Radio logicielle 1/4 : introduction et prise en main de GNU Radio

J.-M Friedt

FEMTO-ST/département temps-fréquence, Besançon

jmfriedt@femto-st.fr

transparents et ressources à `jmfriedt.free.fr`

15 janvier 2025

Plan des interventions

$4 \times 3\ h$:

Introduction à la radio logicielle (SDR) et GNU Radio

1. Concepts associés aux signaux échantillonnés en temps discret,
2. bande de base représentée par des signaux complexes
3. illustration lors de la prise en main de GNU Radio

Utilisation de la radio logicielle pour la conception de RADARs

1. RADAR passif : corrélation et mise en œuvre du RADAR multistatique
2. RADAR actif, synchronisation émission/réception du RADAR monostatique
3. SAR, mise en œuvre par SDR

Utilisation de la radio logicielle pour le traitement de signaux de navigation par satellite

1. Bilan de liaison et compression d'impulsion
2. Exploitation des signaux GPS L1 et Galileo E1
3. Sécurité (brouillage, leurrage) et CRPA

Utilisation "avancée" de GNU Radio

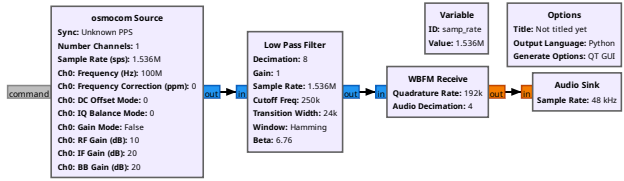
1. Contrôle de la chaîne de traitement (client/serveur)
2. Blocs de traitement dédiés (Python, C++)
3. GNU Radio sur systèmes embarqués (Buildroot)

Matériel v.s logiciel

Récepteur FM matériel

Très spécialisé, nécessite l'équipement adéquat pour déverminer (oscilloscope, analyseur de spectre ...)

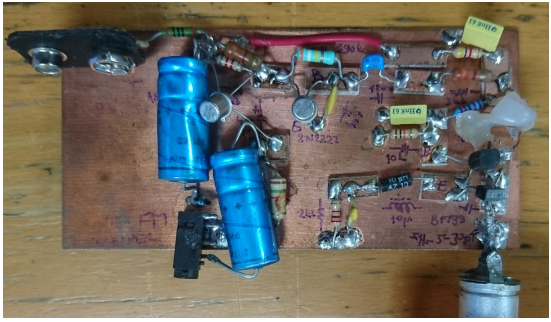
Récepteur logiciel



Un seul matériel pour de multiples applications, mise à jour par logiciel, assemblage de blocs aux fonctions de traitement numérique du signal écantillonné en temps discret maîtrisées

Matériel v.s logiciel

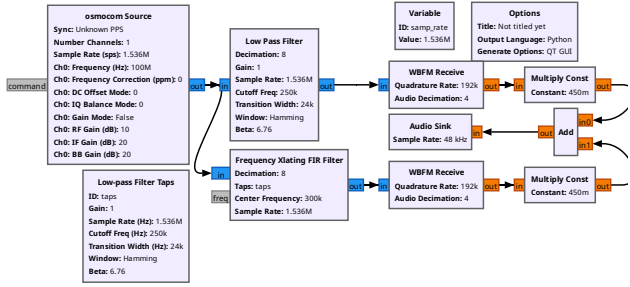
Récepteur FM matériel



Très spécialisé, nécessite l'équipement adéquat pour déverminer (oscilloscope, analyseur de spectre ...)

Réalisation Ivan Ryger, FEMTO-ST, Besançon

Récepteur logiciel

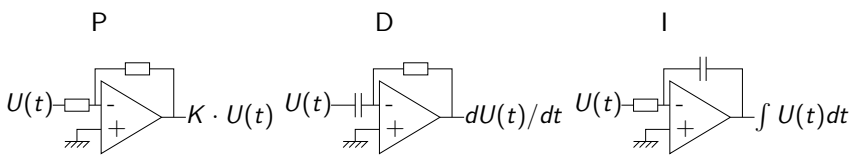


Un seul matériel pour de multiples applications, mise à jour par logiciel, assemblage de blocs aux fonctions de traitement numérique du signal écantillonné en temps discret maîtrisées

Matériel v.s logiciel

Passage de l'implémentation **matérielle** à *logicielle* de centrales inertielles (1950s) [1]

- **Stabilité** : un algorithme numériques ne dérive pas dans le temps ou ne présente pas de variation avec l'environnement
- **Flexibilité** : une même plateforme matérielle supporte une multitude d'algorithmes
- **Reconfigurabilité** : capacité à modifier les paramètres de fonctionement



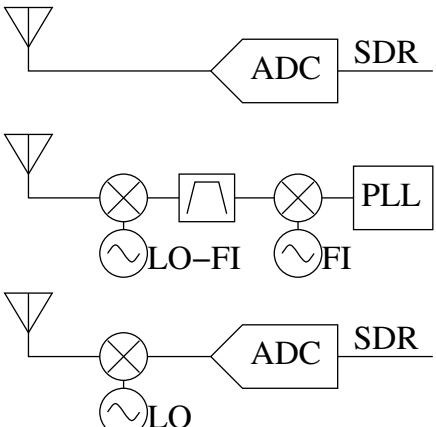
$$s_n = K \cdot \varepsilon_n + K_d \frac{\varepsilon_n - \varepsilon_{n-1}}{T_s} + \sum \varepsilon_n \cdot T_s$$

$$\Leftrightarrow s_n = s_{n-1} + K \cdot (\varepsilon_n - \varepsilon_{n-1}) + K_d \frac{\varepsilon_n - 2\varepsilon_{n-1} + \varepsilon_{n-2}}{T_s} + T_s \varepsilon_n$$

- Excellente **stabilité** du numérique à long terme (pas de dérive), mais **bande passante limitée** (f_s à x00 MHz mais coefficients de FIR)

1. D.A. Mindell, *Digital Apollo : human and machine in spaceflight*, MIT Press (2011)

Récepteur radiofréquence matériel v.s logiciel



- Ce qu'on veut : solution tout logiciel, avec une antenne et un convertisseur analogique numérique.
  Inconvénients : dynamique limitée et l'antenne se comporte comme filtre passe-bande
- Solution **matérielle** : une seule fonction, spécifiée à la conception du circuit (e.g. réception FM)
- Solution **pratique** : une première transposition de fréquence analogique avec amplificateur à gain variable, puis numérisation en bande de base avant de démoduler numériquement le signal.

"In addition to easing the scheduling and configuration burden, an autonomous radio also will **gracefully handle unpredictable or anomalous events**. For example, during entry, descent, and landing (EDL), a spacecraft can undergo large Doppler swings caused by rocket firings, parachute openings, backshell ejection, and a bouncing landing on the surface. Even when all scheduled events occur successfully, there may be Doppler uncertainty due to unpredictable properties of the atmosphere. Ideally, the communication link should operate whether or not each of the EDL events is successful, but the uncertainties involved typically lead to liberal link margins—for example, the **Mars Exploration Rovers** observed link margins that sometimes exceeded 10 dB. An autonomous radio could substantially reduce this margin because it would handle any Doppler swing nearly optimally.

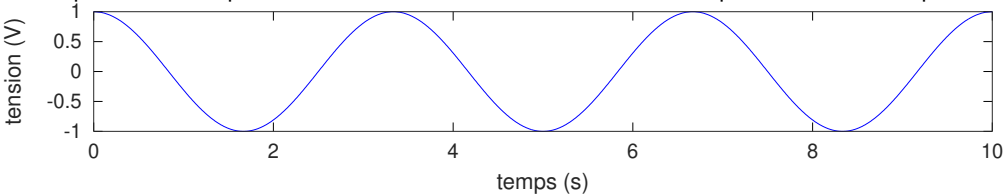
Unfortunately, such flexible technology is not available on NASA's currently flying missions. In perhaps the most glaring example of this, NASA engineers discovered in 2000 that a receiver aboard Cassini, launched in 1997, would fail during the Huygens probe descent onto Titan because it did **not properly account for the Doppler profile of the probe**. Increasing the loop bandwidth of the synchronization loops would have easily fixed the problem, but, unfortunately, these loop bandwidths were hard-wired to fixed values on the spacecraft. With superior engineering and enormous dedication, NASA and the European Space Agency were still able to save the mission by slightly altering the original trajectory, but this solution required forming a large and expensive international recovery team to find the appropriate recommendations on how to overcome the radio's severe limitations."

Échantillonnage en temps et niveaux discrets

- échantillonnage en **niveaux** discrets
  - $valeur = \frac{V}{V_{ref}} \times (2^N - 1)$
  - dynamique limitée par la plus petite variation de tension détectable $\Delta V = V_{ref}/(2^N - 1)$
  - en radiofréquence : puissance $20 \log_{10}(\Delta V)$ vaut 48 dB sur 8 bits, 60 dB sur 10 bits, 72 dB sur 12 bits et 96 dB sur 16 bits
- échantillonnage en **temps** discret
  - fréquence d'échantillonnage f_s (période d'échantillonnage $1/f_s$)
  - hypothèse de périodicité du spectre
  - extension du spectre de $-f_s/2$ à $+f_s/2$
  - f_s connu $\Rightarrow$ on décide de nommer $f_s/2$ (fréquence de Nyquist) à 1 et d'exprimer toutes les fréquences f comme fréquences normalisées

$$f/(f_s/2) = 2 \times f/f_s \in [-1 : 1]$$

  - acquisition sur N points et transformée de Fourier discrète sur N points : résolution spectrale f_s/N

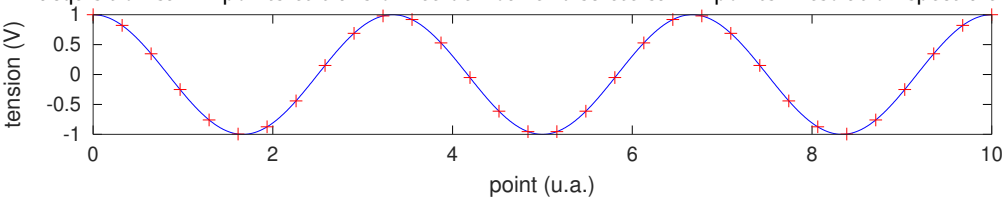


Échantillonnage en temps et niveaux discrets

- échantillonnage en **niveaux** discrets
  - $valeur = \frac{V}{V_{ref}} \times (2^N - 1)$
  - dynamique limitée par la plus petite variation de tension détectable $\Delta V = V_{ref}/(2^N - 1)$
  - en radiofréquence : puissance $20 \log_{10}(\Delta V)$ vaut 48 dB sur 8 bits, 60 dB sur 10 bits, 72 dB sur 12 bits et 96 dB sur 16 bits
- échantillonnage en **temps** discret
  - fréquence d'échantillonnage f_s (période d'échantillonnage $1/f_s$)
  - hypothèse de périodicité du spectre
  - extension du spectre de $-f_s/2$ à $+f_s/2$
  - f_s connu $\Rightarrow$ on décide de nommer $f_s/2$ (fréquence de Nyquist) à 1 et d'exprimer toutes les fréquences f comme fréquences normalisées

$$f/(f_s/2) = 2 \times f/f_s \in [-1 : 1]$$

  - acquisition sur N points et transformée de Fourier discrète sur N points : résolution spectrale f_s/N

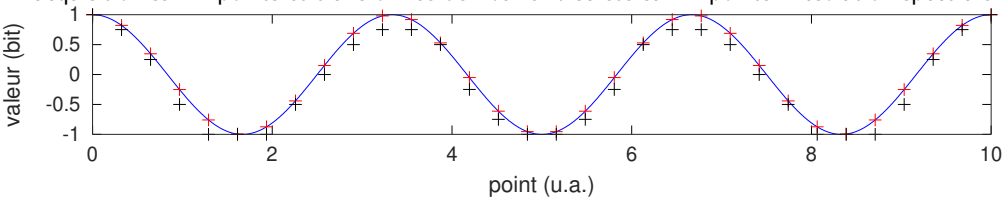


Échantillonnage en temps et niveaux discrets

- échantillonnage en **niveaux** discrets
  - $valeur = \frac{V}{V_{ref}} \times (2^N - 1)$
  - dynamique limitée par la plus petite variation de tension détectable $\Delta V = V_{ref}/(2^N - 1)$
  - en radiofréquence : puissance $20 \log_{10}(\Delta V)$ vaut 48 dB sur 8 bits, 60 dB sur 10 bits, 72 dB sur 12 bits et 96 dB sur 16 bits
- échantillonnage en **temps** discret
  - fréquence d'échantillonnage f_s (période d'échantillonnage $1/f_s$)
  - hypothèse de périodicité du spectre
  - extension du spectre de $-f_s/2$ à $+f_s/2$
  - f_s connu $\Rightarrow$ on décide de nommer $f_s/2$ (fréquence de Nyquist) à 1 et d'exprimer toutes les fréquences f comme fréquences normalisées

$$f/(f_s/2) = 2 \times f/f_s \in [-1 : 1]$$

  - acquisition sur N points et transformée de Fourier discrète sur N points : résolution spectrale f_s/N

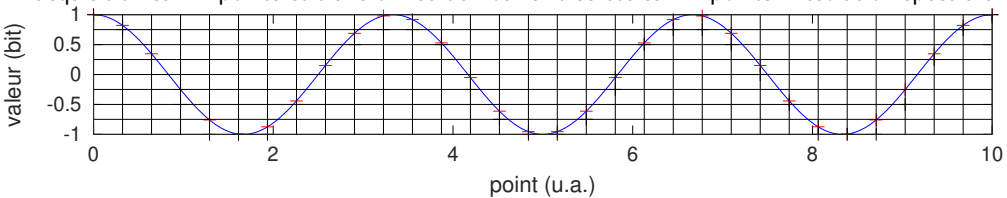


Échantillonnage en temps et niveaux discrets

- échantillonnage en **niveaux** discrets
  - $valeur = \frac{V}{V_{ref}} \times (2^N - 1)$
  - dynamique limitée par la plus petite variation de tension détectable $\Delta V = V_{ref}/(2^N - 1)$
  - en radiofréquence : puissance $20 \log_{10}(\Delta V)$ vaut 48 dB sur 8 bits, 60 dB sur 10 bits, 72 dB sur 12 bits et 96 dB sur 16 bits
- échantillonnage en **temps** discret
  - fréquence d'échantillonnage f_s (période d'échantillonnage $1/f_s$)
  - hypothèse de périodicité du spectre
  - extension du spectre de $-f_s/2$ à $+f_s/2$
  - f_s connu $\Rightarrow$ on décide de nommer $f_s/2$ (fréquence de Nyquist) à 1 et d'exprimer toutes les fréquences f comme fréquences normalisées

$$f/(f_s/2) = 2 \times f/f_s \in [-1 : 1]$$

  - acquisition sur N points et transformée de Fourier discrète sur N points : résolution spectrale f_s/N



Mélangeur I, Q

- mélanger pour transposer une fréquence :

$$A_1 \sin(\omega_1 t + \varphi) \cdot A_2 \cos(\omega_2 t) \propto A_1 \cdot A_2 \cdot \sin((\omega_1 + \omega_2)t + \varphi) + \sin((\omega_1 - \omega_2)t + \varphi)$$

$$\propto A_1 \cdot A_2 \cdot \sin(\varphi) \text{ si } \omega_1 = \omega_2$$

- coefficients **I**(dentité) et **Q**(uadrature) : $A \simeq \sqrt{I^2 + Q^2}$, $\varphi \simeq \arctan(Q/I)$
- un signal réel (porteuse modulée) est devenu complexe en sortie du démodulateur

Mélangeur I, Q

- mélanger pour transposer une fréquence :

$$A_1 \sin(\omega_1 t + \varphi) \cdot A_2 \cos(\omega_2 t) \propto A_1 \cdot A_2 \cdot \sin((\omega_1 + \omega_2)t + \varphi) + \sin((\omega_1 - \omega_2)t + \varphi)$$

$$\propto A_1 \cdot A_2 \cdot \sin(\varphi) \text{ si } \omega_1 = \omega_2$$

- le terme “somme de fréquences” est éliminé par filtre passe-bas

- coefficients **I**(dentité) et **Q**(uadrature) : $A \simeq \sqrt{I^2 + Q^2}$, $\varphi \simeq \arctan(Q/I)$
- un signal réel (porteuse modulée) est devenu complexe en sortie du démodulateur

Mélangeur I, Q

- mélanger pour transposer une fréquence :

$$A_1 \sin(\omega_1 t + \varphi) \cdot A_2 \cos(\omega_2 t) \propto A_1 \cdot A_2 \cdot \sin((\omega_1 + \omega_2)t + \varphi) + \sin((\omega_1 - \omega_2)t + \varphi)$$

$$\propto A_1 \cdot A_2 \cdot \sin(\varphi) \text{ si } \omega_1 = \omega_2$$

- le terme "somme de fréquences" est éliminé par filtre passe-bas
- si $\varphi = 0$, la sortie est toujours nulle quelquesoient A_1, A_2
- utiliser la version transposée de 90^o du signal de référence : cos $\rightarrow$ sin

$$A_1 \sin(\omega_1 t + \varphi) \cdot A_2 \sin(\omega_2 t) \propto A_1 \cdot A_2 \cdot \cos((\omega_1 + \omega_2)t + \varphi) + \cos((\omega_1 - \omega_2)t + \varphi)$$

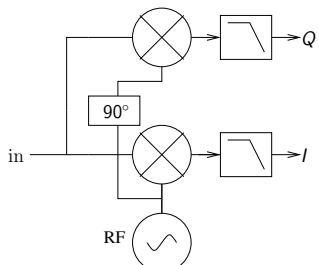
$$\propto A_1 \cdot A_2 \cdot \cos(\varphi) \text{ si } \omega_1 = \omega_2$$

- coefficients **I**(dentité) et **Q**(uadrature) : $A \simeq \sqrt{I^2 + Q^2}$, $\varphi \simeq \arctan(Q/I)$
- un signal réel (porteuse modulée) est devenu complexe en sortie du démodulateur

Mélangeur I, Q

- mélanger pour transposer une fréquence :

$$A_1 \sin(\omega_1 t + \varphi) \cdot A_2 \cos(\omega_2 t) \propto A_1 \cdot A_2 \cdot \sin(\varphi) \text{ si } \omega_1 = \omega_2$$

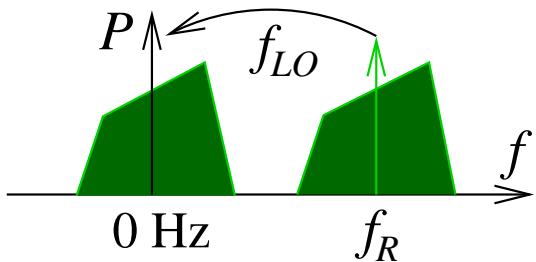


$$A_1 \sin(\omega_1 t + \varphi) \cdot A_2 \sin(\omega_2 t) \propto A_1 \cdot A_2 \cdot \cos(\varphi) \text{ si } \omega_1 = \omega_2$$

- coefficients **I**(dentité) et **Q**(uadrature) : $A \simeq \sqrt{I^2 + Q^2}$, $\varphi \simeq \arctan(Q/I)$
- un signal réel (porteuse modulée) est devenu complexe en sortie du démodulateur

Représentation d'un signal en bande de base

- Un signal (réel) en bande radiofréquence a un certain spectre qui n'a pas de raison d'être symétrique autour de sa porteuse f_R
- Ce signal est transposé par un mélangeur en bande de base (centré sur 0 Hz) : si mélangeur réel, le résultat est réel et son spectre est à symétrie hermitienne (symétrique-conjugué : $X(-\omega) = X^*(\omega) \Rightarrow |X(-\omega)| = |X(\omega)|$)
- Afin de conserver l'information initiale, il faut un spectre asymétrique que seul un complexe peut représenter
- En radio logicielle, on ne parlera jamais de f_R mais toujours du signal (complexe) en bande de base



- Noter que si on a un signal réel, il est possible de le rendre **analytique** par **transformée de Hilbert** (passe la partie négative du spectre à 0 = **crée une partie imaginaire en quadrature de la partie réelle**) pour extraire module ($|\cdot|$) et phase (atan())

Principe du récepteur : coefficients I, Q

Pourquoi traitons nous des nombres complexes ?

- un signal périodique est caractérisé par 3 grandeurs :

$$s(t) = \underbrace{A(t)}_{AM} \cdot \cos\left(\underbrace{\omega(t)}_{FM} \cdot t + \underbrace{\varphi(t)}_{PM} \right)$$

- puissance et {fréquence,phase} ne sauraient être extraites d'une unique mesure
- transposition par mélangeur de l'oscillateur local et sa copie en quadrature ($\cos(x + 90^\circ) = \sin$)
- Problème : la quadrature n'est pas idéale et les gains sont différents

$$\begin{cases} I_{expe} = A \cdot \cos(\omega t + \varphi) \\ Q_{expe} = (A + \varepsilon) \cdot \sin(\omega t) \end{cases} \quad v.s. \quad \begin{cases} I_{ideal} = A \cdot \cos(\omega t) \\ Q_{ideal} = A \cdot \sin(\omega t) \end{cases}$$

- $I_{expe} = A\cos(\omega t)\cos(\varphi) - A\sin(\omega t)\sin(\varphi) = I_{ideal}\cos(\varphi) - Q_{ideal}\sin(\varphi)$
- $Q_{expe} = (1 + \varepsilon/A)Q_{ideal}$: combinaisons linéaires des I et Q

Analogique v.s numérique : I, Q [2]

MPC-S1 **Sentinel-1**

Ref
MPC Nom: DI-MPC-IPFDPM
MPC Ref: MPC-0307
Issue/Revision: 2/2
Date: 07/06/2019

Sentinel-1 Level 1 Detailed Algorithm Definition

4.1 Raw Data Analysis

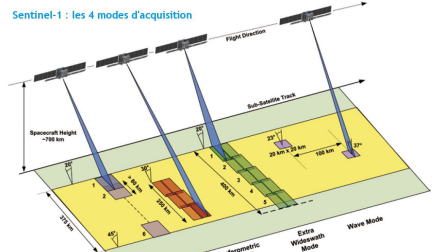
Raw data analysis is required in order to perform corrections of the I and Q channels of the raw signal data. The classical raw data correction (applied for instance in the case of ENVISAT-ASAR and RADARSAT-2) involves (see also Section 9.2):

- I/Q bias removal
- I/Q gain imbalance correction
- I/Q non-orthogonality correction

For Sentinel-1 however, the instrument's receive module performs the demodulation in the digital domain, therefore the I/Q gain imbalance and I/Q non-orthogonality corrections are no longer necessary.

The raw data analysis necessary for the raw data correction of ASAR data is defined in [R-6]. Since the IPF also supports the processing of ASAR data, for completeness, the ASAR raw data analysis scheme is reproduced in this section.

Even though for Sentinel-1 the I/Q gain imbalance and the I/Q non-orthogonality corrections are not necessary, they will be made available optionally, using configuration input parameters. Irrespective to the correction flag though, the Raw



Analogique v.s numérique : I, Q [2]

MPC-S1 **Sentinel-1**

Ref
MPC Nom: DI-MPC-IPFDPM
MPC Ref: MPC-0307
Issue/Revision: 2/2
Date: 07/06/2019

Sentinel-1 Level 1 Detailed Algorithm Definition

4.1 Raw Data Analysis

Raw data analysis is required in order to perform corrections of the I and Q channels of the raw signal data. The classical raw data correction (applied for instance in the case of ENVISAT-ASAR and RADARSAT-2) involves (see also Section 9.2):

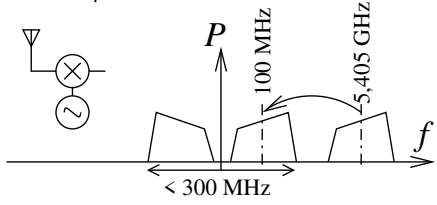
- I/Q bias removal
- I/Q gain imbalance correction
- I/Q non-orthogonality correction

For Sentinel-1 however, the instrument's receive module performs the demodulation in the digital domain, therefore the I/Q gain imbalance and I/Q non-orthogonality corrections are no longer necessary.

The raw data analysis necessary for the raw data correction of ASAR data is defined in [R-6]. Since the IPF also supports the processing of ASAR data, for completeness, the ASAR raw data analysis scheme is reproduced in this section.

Even though for Sentinel-1 the I/Q gain imbalance and the I/Q non-orthogonality corrections are not necessary, they will be made available optionally, using configuration input parameters. Irrespective to the correction flag though, the Raw

- Première transposition radiofréquence : 5,405 GHz → 100 MHz par mélange avec oscillateur local à 5,305 MHz (mélangeur unique matériel, échantillonnage du signal réel)
- Numérisation à 300 MS/s : spectre ±150 MHz
- Seconde transposition numérique "parfaite" pour ramener signal en bande de base : `t=[0:longueur-1]/fs; lo=exp(j*2*pi*freq*t); res=signal.*lo;`
- `lo` numérique est "parfait" en quadrature et amplitude



Analogique v.s numérique : I, Q [2]

Sentinel-1 Level 1 Detailed Algorithm Definition

4.1 Raw Data Analysis

Raw data analysis is required in order to perform corrections of the I and Q channels of the raw signal data. The classical raw data correction (applied for instance in the case of ENVISAT-ASAR and RADARSAT-2) involves (see also Section 9.2):

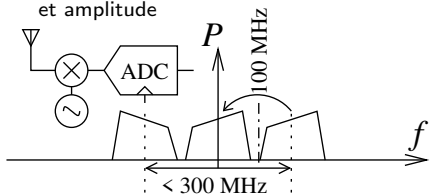
- I/Q bias removal
- I/Q gain imbalance correction
- I/Q non-orthogonality correction

For Sentinel-1 however, the instrument's receive module performs the demodulation in the digital domain, therefore the I/Q gain imbalance and I/Q non-orthogonality corrections are no longer necessary.

The raw data analysis necessary for the raw data correction of ASAR data is defined in [R-6]. Since the IPF also supports the processing of ASAR data, for completeness, the ASAR raw data analysis scheme is reproduced in this section.

Even though for Sentinel-1 the I/Q gain imbalance and the I/Q non-orthogonality corrections are not necessary, they will be made available optionally, using configuration input parameters. Irrespective to the correction flag though, the Raw

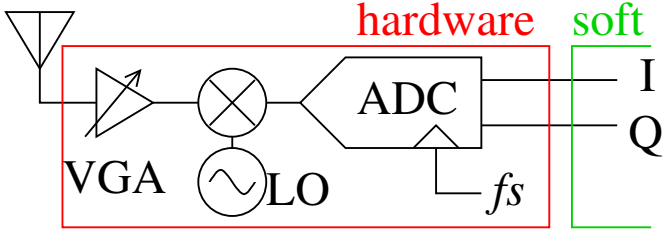
- Première transposition radiofréquence : 5,405 GHz $\rightarrow$ 100 MHz par mélange avec oscillateur local à 5,305 MHz (mélangeur unique matériel, échantillonnage du signal réel)
- Numérisation à 300 MS/s : spectre $\pm$150 MHz
- Seconde transposition numérique "parfaite" pour ramener signal en bande de base : `t=[0:longueur-1]/fs; lo=exp(j*2*pi*freq*t); res=signal.*lo;`
- `lo` numérique est "parfait" en quadrature et amplitude



Architecture d'un récepteur à conversion directe

Récepteur homodyne ou *zero-IF*

- LO transpose signal RF en bande de base : élimine la porteuse
- VGA : gain de réception pour détecter le signal sans le saturer
- f_s détermine la **bande passante** de mesure : la transformée de Fourier s'étend de $-f_s/2$ à $+f_s/2$
- mélange avec LO et sa copie en quadrature génère le complexe I+jQ

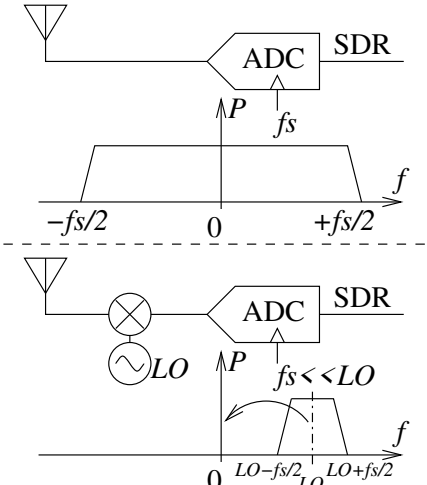


Débit de données :

- 4 octets/conversion (représentation flottante)
- 2 conversions/échantillon (I, Q)
- f_s conversions/seconde

Exemple : f_s=2 Méchantillons/s $\Rightarrow$ 16 MB/s ou 1 GB/minute

Architecture d'un récepteur à conversion directe

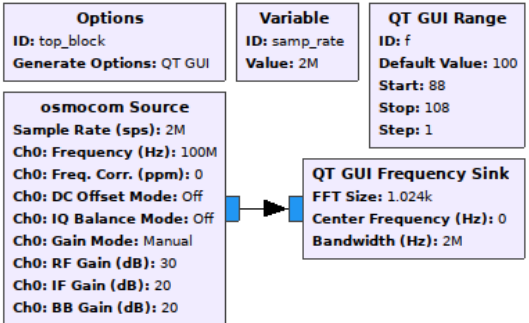


Transposition de fréquence par LO (*Local Oscillator*) avant échantillonnage : perte partielle de flexibilité par rapport à conversion directe mais

- bénéfice en terme de **débit** et taille de stockage (MB/s au lieu de 100-1000 MB/s)
- bénéfice en terme de **réjection** de sources puissantes interférant avec réception d'un signal faible (dynamique ADC : $20\log_{10}(\text{bit})=6\times\text{bits}=48$ dB pour 8 bits)
- ADC lents ont une meilleure **résolution** que ADC radiofréquences

Aspects logiciels : GNU Radio

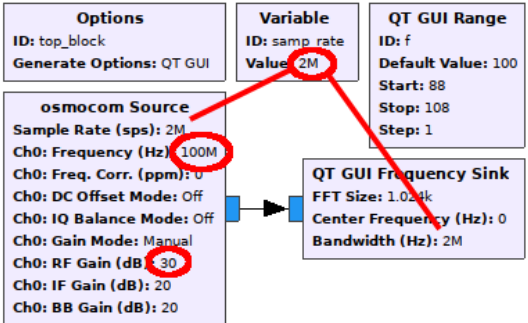
- Aspect logiciel : GNU Radio pour le traitement numérique du signal
- Logiciel libre disponible sous GNU/Linux, MacOS & MS-Windows `https://wiki.gnuradio.org/index.php?title=InstallingGR`
- Ensemble de bibliothèques C++ connectées par Python ou C++
- Générateur de code python : `gnuradio-companion`
- Opensource : comprendre et **ajouter ses propres blocs**




```
from PyQt5 import Qt
from gnuradio import gr
from gnuradio import qtgui
from gnuradio.filter import firdes
from gnuradio.qtgui import Range, RangeWidget
import osmosdr

class top_block(gr.top_block, Qt.QWidget):
    def __init__(self):
        gr.top_block.__init__(self, "Top Block")
        Qt.QWidget.__init__(self)
[...]
        # Variables
        self.samp_rate = samp_rate = 2e6
        self.f = f = 100
        # Blocks
        self._f_range = Range(88, 108, 1, 100, 200)
        self._f_win = RangeWidget(self._f_range, self.set_f, "f", "→
            ↪counter_slider", float)
        self.top_layout.addWidget(self._f_win)
        self.qtgui_freq_sink_x_0=qtgui.freq_sink_c(
          1024, #size
          firdes.WIN_BLACKMAN_hARRIS, #wintype
          0, samp_rate, "", 1 #fc, bw, name, inputs
        )
[... configuration du freq sink]
        self.osmosdr_src_0 = osmosdr.source( args="numchan=" + str(1) + →
            ↪" " + '' )
        self.osmosdr_src_0.set_sample_rate(samp_rate)
        self.osmosdr_src_0.set_center_freq(f*1e6, 0)
[...]
        self.connect((self.osmosdr_src_0, 0), (self.qtgui_freq_sink_x_0,→
            ↪ 0))
[...]
def main(top_block_cls=top_block, options=None):
[...]
```

Aspects logiciels : GNU Radio

- Aspect logiciel : GNU Radio pour le traitement numérique du signal
- Logiciel libre disponible sous GNU/Linux, MacOS & MS-Windows `https://wiki.gnuradio.org/index.php?title=InstallingGR`
- Ensemble de bibliothèques C++ connectées par Python ou C++
- Générateur de code python : `gnuradio-companion`
- Opensource : comprendre et **ajouter ses propres blocs**



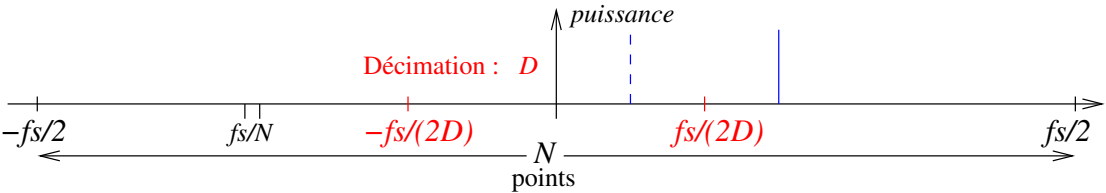

```
from PyQt5 import Qt
from gnuradio import gr
from gnuradio import qtgui
from gnuradio.filter import firdes
from gnuradio.qtgui import Range, RangeWidget
import osmosdr

class top_block(gr.top_block, Qt.QWidget):
    def __init__(self):
        gr.top_block.__init__(self, "Top Block")
        Qt.QWidget.__init__(self)
[...]
        # Variables
        self.samp_rate = samp_rate = 2e6
        self.f = f = 100
        # Blocks
        self._f_range = Range(88, 108, 1, 100, 200)
        self._f_win = RangeWidget(self._f_range, self.set_f, "f", "→
            ↪counter_slider", float)
        self.top_layout.addWidget(self._f_win)
        self.qtgui_freq_sink_x_0=qtgui.freq_sink_c(
          1024, #size
          firdes.WIN_BLACKMAN_hARRIS, #wintype
          0, samp_rate, "", 1 #fc, bw, name, inputs
        )
[... configuration du freq sink]
        self.osmosdr_src_0 = osmosdr.source( args="numchan=" + str(1) + →
            ↪" " + '' )
        self.osmosdr_src_0.set_sample_rate(samp_rate)
        self.osmosdr_src_0.set_center_freq(f*1e6, 0)
[...]
        self.connect((self.osmosdr_src_0, 0), (self.qtgui_freq_sink_x_0 ,→
            ↪ 0))
[...]
def main(top_block_cls=top_block, options=None):
[...]
```

Implémentation logicielle de quelques blocs de traitement

Analogique	Numérique
Redresseur (démod. AM)	valeur absolue
Filtre (passe bas)	FIR : $y_n = \sum b_k x_{n-k}$
Mélangeur	$s_n \leftarrow s_n \cdot sin(n/f_s),\ n \in \mathbb{N}$

Mais aussi des mises en œuvre d'algorithmes uniques au numérique (démodulateur FM)

Réduction du débit de données : la **décimation**. Décimer de D permet de faire une homothétie sur l'axe des abscisses

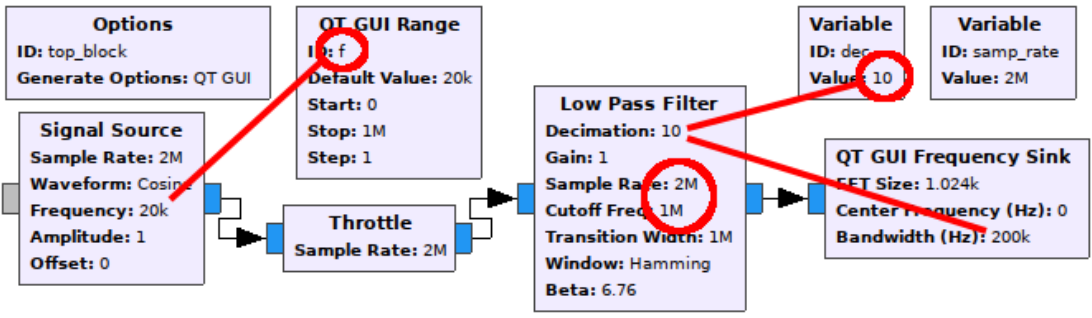


Tout traitement spectral **nécessite de maîtriser la fréquence d'échantillonnage** f_s/D : le spectre s'étend de $-f_s/(2D)$ à $+f_s/(2D)$

GNU Radio comme outil de prototypage rapide

Mise en œuvre simple des concepts de traitement numérique du signal en temps discret.

Exemple : repliement spectral

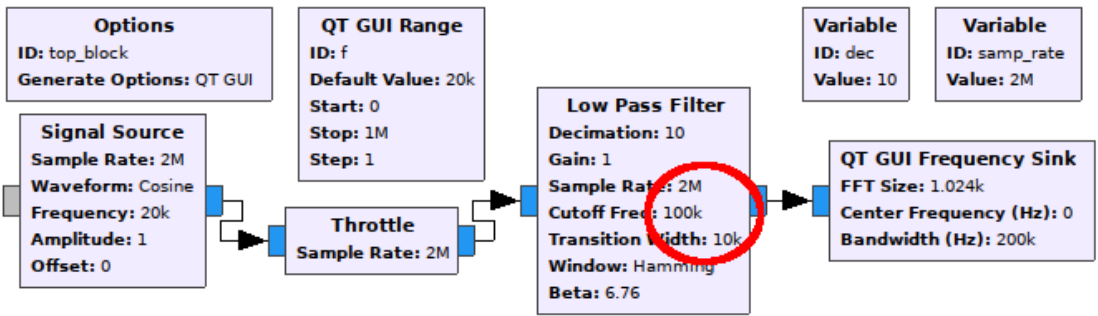


Si nous ne prenons pas soin de filtrer avant décimation, il y aura repliement dans la bande de base.

GNU Radio comme outil de prototypage rapide

Mise en œuvre simple des concepts de traitement numérique du signal en temps discret.

Exemple : repliement spectral



Si nous ne prenons pas soin de filtrer avant décimation, il y aura repliement dans la bande de base.

Résolution du filtre

- Transformée de Fourier bijective : N points temporels $\leftrightarrow$ N points fréquentiels
- Une transformée de Fourier sur N points échantillonnés à f_s présente une **résolution spectrale** de f_s/N
- Un filtre FIR sur N points présente la même résolution spectrale
- Demander une bande de transition δf excessivement fine se traduit par un **nombre énorme de coefficients**

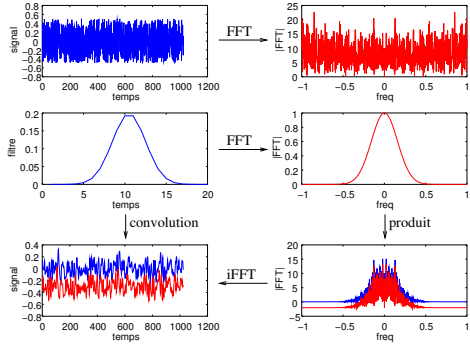
$$\delta f = f_s/N \Leftrightarrow N = f_s/\delta f$$

- $y_n = \sum b_k \cdot x_{n-k}$ nécessite N **multiplications** pour chacun des N coefficients : N^2
- $FFT(y) = FFT(b) \cdot FFT(x) \Leftrightarrow y = iFFT(FFT(b) \cdot FFT(x))$ avec "seulement" $N \log_2(N)$ opérations
- Si un filtre fin est nécessaire, cascader les filtres

Exemple : filtre de 30 Hz de transition sur un échantillonnage à 192 kHz

- $192000/30 = 6400$ coefficients ou $N \log_2(N) \simeq 81000$ opérations
- décimation de 50 (coupure à $f_s/100$ avec une largeur de $f_s/200$) : 200 coefficients pour 1500 opérations
- aprés décimation, $f_s/50 = 3840$ Hz et $3840/30 = 128$ coefficients, pour 900 opérations
- $1500 + 900 \ll 81000$

Tester avec `Filter Design Tool` sous GNU/Linux



Démonstration du théorème de convolution

$$\left.\begin{array}{l} conv(x,y)(\tau) = \int x(t) \cdot y(\tau - t) dt \\ TF(x)(f) = \int x(t) e^{j2\pi ft} dt \end{array}\right\} \Rightarrow$$

$$TF(conv(x,y)) = \iint x(t) y(\tau - t) dt e^{j2\pi f\tau} d\tau$$

on définit $s = \tau - t \Rightarrow ds = d\tau$ et

$$TF(conv(x,y)) = \iint x(t) y(s) dt e^{j2\pi f(t+s)} ds$$

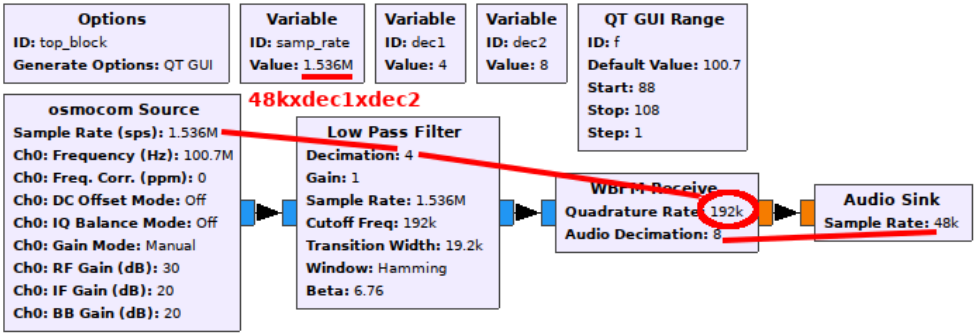
$$\Leftrightarrow TF(conv(x,y)) = \int x(t) e^{j2\pi ft} dt \times \int y(s) e^{j2\pi fs} ds$$

$$\Leftrightarrow TF(conv(x,y)) = TF(x) \cdot TF(y)$$

Importance de la gestion du flux de données

- Plus on avance dans la chaîne de traitement, plus la bande passante se réduit
- $\Rightarrow$ décimation pour réduire le débit de données à traiter
- on ne peut pas créer d'information au cours du traitement, uniquement en perdre (porteuse + modulation $\rightarrow$ bande de base $\rightarrow$ contenu informatif)
- exemple : FM commerciale nécessite 250 kHz de bande pour encoder du son à 16 kHz (gauche + droite) + RDS

Réduire le débit de données pour réduire la charge sur le processeur, ou permettre des calculs de plus en plus complexes.



Conclusion sur la gestion du flux de données

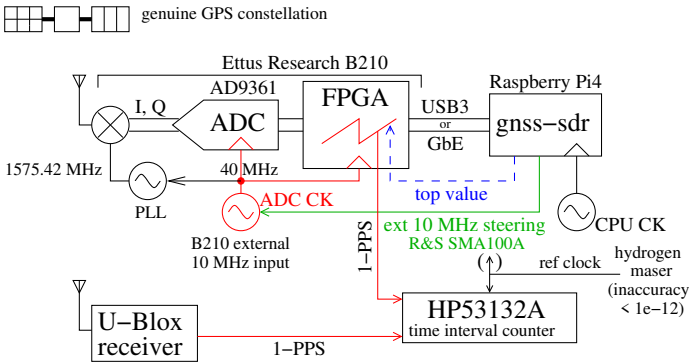
- Mémoriser l'évolution du débit de données au cours du traitement
- Ajuster le débit de données de la source si la cible impose sa fréquence d'échantillonnage à l'arrivée
- Filtre passe bas pour empêcher le repliement spectral
- Penser aux variables pour paramétrer les facteurs de décimation, afin de garantir la cohérence + faciliter le test de diverses valeurs (e.g. bande optimale pour démoduler FM)

Démonstrations[3] **:**

1. source réelle → oscilloscope et analyseur de spectre
2. source complexe → oscilloscope et analyseur de spectre, concept de tension imaginaire et fréquence négative
3. décimation et repliement spectral, périodicité du spectre (signal réel et signal complexe)
4. filtrage avant décimation
5. bande de transition et nombre de coefficients, interaction avec le script Python

3. `https://gitlab.com/gnuradio_book/flowcharts`

Datation des échantillons

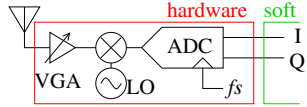


D. Rabus, G. Goavec-Merou, G. Cabodevila, F. Meyer, J.-M Friedt, *Generating a timing information (1-PPS) from a software defined radio decoding of GPS signals*, Proc. Joint IEEE EFTF/IFCS (2021)

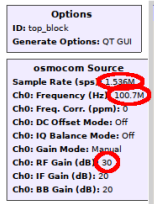
- En SDR, la seule **date connue** est a conversion analogique-numérique ...
- ... tout transfert ultérieur peut être asynchrone (USB, Ethernet ...)
- Si aucun échantillon n'est perdu : $t_n = n/f_s$
- Problème cependant en boucle fermée quand la lattence dans la rétroaction n'est pas contrôlée
- Exemple de l'oscillateur asservi sur GNSS : seule l'horloge du FPGA contrôlant l'acquisition importe, les autres horloges sont libres (tant que le traitement prend moins de temps que l'acquisition)

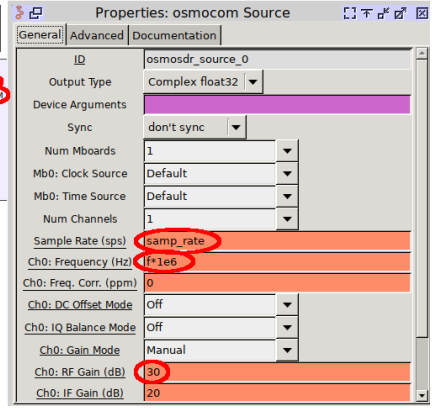
Interface matérielle : récepteur TNT RTL-SDR

- Détournement d'un récepteur de télévision numérique terrestre pour la réception de signaux radiofréquences



- R820T(2) (RF)+RTL2832U (ADC+USB) pour moins de 10 $/pièce
- fréquence d'échantillonnage 1 MHz $\leq f_s \leq$ 2, 4 MHz
- porteuse $\in$ [50 : 1600] MHz
- gain variable de l'étage radiofréquence : [0 : 39] dB
- source GNU Radio : `gr-osmosdr` ou `soapysdr-module-osmosdr`





Applications

Reconfiguration du logiciel : une même plateforme matérielle permet d'appréhender une multitude de modes de communication

Mode	modulation	porteuse (MHz)	bande passante (kHz)
WBFM	FM analogique	88–108	120
DAB	OFDM (FM num)	175–250	1537
POCSAG	FSK	466	9
ADS-B	ASK	1090	50
XE1203	FSK	434, 868	dépend du débit
ACARS	AM	131,725	5
Iridium	PSK	1620	31,5
GPS	BPSK	1575,42	1000
NOOA (APT)	analog	137	34
Meteor M2 (LRPT)	QPSK	137,9	120
RADAR passif	bruit/xcorr		max ($\Delta R = c_0/(2B)$)

Tous[4] ces modes sont accessibles avec un récepteur échantillonnant à $fs \leq 2,4$ MS/s ($f_s \geq 2BW$)

4. https://www.sigidwiki.com/wiki/Signal_Identification_Guide

Bibliographie 1/2

1. J.-M. Friedt, H. Boeglen, *Communication Systems Engineering with GNU Radio : A Hands-on Approach*, Wiley (2024)
2. T. Collins *& al.*, *Software-Defined Radio for Engineers*, (2018) at `https://www.analog.com/en/education/education-library/software-defined-radio-for-engineers.html`
3. S.W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing, 2nd Ed* (1999) at `https://www.dspguide.com/pdfbook.htm`
4. T. McDermott, *Wireless Digital Communications : Design and Theory*, Tucson Amateur Packet Radio Corporation – TAPR (1997)
5. J.G. Proakis, D.K. Manolakis, *Digital Signal Processing*, Prentice Hall (2006)
6. R.G. Lyons, *Understanding Digital Signal Processing*, Prentice Hall (2004)
7. R.G. Lyons (Ed.), *Streamlining Digital Signal Processing, 2nd Ed*, Wiley/IEEE Press (2012)
8. A.V. Oppenheim, R.W. Schafer, *Discrete-Time Signal Processing (3rd Edition)*, Prentice-Hall Signal Processing Series (2009), and videos of his lectures at `ocw.mit.edu/resources/res-6-007-signals-and-systems-spring-2011/video-lectures/lecture-1-introduction/`
9. K. Borre, D.M. Akos, N. Bertelsen, *A Software-Defined GPS and Galileo Receiver : A Single-Frequency Approach*, Birkhäuser (2007)
10. K. Borre, I. Fernández-Hernández, J.A. López-Salcedo, M.Z.H. Bhuiyan, *GNSS Software Receivers*, Cambridge Univ. Press (2023)
11. E.D. Kaplan, C. Hegarty, *Understanding GPS : Principles and Applications, 2nd Ed.*, Artech House (2005)
12. C.A. Balanis, *Antenna Theory, Analysis and Design*, Wiley Interscience (2005)

Bibliographie 2/2

Sur le Web :

1. Principles of Digital Communications course at `ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-450-principles-of-digital-communications-i-fall-2006/video-lectures/`
2. Balint videos at `www.youtube.com/playlist?list=PL618122BD66C8B3C4` and `www.youtube.com/watch?v=1bgC3AjCnA4`
3. Tom Rondeau's presentations, for example `gnuradio.org/redmine/projects/gnuradio/wiki/Guided_Tutorial_PSK_Demodulation` and `www.youtube.com/watch?v=_hGNT1w-jig`
4. M. Lichtman, *PySDR : A Guide to SDR and DSP using Python* at `https://pysdr.org` (accédé 01/2025 [5])
5. *Learning DSP Illustrated* at `https://dspillustrations.com/pages/index.html` (accédé 01/2025)
6. FOSDEM Radio Devroom (e.g. `https://fosdem.org/2025/schedule/track/radio/`)

5. presenté au FOSDEM2021 à `https://fosdem.org/2021/schedule/event/fsr_pysdr_guide_to_sdr_and_dsp_using_python/`