000 points ou duréeMaxAff secondes).

```
In[130]:= maxAff = 300 000;
duréeMaxAff = maxAff tEch;
sousEchant = Ceiling[nbPoints / maxAff];
    [plafond
tt1 = Take[data[[1]], {1, -1, sousEchant}]; [prends
tt2 = Take[data[[2]], {1, -1, sousEchant}]; [prends
nbPointsSousE = Dimensions[tt1][[1]] [dimensions
d1 = 1; f1 = nbPointsSousE;
If[f1 > nbPointsSousE || d1 > f1, d1 = 1; f1 = nbPointsSousE, rien];
[si
df = " Visualisation R I 1/" <> ToString[sousEchant] <>
    [unité im… en chaîne de caractères
" points\nenregistrement : " <> ToString[date] <>
    [en chaîne de caractères
" \nSous-échantillons, début : " <> ToString[d1] <> ", fin : " <> ToString[f1];
    [en chaîne de caractères [en chaîne de caractères
ListPlot[{tt1[[d1 ;; f1]], tt2[[d1 ;; f1]]}, PlotRange → All, PlotLabel → df]
[tracé de liste [zone de tracé [tout [étiquette de tracé
dPar = Abs[Transpose[{tt1[[d1 ;; f1]], tt2[[d1 ;; f1]]}.{1, I}]];
    [va… transpose [unité imaginaire
cutoff = 10 000;
dParLP = LowpassFilter[dPar, cutoff 2 π, SampleRate → 1 / tEch];
    [filtre passe-bas [taux d'échantillonnage
cutoffH = 0;
dParLPH = HighpassFilter[dParLP, cutoffH 2 π, SampleRate → 1 / tEch];
    [filtre passe-haut [taux d'échantillonnage
temps = Range[f1 - d1 + 1] tEch sousEchant;
    [plage
ListPlot[Transpose[{temps, dParLPH}], PlotRange → All,
[tracé de li… transpose [zone de tracé [tout
AxesLabel → {"Temps (s)", "RF(ua)"}, PlotLabel → "Visualisation du module"]
[étiquette des axes [étiquette de tracé
```

Out[135]= 276 289

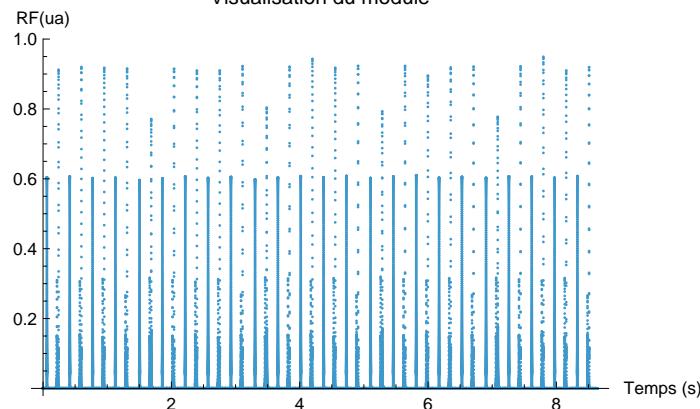
Out[139]=

Visualisation R I 1/6 points  
enregistrement : 2024-12-02\_15-16-43.wav  
Sous-échantillons, début : 1, fin : 276289



Out[145]=

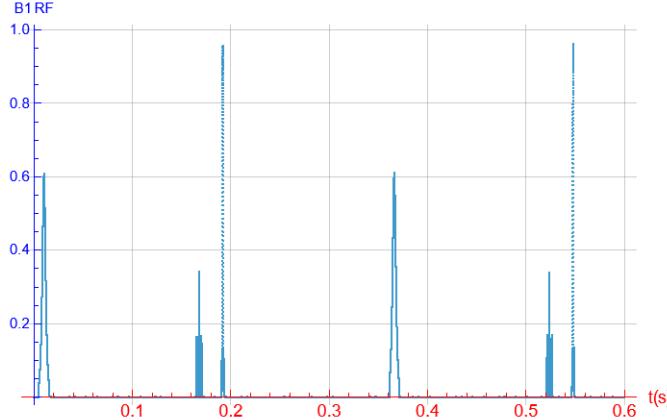
Visualisation du module



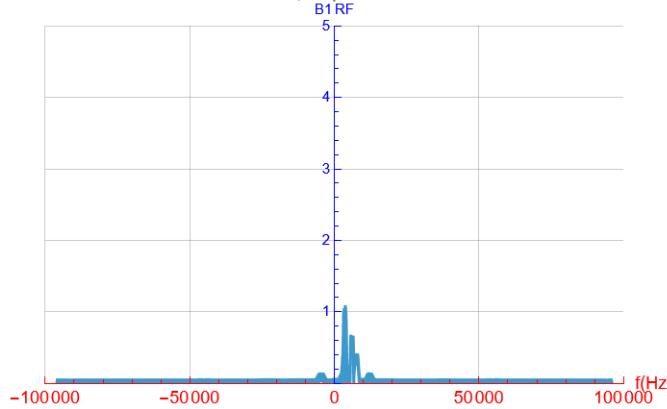
## 4. Analyse fréquentielle (ici temps en secondes)

Dans cet exemple, deux répétitions complètes de 50 à 650 ms => capture d'écran

enregistrement RI : 2024-12-02\_15-16-43.wav  
début : 9600, fin : 124800  
début (ms) : 50, fin (ms) : 650  
Δt affiché (ms) : 600



Max : 1.02695, fréquence Hz: 2998.31



In[146]:=

```

ok = False;
  [faux
ok = True
  [vrai
deb = 1;
If[ok, Manipulate[
  [si      [manipule
    duréeTotale = N[tEch nbPoints];
      [valeur numérique
    pasDurée = N[Round[100 duréeTotale] / 1000];
      [ arrondis
    débutP = Max[Round[débutSecondes / tEch], 1];
      [max [arrondis
    deltaVisu = Round[duréeObs / tEch];
      [arrondis
    finP = débutP + deltaVisu;
    If[finP > nbPoints, finP = nbPoints, rien];
      [si
    If[débutP ≥ nbPoints || débutP ≥ finP, débutP = 1, rien];
      [si
(* *)

```

```

(*coupure en Hz désormais;*);
dPar = Abs[Transpose[{data[[1]][débutP ;; finP], data[[2]][débutP ;; finP]}].{1, I}];  

    [va... [transpose] [unité im
riPar = {data[[1]][débutP ;; finP], data[[2]][débutP ;; finP]};  

Δts = N[tEch];  

    [valeur numérique
longueurP = finP - débutP + 1;
ΔfHz = 1 / (longueurP Δts);
tfHz = Table[((ii - 1) - Round[longueurP / 2]) ΔfHz, {ii, 1, longueurP}];  

    [table] [arrondis
tF = Abs[Fourier[(Transpose[riPar].{1, I})]];  

    [va... [transform... [transpose] [unité imaginaire
tF = RotateLeft[tF, Round[longueurP / 2]];  

    [faire pivoter à gauche [arrondis

df = " enregistrement RI : " <> ToString[date] <>
    [en chaîne de caractères
" \ndébut : " <> ToString[débutP] <>, fin : " <> ToString[finP] <>
    [en chaîne de caractères] [en chaîne de caractères
"\n début (ms) : " <> ToString[Round[N[1000 débutP tEch]]] <>
    [en chaîne... [arrondi... [valeur numérique
", fin (ms) : " <> ToString[Round[N[1000 (finP) tEch]]] <>
    [en chaîne... [arrondi... [valeur numérique
"\n Δt affiché (ms) : " <> ToString[Round[N[1000 (longueurP) tEch]]];  

    [en chaîne... [arrondi... [valeur numérique

fa = "Max : " <> ToString[Max[tF]] <>, fréquence Hz: " <>
    [maximum] [en chaîne... [maximum
ToString[Flatten[FindPeaks[tF, 1, 0, Max[tF] / 2][[1]]] ΔfHz + tfHz[[1]]];
    [en chaîne d... [aplatis] [trouve pics] [maximum

GraphicsColumn[
    [colonne de graphique
{ListPlot[TimeSeries[LowpassFilter[dPar, freqCoupure 2 π, SampleRate → 1 / tEch],
    [tracé de li... [série tempore... [filtre passe-bas] [taux d'échantillonage
    {0 tEch, (longueurP - 1) tEch}], PlotRange → All, AxesLabel → {"t(s)", RF B1},
    [zone de tracé] [tout] [étiquette des axes
    AxesStyle → {Directive[Red, 12], Blue}, PlotLabel → df, GridLines → Automatic,
    [style des axes] [directive] [rouge] [bleu] [étiquette de tracé] [lignes de grille] [automatique
    ListPlot[Transpose[{tfHz, tF}], PlotRange → {{-ftDf, ftDf}, {0, ftAmp}},
    [tracé de li... [transpose] [zone de tracé
    AxesLabel → {"f(Hz)", RF B1}, AxesStyle → {Directive[Red, 12], Blue},
    [étiquette des axes] [style des axes] [directive] [rouge] [bleu]
    PlotLabel → fa, GridLines → Automatic, Joined → True]],  

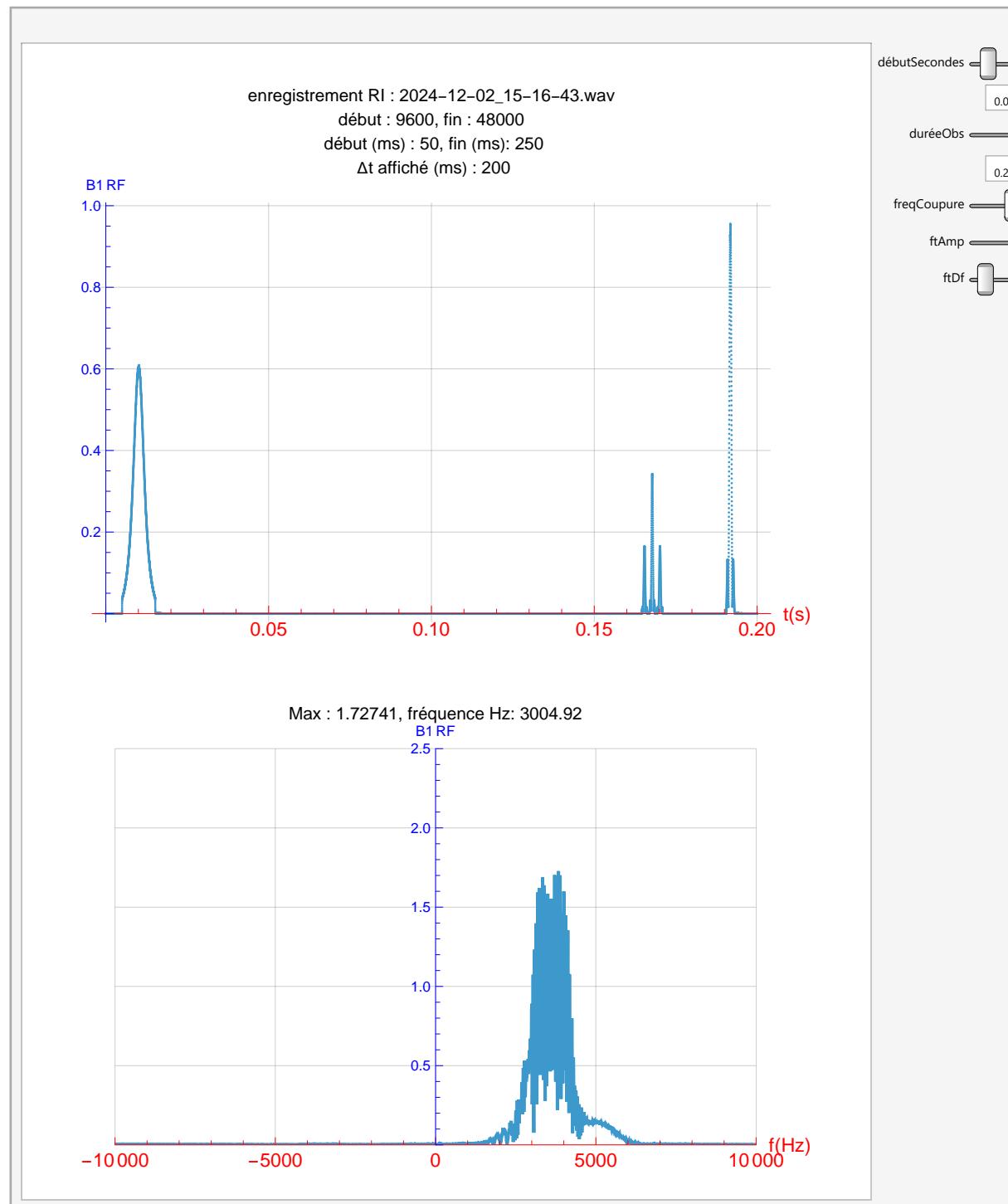
    [étiquette de tracé] [lignes de grille] [automatique] [joint] [vrai
{débutSecondes, 0, duréeTotale, pasDurée},
{{duréeObs, N[Round[1000 duréeMaxAff / 4] / 1000]}, 0, duréeMaxAff, 0.01},
    [..] [arrondis
{{freqCoupure, 10000}, 1000, 1 / (tEch 2), 1000},
{{ftAmp, 5}, 0.5, 10, 1}, {{ftDf, 100000}, 10000, 100000, 10000},
TrackedSymbols :> {débutSecondes, duréeObs, freqCoupure, ftAmp, ftDf},
[symboles suivis
ContinuousAction → False], rien]
[action continue] [faux

```

Out[147]=

True

Out[149]=




---

**5. Visualisation détaillée**  
attention on reprend le fichier complet et on regarde de début à longueur.

seuilT = seuil sur l'amplitude temporelle pour le calcul de la phase

Toutes les fréquences ci-dessous sont en Hz

Filtre passe-bas : fCoupureT (sur signal temporel)

Démodulation : la démodulation permet de décaler la fréquence du signal (=fIRM-fOscillateur local Radio logicielle) par exemple pour la ramener à 0 avec :

fDemod (quelques 10 kHz),

vernierfDemod (petites variations).

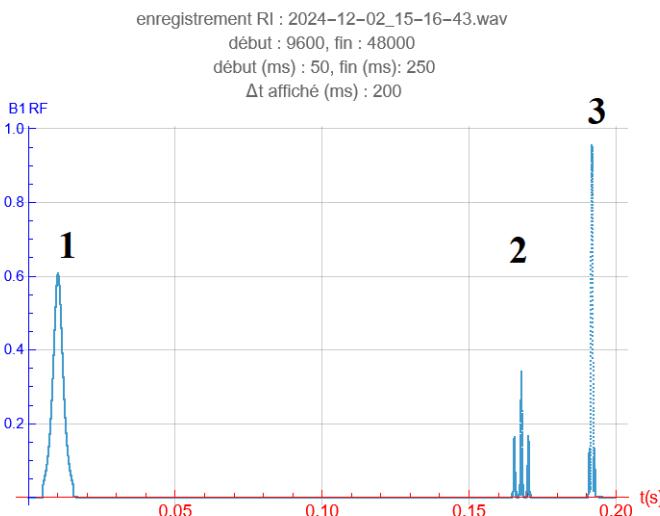
Attention :

- en 2D multicoupes la fréquence des impulsions fIRM change beaucoup d'une coupe à l'autre,

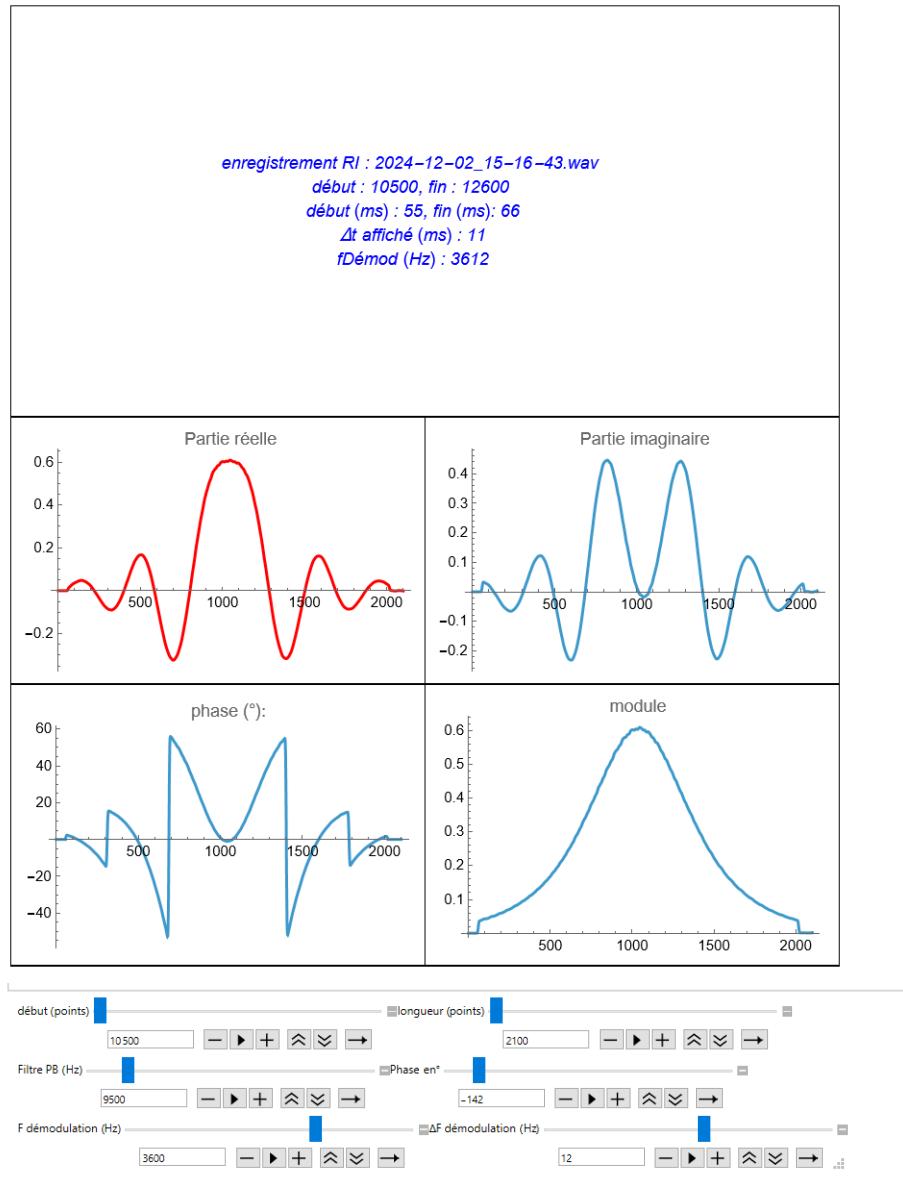
- Attention : la démodulation est appliquée à partir du temps initial.

phase0 : un décalage de phase peut être appliquée au signal pour mieux le visualiser.

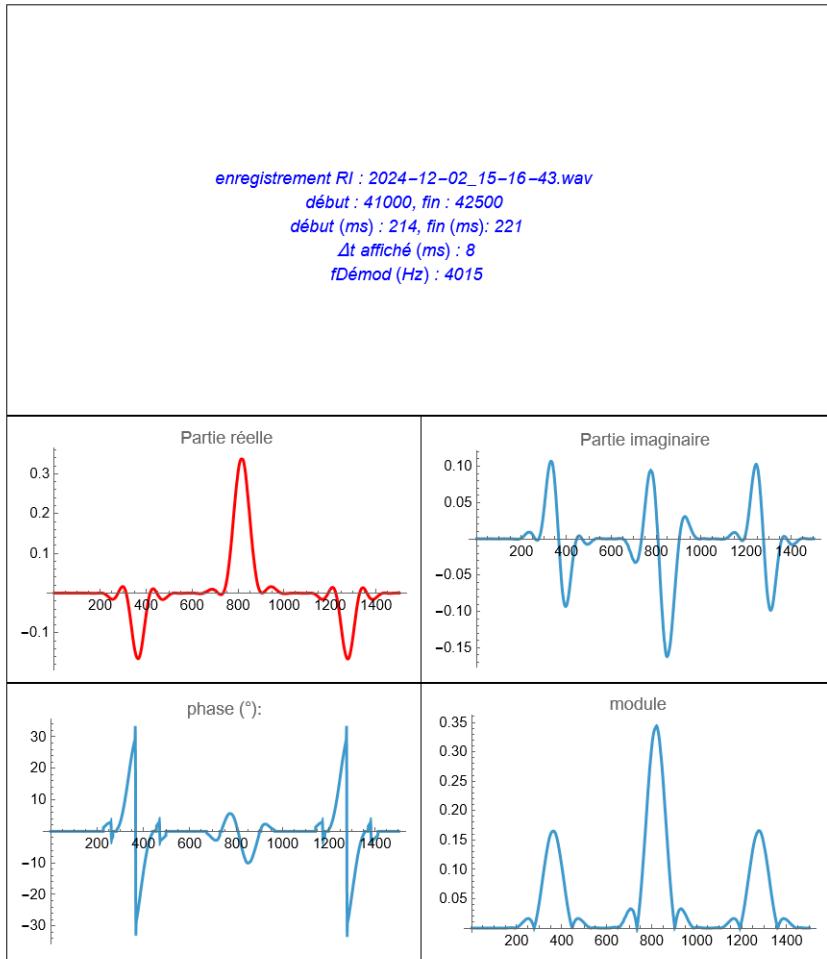
Analyse des impulsions successives :



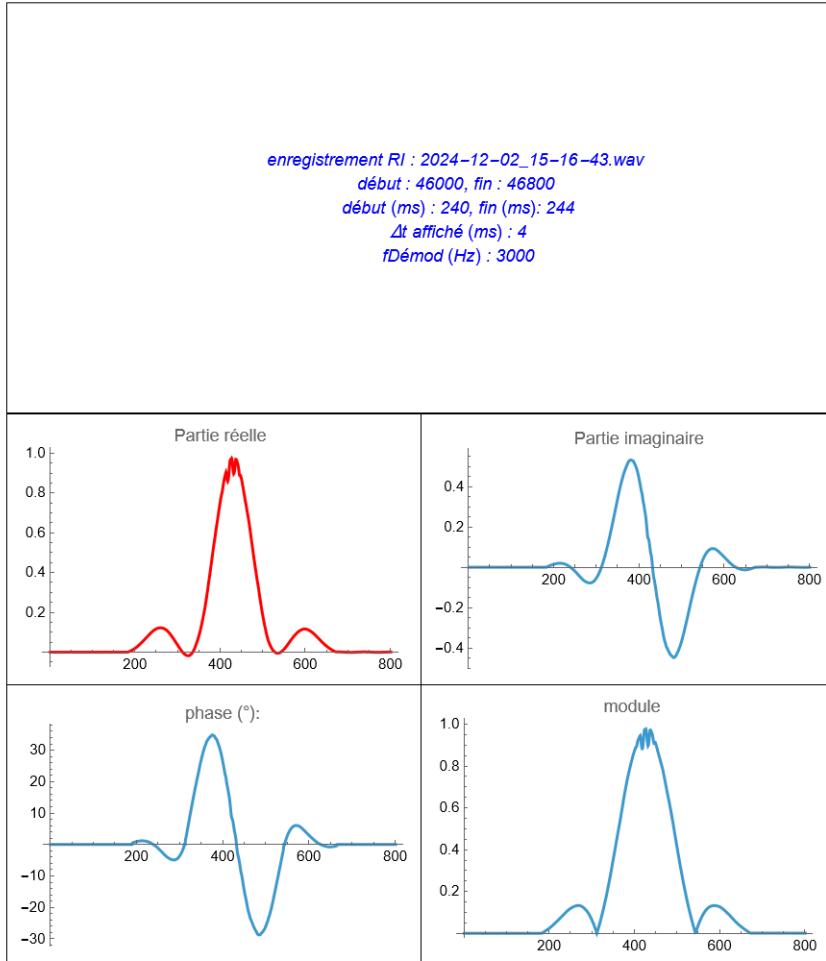
## 1. STIR



## 2. Excitation de l'eau



### 3. Pi



```
In[150]:= débutP = .;
longueurP = .;
Manipulate[
manipule
seuilT = 0.01;
maxVisu = Min[maxAff, nbPoints - 1];
minimum
If[débutP + longueurP > nbPoints, débutP = nbPoints - longueurP - 1, rien];
si
If[longueurP > maxVisu, longueurP = maxVisu, rien];
si
df = " enregistrement RI : " <> ToString[date] <> " \ndébut : " <> ToString[débutP] <>
    [en chaîne de caractères] <> ToString[débutP + longueurP] <> "\n fin : " <>
    [en chaîne de caractères]
ToString[Round[N[1000 débutP tEch]]] <> ", fin (ms) : " <> ToString[Round[N[1000
    [en chaîne... l'arrondi... valeur numérique] tEch]]] <> "\n Δt affiché (ms) : " <> ToString[Round[N[1000
    [en chaîne... l'arrondi... valeur numérique] (longueurP) tEch]]] <> "\n fDémod (Hz) : " <> ToString[fDemod + vernierfDemod];
    [en chaîne de caractères]
pR = "Partie réelle";
pI = "Partie imaginaire";
```

```

pPhase = " phase (°): ";
pMod = " module ";
temps = Range[longueurP + 1] tEch;
    plage
dParDt = Transpose[
    transpose
    {data[[1]] [[débutP ;; débutP + longueurP]], data[[2]] [[débutP ;; débutP + longueurP]]}];
dParD = dParDt.{1, I} Exp[I (2 π (fDemod + vernierfDemod) temps + phase0 π / 180)];
    ... lex... [unité imaginaire

dParD = LowpassFilter[dParD, fCoupureT 2 π, SampleRate → 1 / tEch];
    filtre passe-bas                                taux d'échantillonnage

dParM = Abs[dParD];
    valeur absolue

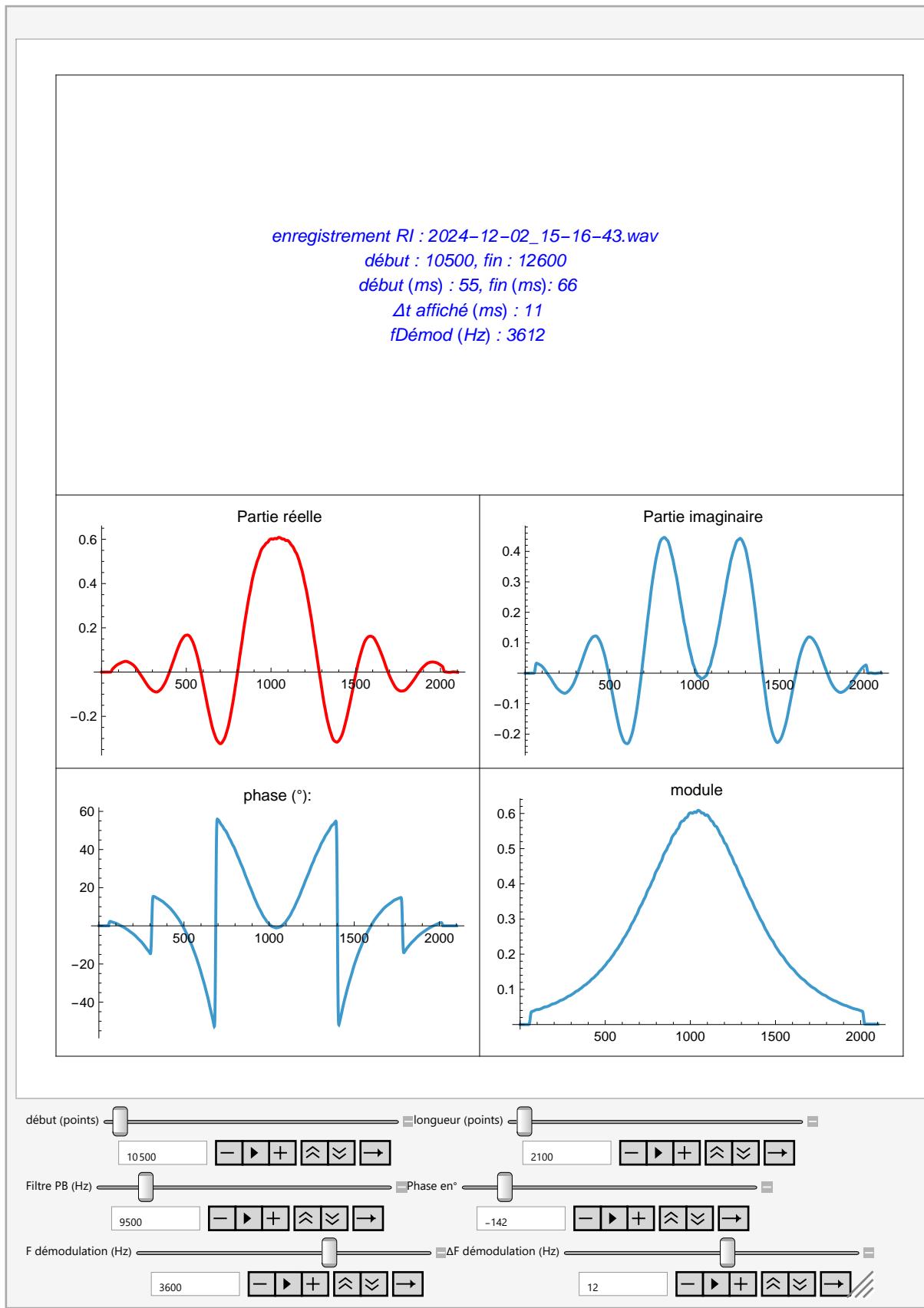
unZero = Threshold[dParM, seuilT];
    seuil

dParP = LowpassFilter[Arg[dParD], fCoupureT 2 π, SampleRate → 1 / tEch] unZero;
    filtre passe-bas      argument                taux d'échantillonnage

GraphicsGrid[{{Graphics[Text[Style[df, Blue, Italic, 12]]], SpanFromLeft}, {ListPlot[
    grille de graphique  graphique  texte style bleu italic recouvre depuis la ... tracé de liste
    {N[Re[dParD]]}, Joined → True, PlotRange → All, PlotLabel → pR, PlotStyle → Red],
        partie réelle joint vrai zone de tracé tout étiquette de tracé style de tracé rouge
    ListPlot[{N[Im[dParD]]}, Joined → True, PlotRange → All, PlotLabel → pI}],
        tracé de liste ... partie imaginaire joint vrai zone de tracé tout étiquette de tracé
    {ListPlot[{N[180 dParP / π]}, Joined → True, PlotRange → All, PlotLabel → pPhase],
        tracé de liste valeur numérique joint vrai zone de tracé tout étiquette de tracé
    ListPlot[{N[dParM]}, Joined → True, PlotRange → All, PlotLabel → pMod]}
        tracé de liste valeur numér... joint vrai zone de tracé tout étiquette de tracé
}, Frame → All], Row[{Control[
    cadre tout ran... contrôle
    {{débutP, 1, "début (points)"}, 1, nbPoints, 500, ControlPlacement → Bottom}],
        positionnement de cont... bas
    Control[{{longueurP, Round[maxVisu / 2], "longueur (points)"},,
        contrôle arrondis
        1, maxVisu, 1000, ControlPlacement → Bottom}]],
        positionnement de cont... bas
    Row[{Control[{{fCoupureT, 30000, "Filtre PB (Hz)"}, 500,
        ran... contrôle
        70000, 1000, ControlPlacement → Bottom}],
        positionnement de cont... bas
    Control[{{phase0, 0, "Phase en°"}, -180, 180, 2, ControlPlacement → Bottom}]}],
        contrôle positionnement de cont... bas
    Row[{Control[{{fDemod, 0, "F démodulation (Hz)"}, -30000, 20000, 1000,
        ran... contrôle
        ControlPlacement → Bottom}], Control[{{vernierfDemod, 0, "ΔF démodulation (Hz)"},,
            positionnement de cont... bas     contrôle
            -100, 100, 0.5, ControlPlacement → Bottom}]}],
            positionnement de cont... bas
    ControlPlacement → {Bottom, Bottom, Bottom, Bottom},
        positionnement de contrôle has has has has
    ]}]

```

```
TrackedSymbols :> {débutP, longueurP, fCoupureT, phase0, fDemod, vernierfDemod},  
|symboles suivis  
Method -> {"ControlAreaDisplayFunction" ->  
|méthode  
  (Pane[#1, ImageSize -> Automatic, ImageSizeAction -> "ResizeToFit",  
  |volet    |taille d'image  |automatique  |action de taille d'image  
  AppearanceElements -> {"ResizeArea"}] &) }, ContinuousAction -> False]  
|éléments d'apparence           |action continue   |faux
```



Graphique de la dernière impulsion observée au §5

## (R, I, module)

dec : permet de centrer l'impulsion (axe des temps)

PlotRange $\rightarrow\{-8,8\},\{-0.4,0.8\}\Rightarrow$  ajustement du graphique

```
In[195]:= ms = 1000; np = Length[dParD];
dec = -0.0055;
tax = Table[dec + ii tEch, {ii, 1, np}] ms;
ListPlot[{Transpose[{tax, N[Re[dParD]]}], Transpose[{tax, N[Im[dParD]]}], Transpose[
{tax, N[dParM]}]}, Joined -> True, GridLines -> Automatic, PlotStyle -> {Red, Blue,
Green}, AxesStyle -> {{14, Bold}, {14, Bold}}, PlotRange -> {{-8, 8}, {-0.4, 0.8}},
AxesLabel -> {"t (ms)", "B1 (a.u.)"}, LabelStyle -> Directive[Black, Bold]]
Out[197]=
```

