TP6 - Modulation de largeur d'impulsion

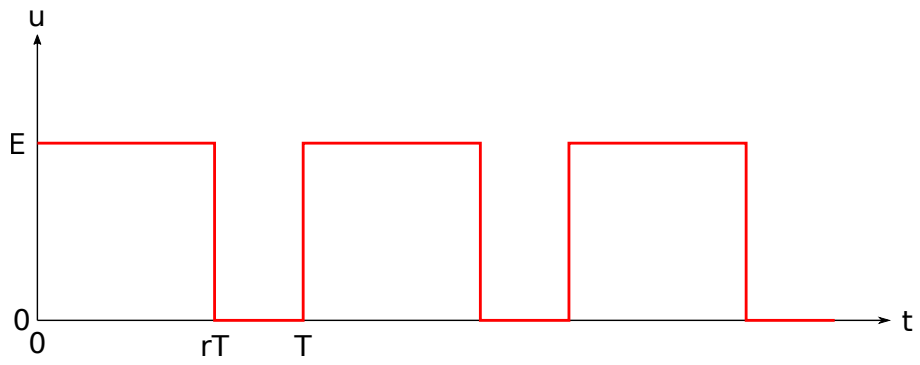
1. Introduction

La modulation de largeur d'impulsion (MLI) est une technique utilisée en électronique de puissance pour convertir une tension continue en tension continue (conversion DC-DC) ou une tension continue en tension alternative (conversion DC-AC). Elle est ausi utilisée dans certains amplificateurs audio. La modulation de largeur d'impulsion est aussi nommée PWM (Pulse Width Modulation). L'objectif de ces travaux pratiques est de réaliser une MLI au moyen d'un circuit analogique. Il est aussi possible de générer un signal MLI au moyen d'un microcontrôleur.
Matériel :

- ▷ Plaque d'essai avec fils.
- ▷ Deux ALI TL081.
- ▷ Deux résistances de $1\,\mathrm{k}\Omega$.
- ▷ Condensateurs de $100\,\mathrm{nF}$ et $1\,\mu\mathrm{F}$.
- ▷ Arduino MEGA.

2. Principe

La MLI repose sur la génération d'une tension carrée à rapport cyclique variable, définie sur la figure ci-dessous :



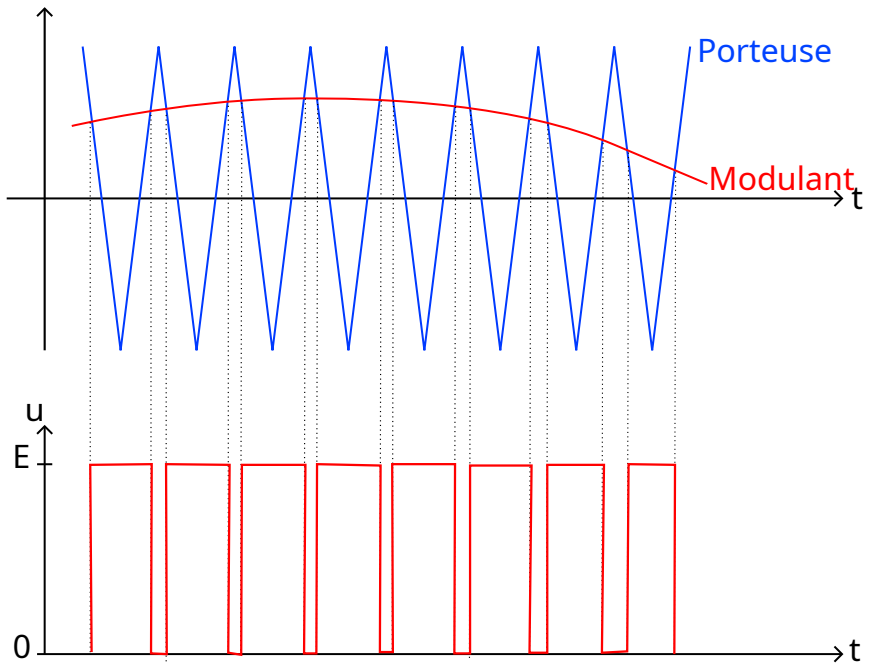
La tension est périodique, de période T, égale à E pendant une durée rT, à 0 pendant $(1-r)T$. Le paramètre r est le *rapport cyclique*.

[1] Exprimer la valeur moyenne de $u(t)$ en fonction de E et r.

[2] Un filtre RC permet d'extraire la valeur moyenne. Faire le schéma de ce filtre et exprimer sa fréquence de coupure. Comment faut-il choisir cette fréquence de coupure en fonction de la période T.

En électronique de puissance, la tension u(t) est obtenue à partir d'une tension continue E par un circuit à transistors nommé *hacheur*. Le hacheur est suivi d'un filtre passe-bas. On procède

ainsi à la conversion d'une tension continue E en une tension continue ajustable avec le rapport cyclique. Ce principe est utilisé dans les alimentations à découpage (avec un filtre RL). Un système de régulation ajuste automatiquement le rapport cyclique pour que la tension de sortie reste constamment égale à la valeur de consigne, quel que soit le courant débité. Lorsqu'on alimente une charge inductive, par exemple un moteur électrique, il n'est pas nécessaire d'utiliser un filtre.
La modulation de largeur d'impulsion proprement dite consiste à faire varier très lentement le rapport cyclique (à une fréquence beaucoup plus faible que $1/T$) de manière à obtenir après filtrage passe-bas une forme d'onde quelconque, par exemple sinusoïdale. La MLI est par exemple utilisée dans les onduleurs qui permettent de convertir la tension continue d'une batterie en tension alternative sinusoïdale.
Pour générer un signal MLI, on dispose *d'une porteuse*, qui est un signal triangulaire de période T et d'un signal modulant, que l'on prendra sinusoïdal dans un premier temps. La fréquence du signal modulant doit être beaucoup plus faible que celle de la porteuse (au moins un facteur 100) et elle doit être si possible dans la bande passante du filtre passe-bas. L'amplitude du signal modulant doit être inférieure à celle de la porteuse. La génération du signal MLI se fait par simple comparaison de la porteuse avec le signal modulant, comme le montre la figure suivante :

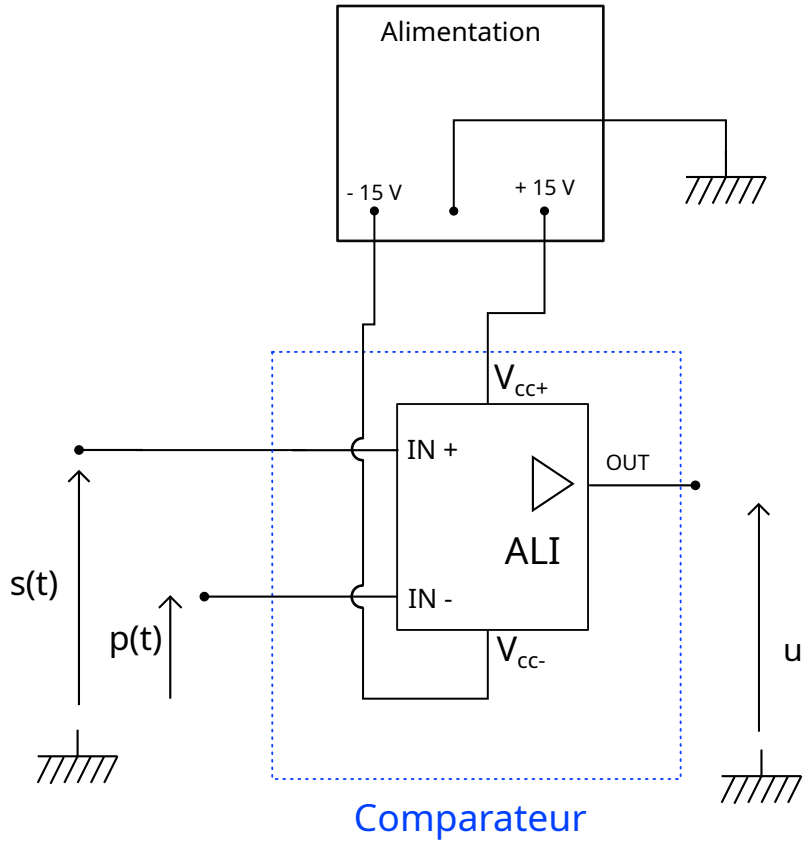


Soit $p(t)$ la porteuse et $s(t)$ le signal modulant. Le signal $u(t)$ est généré de la manière suivante :

$$u(t) = 0 \text{ si } s(t) < p(t)$$
$$u(t) = E \text{ si } s(t) \geq p(t)$$

3. Modulateur analogique

La modulation analogique repose sur l'utilisation d'un *comparateur*, réalisé au moyen d'un ALI fonctionnant en régime non linéaire :



Cette configuration de l'ALI ne comporte pas de rétroaction de la sortie vers l'entrée inverseuse (IN-). En conséquence, il fonctionne toujours en régime non linéaire, c'est-à-dire que la tension de sortie par rapport à la masse est soit égale à V_1 lorsque $V_+ > V_-$, soit égale à V_2 lorsque $V_+ < V_-$. La tension V_1 est proche de la tension d'alimentation positive : elle vaut environ 14 V. La tension V_2 est proche de la tension d'alimentation négative : elle vaut environ -14 V. V_1 et V_2 sont les *tensions de saturation*, respectivement positive et négative. Ce montage permet de générer un signal MLI mais la tension $u(t)$ permute entre V_2 et V_1, au lieu de 0 et E. Pour obtenir effectivement une tension $u(t)$ permuttant entre 0 et E, il faudrait utiliser un ALI spécialement conçu pour un usage en comparateur et l'alimenter avec une alimentation simple de tension E. Nous gardons ce montage avec un ALI d'usage général car il nous permettra d'obtenir par démodulation un signal alternatif similaire à $s(t)$. Notons cependant que pour piloter des transistors (en électronique de puissance), un signal MLI permuttant entre 0 et E (typiquement 5 V) est obligatoire.

[3] Réaliser le comparateur sur la plaque d'essai avec l'ALI TL081. Les signaux $s(t)$ et $q(t)$ sont délivrés par les voies A et B du générateur de signaux.
La fréquence de la porteuse est $10\,\mathrm{kHz}$ et son amplitude de crête à crête est $4\,\mathrm{V}$. Dans un premier temps, on étudie le fonctionnement du modulateur lorsque $s(t)$ est constant. Pour obtenir un signal constant avec le générateur de signaux, on utilise l'offset et on annule l'amplitude.

[4] Observer la tension $u(t)$ lorsque $s(t)$ est constant, égal à 1,0 V. En déduire les valeurs de V_1 et V_2.

[5] La tension en sortie du comparateur est-elle de forme carrée ? Vérifier la valeur du rapport cyclique.
Remarque : l'utilisation d'un ALI spécialisé pour le fonctionnement en régime saturé (nommé comparateur) permettrait d'obtenir des passages de V_1 à V_2 beaucoup plus rapides.

[6] On souhaite faire la numérisation de $u(t)$ avec la carte Sysam SP5 mais celle-ci n'accepte pas les tensions en dehors de l'intervalle [-10,10] V. Utiliser les deux résistances de $2\,\mathrm{k}\Omega$ pour diviser par deux la tension $u(t)$. Brancher la voie EA0.
On numérisera $u(t)$ (divisée par deux) au moyen du script suivant :

acquisition.py

```
import numpy as np
import matplotlib.pyplot as plt
import pycanum.main as pycan
can = pycan.Sysam("SP5")
Vmax = 10.0
can.config_entrees([0],[Vmax])
fe = 700e3
N = 2**18 # valeur maximale de N
T = N/fe
print("T = %f s"%T)
te = 1/fe
can.config_echantillon(te*1e6,N)
can.acquerir()
t=can.temps()[0]
u=can.entrees()[0]
can.fermer()
te=t[1]-t[0]
fe=1/te
print("fe = %f"%fe)
# modifier le nom du fichier fichier pour chaque expérience
np.save('signal-MLI-1.npy',np.array([t,u0]))
plt.figure()
plt.plot(t,u)
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('u (V)')
plt.xlim(0,1e-2) # échelle à choisir en fonction de la période du signal
plt.show()
```

[7] Justifier le choix de la fréquence d'échantillonnage pour numériser ce signal. Est-elle en théorie assez grande ?

[8] Quelle est la durée T du signal échantillonné ? Quel sera en conséquence la résolution fréquentielle du spectre ?

[9] Exporter le fichier NPY contenant le signal vers le Jupyter Hub, dans le dossier `partage/TP/TP6` et dans le sous-dossier `NOM1-NOM2`. Créer dans ce dossier un notebook intitulé `AnalyseSignaux`, dans lequel on écrira le code d'analyse spectrale de ce signal et des suivants.

[10] Après avoir importé les modules nécessaires, récupérer le signal de la manière suivante : `[t,x]=np.load('nom_fichier.npy')`. Faire le tracé de sa représentation temporelle (ne pas oublier les légendes et le bon choix des échelles).

[11] Calculer la TFD puis le spectre de la manière suivante :

```
from numpy.fft import fft
from scipy.signal import blackman
N = len(x)
te = t[1]-t[0]
x1=np.concatenate((x*blackman(N),np.zeros(6*N)))
N1 = len(x1)
T = N1*te
spectre = np.absolute(fft(x1))*2/N/0.42
freq = np.arange(N1)*1.0/T1
```

[12] Tracer le spectre du signal en échelle décibel.

[13] Analyser le même signal avec la fonction FFT de l'oscilloscope. Comparer les deux spectres.

[14] Le spectre obtenu est-il affecté par le repliement de bande ?

[15] On considère à présent un signal modulant sinusoïdal, de fréquence $10\,\mathrm{Hz}$ et d'amplitude $1{,}0\,\mathrm{V}$. Enregistrer le signal numérisé dans un fichier puis faire son analyse dans le notebook.

[16] Tracer le spectre complet puis tracer deux autres spectres : l'un pour la plage de fréquences allant de 0 à $100\,\mathrm{Hz}$, l'autre pour la plage allant de $9900\,\mathrm{Hz}$ à 10100. Faire de même avec la fonction FFT de l'oscilloscope.

[17] Comment le signal modulant se manifeste-t-il dans le spectre ?

[18] D'après ce spectre, quel filtrage faut-il appliquer au signal $u(t)$ pour effectuer la démodulation, c'est-à-dire pour obtenir le signal $s(t)$.

[19] Réaliser un filtre adapté au moyen des deux résistances de $10\,\mathrm{k\Omega}$ et d'un des deux condensateurs (justifer le choix). Ce filtre doit avoir un gain égal à 0,5 dans la bande passante (pour la raison donnée plus haut).

[20] Observer le signal démodulé et en faire l'analyse spectrale.

[21] À quoi voit-on sur le spectre que la démodulation n'est pas parfaite ?

[22] Augmenter la fréquence de la porteuse ($30\,\mathrm{kHz}$). Quel effet cela a-t-il sur la qualité de la démodulation ? Expliquer. Sans changer la fréquence de la porteuse, que faudrait-il faire pour améliorer la qualité de la démodulation ?

[23] Effectuer la modulation et la démodulation avec un signal triangulaire puis avec un signal carré (pour une porteuse à $30\,\mathrm{kHz}$). Commenter les résultats.

4. Modulation numérique

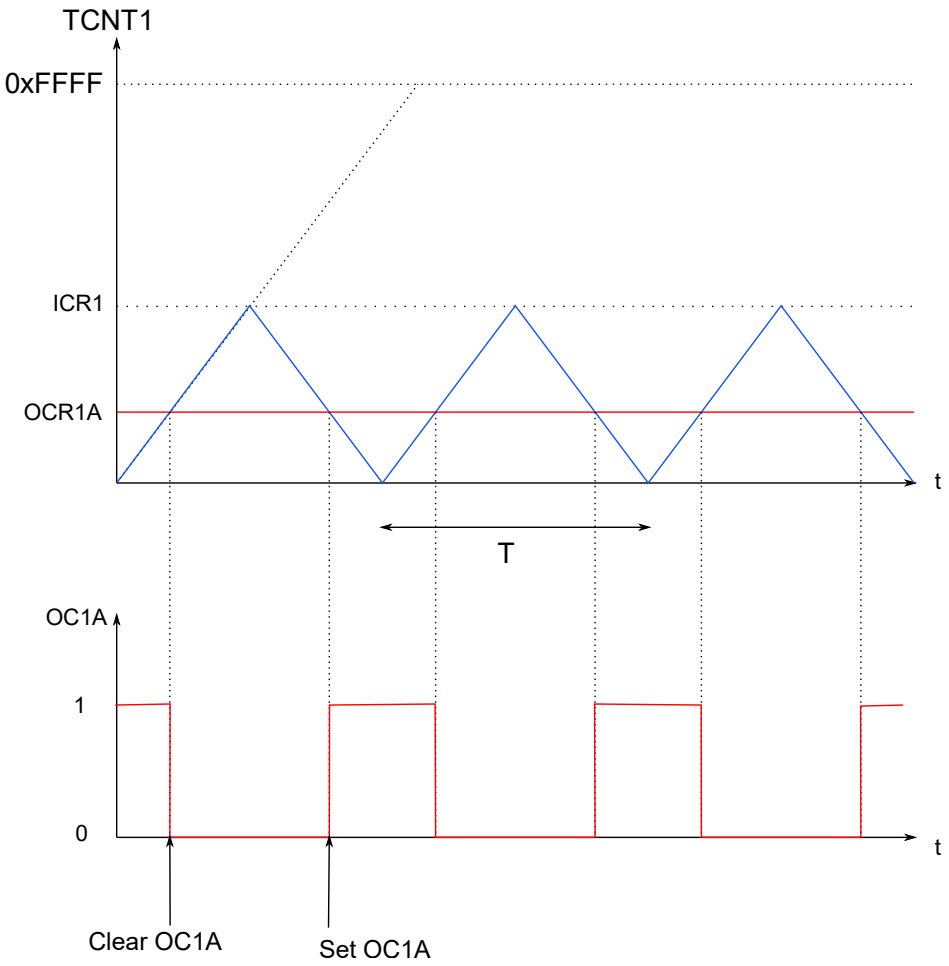
4.a. Principe

La génération numérique d'un signal MLI consiste à utiliser un compteur pour générer la porteuse. La comparaison avec la valeur du signal modulant se fait par un test numérique. Le moyen le plus simple de faire une génération numérique d'un signal MLI est d'utiliser un microcontrôleur.

Un microcontrôleur comporte en général plusieurs compteurs (ou *timers*) qui peuvent effectuer différentes opérations, en particulier la génération de signaux MLI (connus aussi sous le nom de PWM pour *pulse width modulation*). Un compteur est programmé par le microprocesseur mais il effectue ses opérations indépendamment de celui-ci.

Le microcontrôleur de l'arduino MEGA (ATmega 2560) comporte quatre compteurs 16 bits (Timers 1,3,4,5). Un compteur 16 bits comporte un registre 16 bits qui est incrémenté d'une unité à chaque top d'horloge (un registre est un emplacement de la mémoire réservé à un

usage particulier). La fréquence de l'horloge de l'Arduino MEGA est 16 MHz. Cette fréquence étant très élevée, il est possible d'incrémenter le compteur tous les 8, 64, 256 ou 1024 tops d'horloge. Le facteur appliqué (1, 8, 64, 256 ou 1024) est le *diviseur d'horloge*. Par exemple, si le diviseur 64 est utilisé, la fréquence d'incrémentation du compteur est 16/64 MHz. Le registre du compteur 1 est nommé TCNT1. On peut à tout instant lire ce registre pour connaître la valeur du compteur. Lorsque la valeur de TCNT1 atteint la valeur du registre ICR1, le compteur entre dans une phase de décrémentation jusqu'à la valeur nulle, à partir de laquelle l'incrémentation recommence. Pour générer un signal MLI, un troisième registre 16 bits est utilisé : le registre OCR1A. Le signal généré est nommé OC1A. Il est émis sur la sortie D11 de la carte Arduino MEGA. Lorsque la valeur de TCNT1 est supérieure à OCR1A, OC1A est à l'état bas. Lorsque la valeur de TCNT1 est inférieure à OCR1A, OC1A est à l'état haut. Le signal carré émis a un rapport cyclique égal à OCR1A/ICR1. En modifiant la valeur stockée dans le registre OCR1A, il sera donc possible de moduler le rapport cyclique.



Pour choisir la période T du signal MLI, on joue à la fois sur le diviseur d'horloge et sur la valeur de ICR1, sachant que la période T est égale à deux fois la valeur de ICR1 multipliée par la période de l'horloge et divisée par le diviseur d'horloge.

[24] On suppose que le facteur de division d'horloge est 1. Quelle valeur de ICR1 faut-il choisir pour avoir une porteuse dont la fréquence est au plus proche de 10 kHz ? Quelle est la fréquence effective ? De combien de valeurs de rapport cyclique dispose-t-on ?

4.b. Programme Arduino

generationTimer.ino

```
#define SORTIE 11

uint32_t icr;
uint32_t period;

//Initialisation du Timer1 pour génération MLI
void init_pwm_timer1(uint32_t period) { // période en microsecondes
  uint16_t diviseur[6] = {0,1,8,64,256,1024};
  TCCR1A = (1 << COM1A1); //Clear OC1A on compare match when upcounting, set OC1A on com
  TCCR1B = 1 << WGM13; // phase and frequency correct pwm mode, top = ICR1
  int d=1;
  icr = (F_CPU/1000000*period/2);
  while ((icr>0xFFFF)&&(d<6)) { // choix du diviseur d'horloge
     d++;
     icr = (F_CPU/1000000*period/2/diviseur[d]);
  }
  Serial.print("diviseur_1= "); Serial.println(d);
  Serial.print("icr_1 = "); Serial.println(icr);
  float periode_effective = icr*2*d*1000000.0/F_CPU;
  Serial.print("periode_1 (us) = "); Serial.println(periode_effective);
  ICR1 = icr; // valeur maximale du compteur
  TCNT1 = 0; // mise à zéro du compteur
  TCCR1B |= d; // déclenchement du compteur
}

void rapport_cyclique(float r) {
  OCR1A = icr*r;
  Serial.print("OCR1A = "); Serial.println(OCR1A);
}

void setup() {
  Serial.begin(9600);
  pinMode(SORTIE,OUTPUT);
  period = 100; // période en microsecondes
  init_pwm_timer1(period);
  rapport_cyclique(0.1);
}

void loop() {

}
```

La fonction `init_pwm_timer1` configure le Timer 1 pour qu'il génère un signal MLI dont la période est donnée en microsecondes (sur la sortie D11). Une fois le compteur lancé, le changement du rapport cyclique se fait simplement en modifiant la valeur stockée dans le registre OCR1A. La fonction `rapport_cyclique` permet de faire cela à partir d'un rapport cyclique donné sous la forme d'un `float` compris entre 0 et 1. La fonction de configuration affiche le diviseur d'horloge, la valeur de ICR et la période effective.

[25] Ouvrir le fichier avec le logiciel Arduino.

[26] Brancher la carte Arduino avec le câble USB, qui sert à transférer le programme et qui permet d'alimenter la carte. Dans le menu Outils, sélectionner le port sur lequel se trouve

la carte et le type de carte (Mega 2560). Téléverser le programme.

[27] Brancher les bornes GND et D11 sur la plaque d'essai. Relier à la borne GND de l'oscilloscope et à la borne de signal de la voie 1. Observer le signal à l'oscilloscope.

[28] Effectuer une analyse spectrale afin de déterminer la fréquence au dixième de hertz près. Est-elle conforme à la valeur programmée ? Comment expliquer l'écart ?
Pour effectuer une modulation sinusoïdale du rapport cyclique, compléter la fonction `loop` par :

```
void loop() {
  float r;
  float f = 1;
  float t = micros()*1e-6;
  r = 0.5+0.3*sin(2*PI*f*t);
  rapport_cyclique(r);
}
```

La fonction `loop` est exécutée en boucle. La fonction `micros()` renvoie le temps écoulé depuis l'initialisation de l'Arduino en microsecondes (sous la forme d'un entier 32 bits).

[29] Observer le signal MLI à l'oscilloscope puis effectuer la démodulation au moyen du filtre RC. Faire varier la fréquence de modulation.
Le Timer 1 comporte deux autres sorties (OC1B et OC1C) dont les rapports cycliques se règlent avec les registres OCR1B et OCR1C. Les bornes correspondantes sont D12 et D13.

[30] Compléter le programme afin qu'il génère deux signaux MLI sur les sorties D11 et D12, sinusoïdaux et de mêmes amplitudes mais déphasés de $\pi/2$.

5. Annexe : fonction FFT de l'oscilloscope

La fonction FFT (Fast Fourier Transform) permet d'obtenir le spectre d'un signal. Le calcul du spectre par TFD se fait comme dans la partie 3 (sans complément par des zéros). L'oscilloscope trace le spectre pour des fréquences dans l'intervalle $[0, \frac{f_e}{2}]$, qui est le domaine de fréquence correspondant au spectre du signal analogique qu'on étudie.

- ▷ Appuyer sur FFT pour activer le calcul du spectre.
- ▷ Si le signal à analyser est sur la voie 1, sélectionner la voie 1 comme source.
- ▷ Il est conseillé de garder la trace du signal mais de la décaler vers le haut.
- ▷ L'échelle verticale du spectre (l'amplitude) est en décibel ou bien en Volts (RMS). Pour changer le type d'échelle, aller dans *Param*. Le bouton *Echelle* permet de modifier cette échelle.
- ▷ La durée T du signal échantillonné (signal dont la TFD est calculée) correspond à la totalité du signal visible sur l'écran. Si on augmente cette durée en tournant le bouton de réglage de l'échelle de temps, la résolution fréquentielle du spectre augmente (celle-ci est affichée sur l'écran).
- ▷ L'intervalle de fréquences du spectre tracé à l'écran se règle avec les boutons *Plage* et *Centre*.
- ▷ Le bouton *Décal* permet de décaler le spectre verticalement.