TP 3: Conception d'un émetteur sans fil

```
close('all'); clear; clc;
```

Partie A : Conception de l'émetteur

L'objectif de cette partie est de commencer la réalisation d'un système de communication numérique, en commençant par l'émetteur (figure 1) d'un appareil portable. Le système utilise une modulation d'amplitude M-aire (M-ASK) avec un débit de symboles R_s = 3,84 MHz, et un filtre de mise en forme d'impulsions rectangulaires. La bande de fonctionnement de cet émetteur est la bande I (1920 MHz-1980 MHz) définie par la norme 3GPP (émission UE, réception station de base). La fréquence porteuse doit alors être ajustable pour prendre des valeurs à l'intérieur de cette bande. Pour le projet final, le même système sera mis en œuvre mais avec une modulation différente (selon votre avancement).

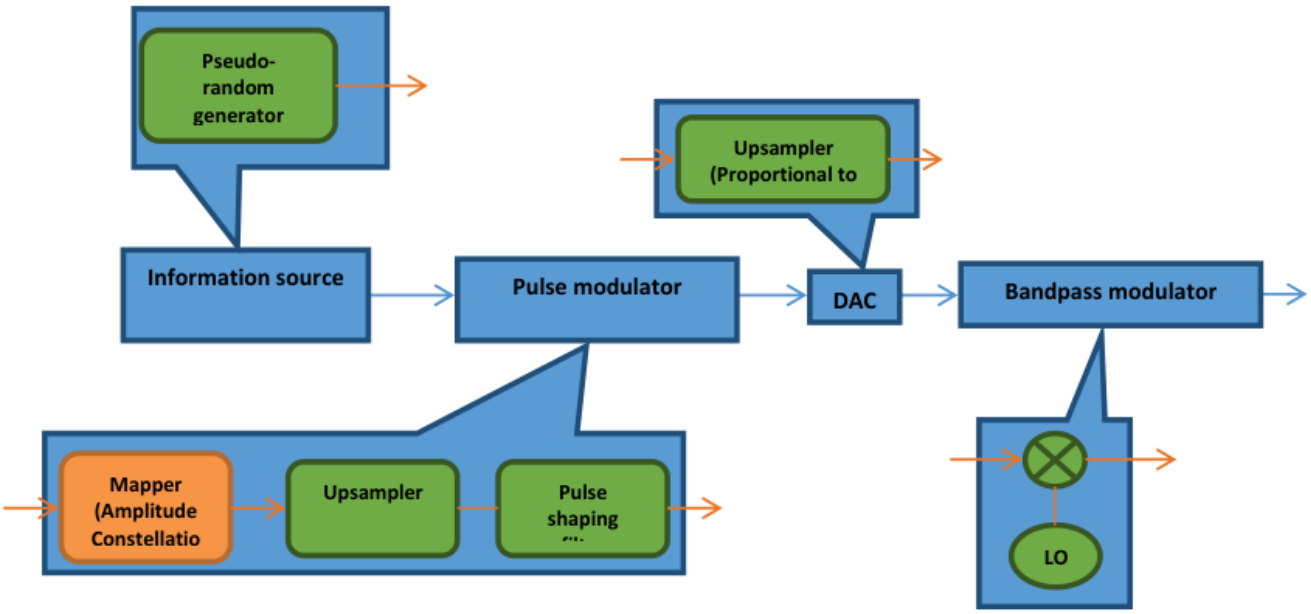


Figure 1. Emetteur

Source d'information

1. Source d'information :

Pour modéliser la source d'information aléatoire, nous utiliserons un générateur d'entiers pseudo-aléatoires remplaçant le bloc de formatage qui pourra être inclus ultérieurement. Nous voulons mettre en œuvre un système M-aire, et donc il y a M symboles possibles.

Les entiers générés (en utilisant la fonction ***randi*** de Matlab) représentent des numéros de symboles (0,1,..., M-1). La fonction ***randi()*** peut générer des entiers aléatoires dans un intervalle arbitraire, avec une distribution uniforme. Pour notre simulation, nous voulons générer un nombre, NbSymb, de symboles (par exemple, NbSymb=20 puis 500).

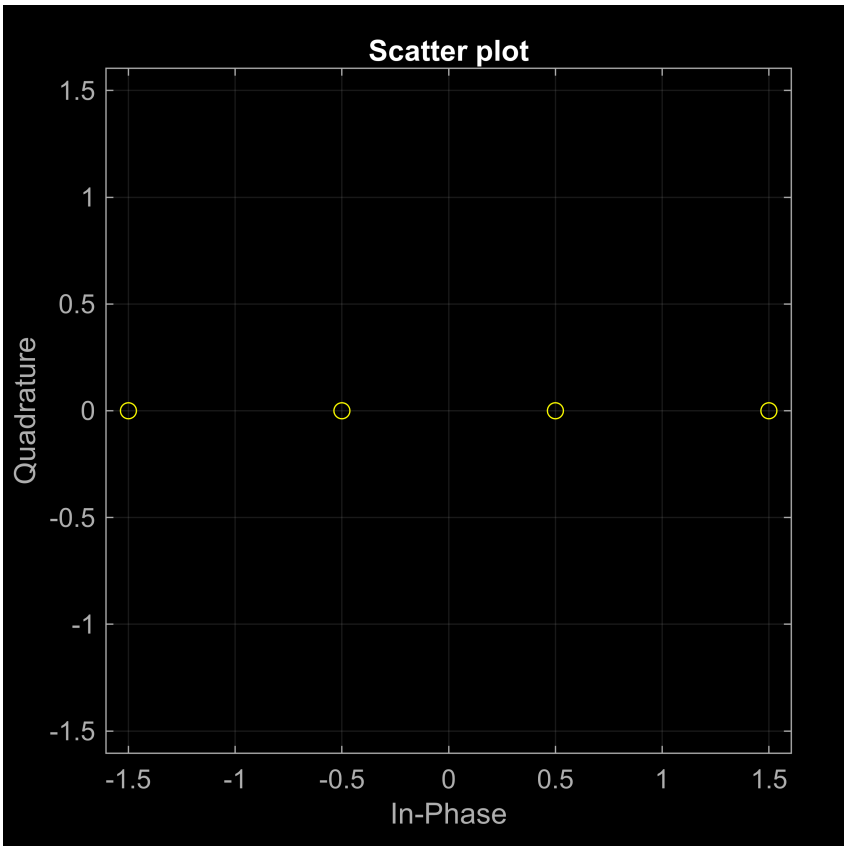
```
M = 4;
Ns = 16;
symbols = randi(M, [1, Ns]);
T = Transmitter(symbols, M);
```

Modulateur en bande de base

a. *Mappeur :*

Le rôle du mappeur est d'attribuer une amplitude prédéfinie pour chaque symbole. Pour déterminer la constellation de cette modulation (valeurs d'amplitude possibles), nous définissons un paramètre V_d (unités Volts) qui est égal à la différence de tension entre deux amplitudes de symboles consécutives. Nous devons donc d'abord déterminer la constellation de la modulation M-ASK à l'aide de ce paramètre. Le signal de sortie du Mapper, est un train d'impulsions dont la période est égale à la durée symbole T_s.

```
plt_con = scatterplot(T.mapping.constellation);
plt_con.Children(2).Children.Marker = 'o';
grid on;
```



b. *Suréchantillonneur :*

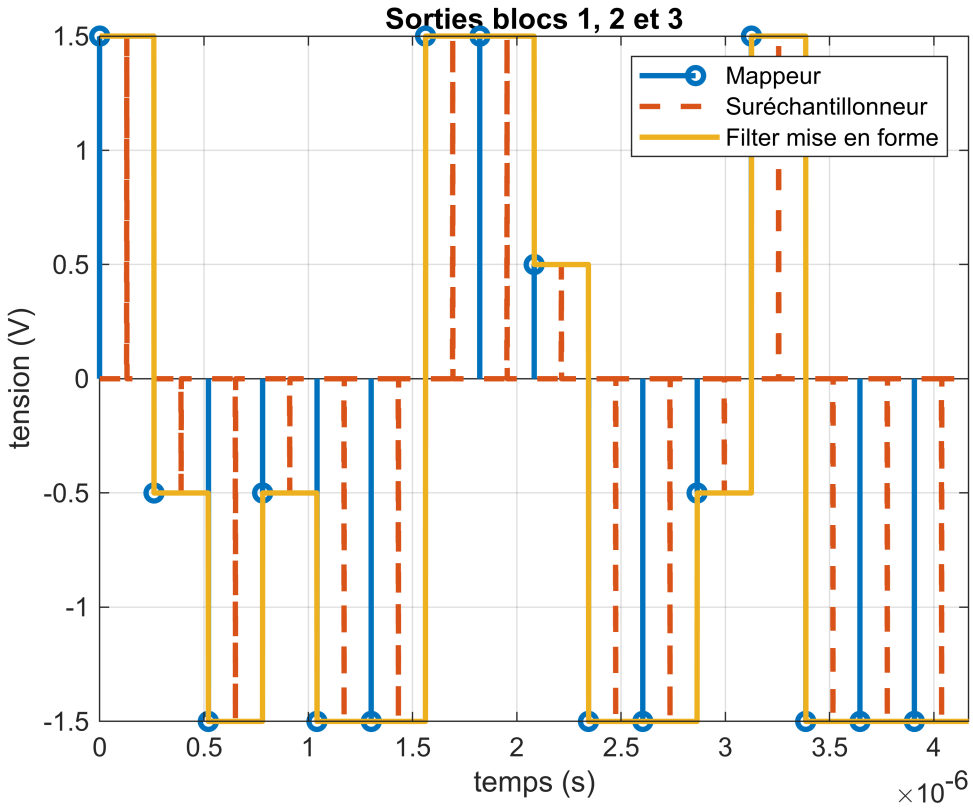
Le suréchantillonneur augmente la fréquence d'échantillonnage de ce signal jusqu'au taux d'échantillonnage du filtre numérique de mise en forme des impulsions. Ce suréchantillonneur définit le nombre d'échantillons par symbole, il est généralement choisi comme une puissance de 2 (2^n échantillons par symbole). Pour générer le signal de sortie de ce bloc, nous pouvons utiliser la fonction ***upsample*** de Matlab.

c. *Filtre de mise en forme :*

Le troisième sous-bloc est le filtre de mise en forme des impulsions, qui doit être une impulsion rectangulaire (dans un premier temps). Pour concevoir ce filtre, vous pouvez utiliser la fonction ***window*** de Matlab et ensuite ajouter des échantillons à zéro pour obtenir une forme complète. Vous devez définir le nombre d'échantillons nécessaires pour ce filtre étant donné que l'impulsion a une durée de T_s. Pour visualiser les réponses impulsionnelles et fréquentielles de ce filtre, nous pouvons utiliser l'outil ***fvtool***.

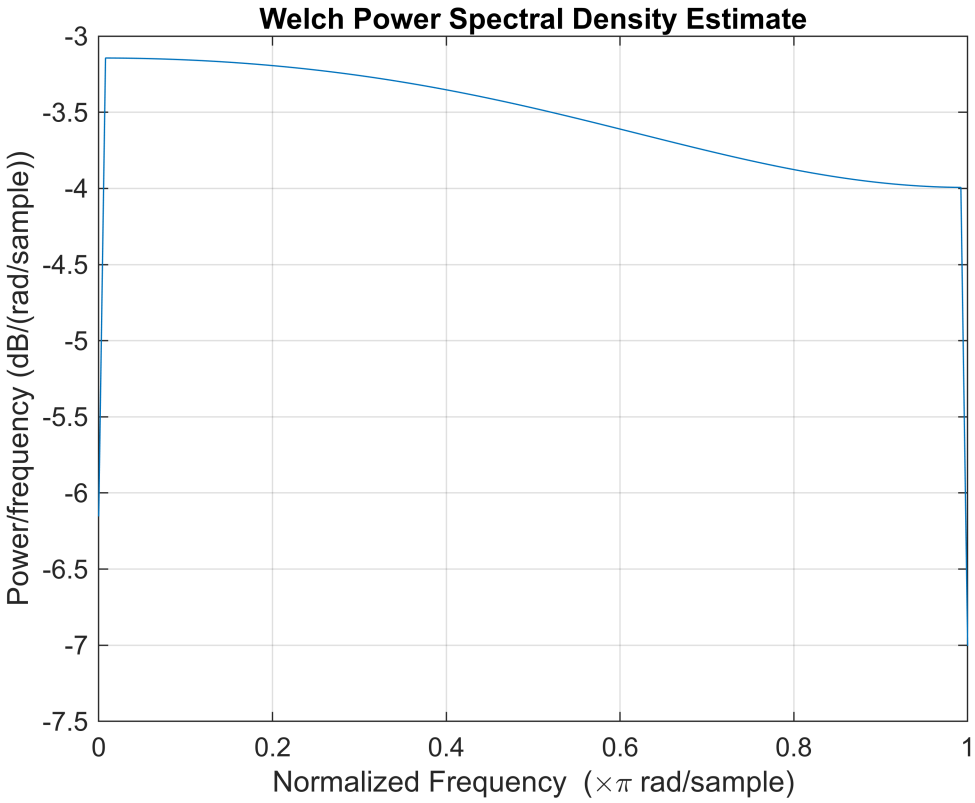
Pour générer le signal de sortie, vous pouvez utiliser la fonction ***conv*** de Matlab, pour calculer la convolution entre le signal d'entrée et l'impulsion.

```
fig1 = figure();
ax1 = axes(fig1);
plt_map = stem(T.mapping.time, T.mapping.signal);
plt_map.LineWidth = 2;
hold on;
plt_ups = plot(T.upsampling.time, T.upsampling.signal);
plt_ups.LineWidth = 2;
plt_ups.LineStyle = '--';
plt_ps = plot(T.pulse_shaping.time, T.pulse_shaping.signal);
plt_ps.LineWidth = 2;
ax1.XLim = [0, T.Ts * T.Ns];
ax1.YLim = [min(T.mapping.constellation), max(T.mapping.constellation)];
grid on;
hold off;
title('Sorties blocs 1, 2 et 3');
xlabel('temps (s)');
ylabel('tension (V)');
legend('Mappeur', 'Suréchantillonneur', 'Filter mise en forme');
```

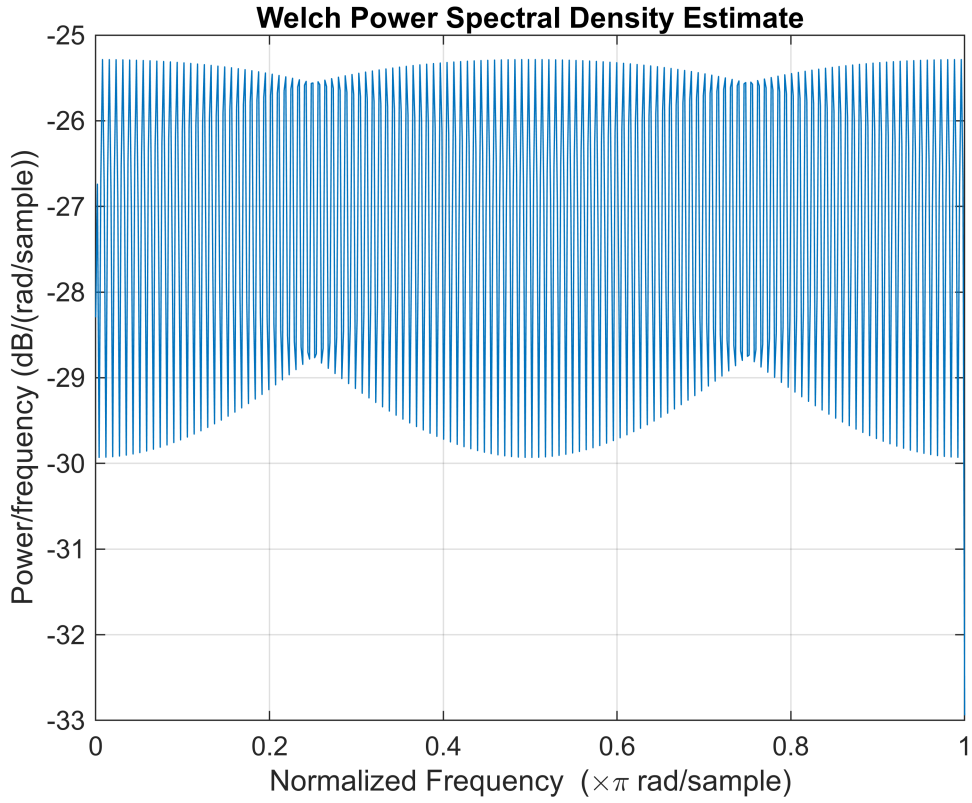


Tracez sur la même figure, avec l'axe de temps approprié pour chaque signal, les signaux de sortie des trois blocs. Tracez également la densité spectrale de puissance de ces signaux (nous pouvons utiliser la classe ***psd*** et la méthode ***welch*** pour calculer la densité spectrale de puissance. Créez d'abord un objet de spectre en utilisant ***spectrum.welch***).

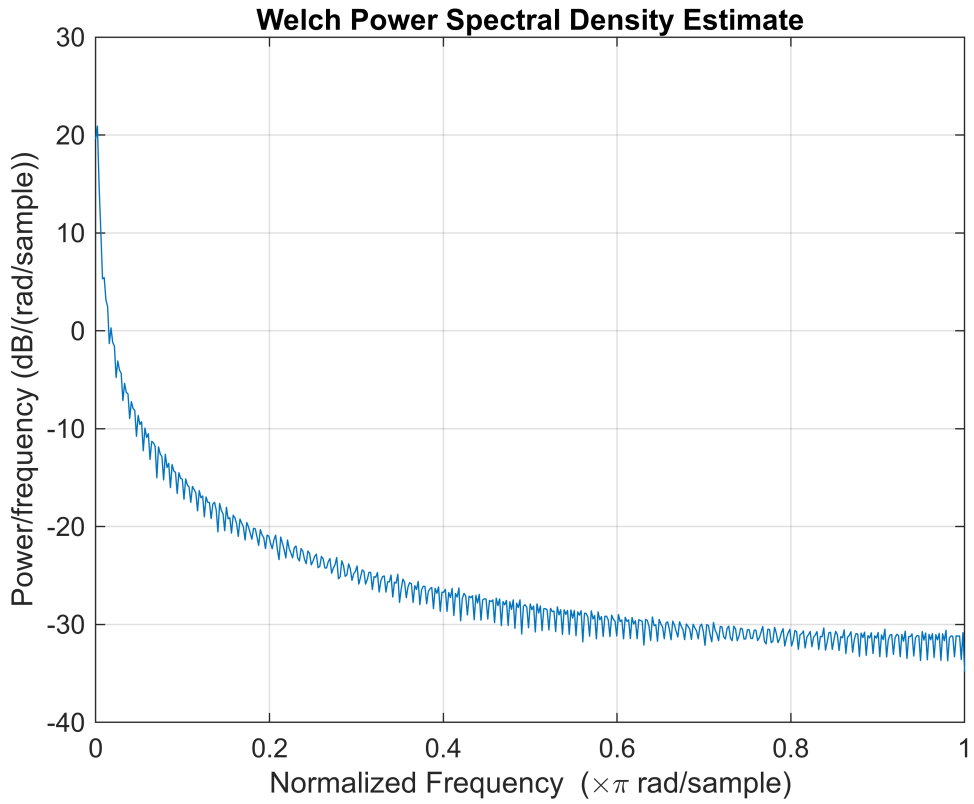
```
% psd.welch() n'existe plus, utilisez plutôt pwelch()
pwelch(T.mapping.signal)
```




```
pwelch(T.upsampling.signal)
```



```
pwelch(T.pulse_shaping.signal)
```



CNA et Modulateur en bande transposée

3. CNA :

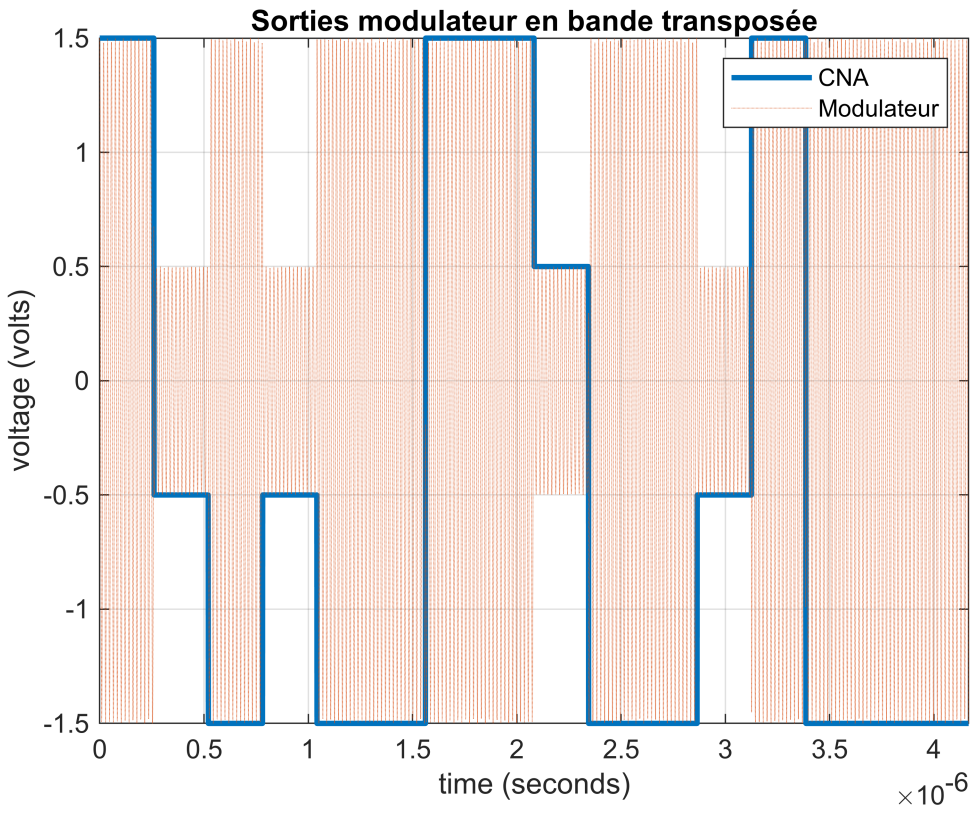
Pour modéliser la conversion du numérique à l'analogique après le filtrage numérique, nous utiliserons un suréchantillonneur dont la fréquence est relativement très élevée par rapport au taux d'échantillonnage du filtre numérique. La fréquence d'échantillonnage à la sortie de ce bloc doit donc être suffisamment élevée pour numériser correctement la porteuse. Nous pouvons donc choisir un facteur de suréchantillonnage pour avoir une nouvelle fréquence d'échantillonnage multiple de f_c. La fonction ***interp1*** peut être utilisée pour suréchantillonner et interpoler en même temps.

4. Modulateur en bande transposée :

Pour modéliser ce bloc, nous devons générer une porteuse (cosinus ou sinus), de fréquence égale à f_c. Cette porteuse est la sortie de l'oscillateur local (OL). Ensuite, comme le montre la figure 1, le signal en bande transposée peut être généré en multipliant simplement la sortie du CNA par la porteuse.

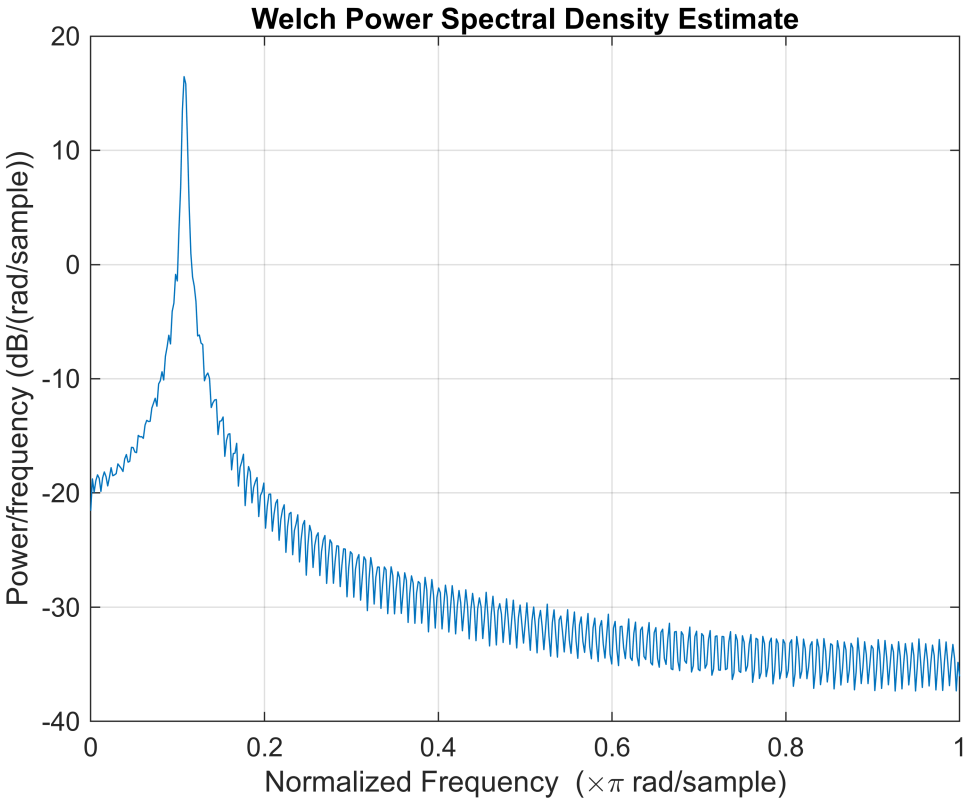
```
fig2 = figure;
ax2 = axes(fig2);
plt_dac = plot(T.dac.time, T.dac.signal);
plt_dac.LineWidth = 2;
ax2.XLim = [0, T.Ts * T.Ns];
```

```
ax2.YLim = [min(T.mapping.constellation), max(T.mapping.constellation)];
hold on;
grid on;
plt_mod = plot(T.modulation.time, T.modulation.signal);
plt_mod.LineWidth = 0.25;
plt_mod.LineStyle = ':';
title('Sorties modulateur en bande transposée');
xlabel('time (seconds)');
ylabel('voltage (volts)');
hold off;
legend('CNA', 'Modulateur');
```



Tracez sur la même figure le signal de sortie de ce bloc, et le signal de sortie du DAC. Tracez le PSD du signal de sortie de ce bloc. Commentez les résultats.

```
pwelch(T.modulation.signal)
```


Welch Power Spectral Density Estimate
Power/frequency (dB/(rad/sample))
Normalized Frequency (×π rad/sample)
20
10
0
-10
-20
-30
-40
0
0.2
0.4
0.6
0.8
1