

R&T

Semestre 2

2022/2023

SAE22 : Mesurer et caractériser un signal**Simulations numériques**

Cette Situation Acquisition et d'Evaluation va traiter de réalisation ou acquisition de signaux numériques et de leurs exploitations. Elle se décompose en 4 parties :

- Partie 1 : Prise en main de Matlab
- Partie 2 : Calcul numérique d'une intégrale
- Partie 3 : Etude d'un signal temporel
- Partie 4 : Etude d'un canal Radio

Pour réaliser le travail demandé, vous disposez de :

- 6 séances de 2h encadrées
- 3 séances de 2h en autonomie à réaliser en salle à l'IUT.
- Du travail personnel à réaliser entre les séances.

L'évaluation se fera en contrôle continu sur la progression et vous aurez à produire :

- Un dossier de fichiers pdf comprenant vos programmes commentés
- Un compte-rendu répondant aux différentes questions et analyses demandées.

Partie 1 : Prise en main de Matlab

I/ Formation Matlab : « Onramp »

👉 Suivre les chapitres 1 (course Overview) à 9 (Plotting data) du module « Onramp » d'autoformation de Matlab : <https://matlabacademy.mathworks.com/>

II/ Génération d'un signal sinusoïdal $x(t)$

On désire générer et afficher le signal $x(t) = 5 \times \cos(2 \times \pi \times 1000 \times t)$ où $x(t)$ est une tension en Volts.

Matlab étant un logiciel numérique, il va devoir **échantillonner** ce signal. On choisira **$f_e = 40\text{kHz}$** comme fréquence d'échantillonnage.

Le début du programme Matlab est le suivant :

```
1 - f=1000; %fréquence du signal sinusoïdal
2 - A=5; %amplitude de la sinusoïde
3
4 - fe=40000; %fréquence d'échantillonnage
5 - Te=1/fe; %durée d'un échantillon
6 - Ns=200; %nombre d'échantillons
7
8 - t=0:Te:(Ns-1)*Te;
9
10 - x=5*cos(2*pi*f*t);
```

Figure 1: début du programme de génération d'un cosinus

1) Que représente la variable t ? Quelle est sa nature ?

La variable t représente le temps. Tous les échantillons seront pris à un instant t commençant à 0, finissant à $(N_s-1) \times T_e$ avec un pas de T_e . C'est une matrice.

2) Que fait la ligne 10 ? Quelle est la nature de x ?

La ligne 10 crée le signal $x(t)$ et l'attribue à la variable x en calculant, $5 \times \cos(2 \times \pi \times f \times t)$ pour tous les t précédemment calculés. C'est aussi une matrice de points.

3) Combien de période du signal $x(t)$ seront calculées ?

$$f = 1000\text{Hz}$$

$$f_e = 40000\text{ Hz}$$

$$N_s = 200$$

$$\text{Donc } t = 199 / 40000 = 0.004975\text{s.}$$

$$T = 1/f = 10^{-3}\text{ s}$$

$$t/T = 0.004975/0.001 = 4.975$$

Il y a donc quasiment 5 périodes calculées.

4) Quelle petite critique ou amélioration peut être apportée à ce programme ?

III / Affichage du signal $x(t)$

✎ Chercher de l'aide sur la fonction « plot » puis compléter le programme suivant afin d'afficher $x(t)$ avec les informations suivantes :

- Couleur : rouge
- Mettre un titre : title
- Mettre une légende : legend
- Mettre le nom des axes sur la figure : xlabel et ylabel
- Fixer les axes suivants : de 0 à 2ms et de -10V à +10V : axis

Le code de $x(t)$ permettant d'afficher le signal avec les informations ci-dessus est :

```
plot(t,x, 'red'); %permet l'affichage du signal x en fonction de t
title("Représentation d'un cosinus de 1000Hz");
legend("x(t)=5*cos(2pi*1000*t)");
ylabel("Volt"); %appliquer un titre à l'axe des y
xlabel("t(ms)"); %appliquer un titre à l'axe des x
grid on; %on affiche la grille
axis([0 0.002 -10 10]); %nous définissons les limites des axes
yticks(-10:2:10); %nous définissons l'échelle de chaque axe
xticks(0:0.0002:0.002);
yline(0); %nous mettons l'axe des origines
```

Figure exp 1 : script permettant un affichage clair du signal $x(t)$

Il faut obtenir une figure similaire à la Figure 2 :

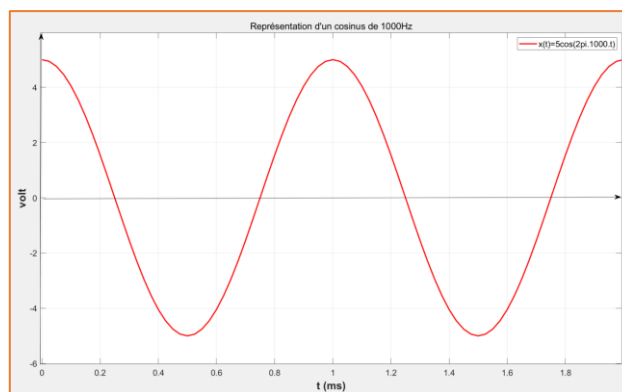


Figure 2 : affichage de la fonction cosinus $x(t)$

Nous obtenons sur Matlab la figure :

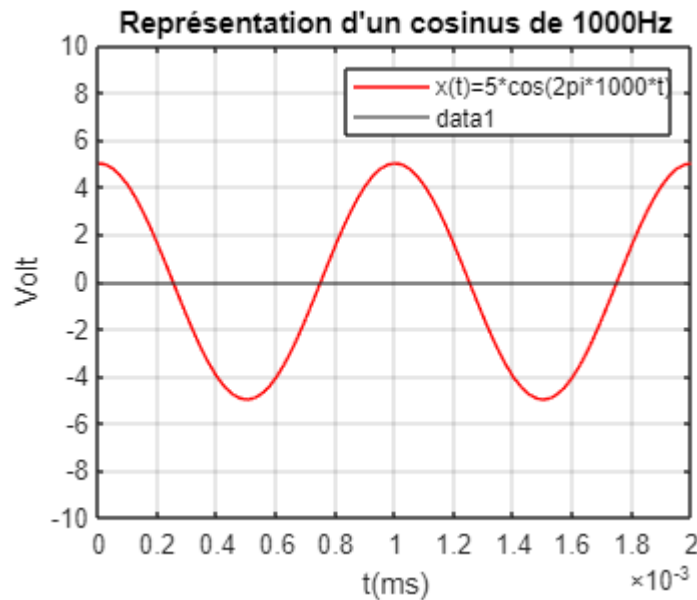


Figure exp 2 : Observation du signal $x(t)$ sur Matlab

On veut maintenant créer un signal $y(t) = 3 \times \sin(2 \times \pi \times 1000 \times t)$

👉 Compléter le programme afin de créer et d'afficher le signal $y(t)$ en bleu.

Nous obtenons le programme :

```
y=3*sin(2*pi*f*t);
plot(t,y,'blue');
title("Représentation d'un sinus de 1000Hz et d'amplitude 3V");
legend("y(t)=3*sin(2*pi*1000*t)");
ylabel("Volt");
xlabel("t(ms)");
grid on;
axis([0 0.002 -10 10]);
yticks(-10:2:10);
xticks(0:0.0002:0.002);
yline(0);
```

Figure exp 3 : script permettant l'affichage de $y(t)$

👉 Etudier la fonction « subplot » au moyen du menu « Help ».

Subplot permet d'afficher plusieurs tracés en même temps dans la console.

👉 Modifier le programme afin d'afficher les deux courbes sur deux graphes sur la même figure et obtenir la représentation suivante :

Ensuite, juste avant de déclarer le plot de x , nous mettons la ligne `subplot(2, 1, 1)` ; et juste avant de déclarer le plot de y , nous mettons la ligne `subplot(2, 1, 2)` ;.

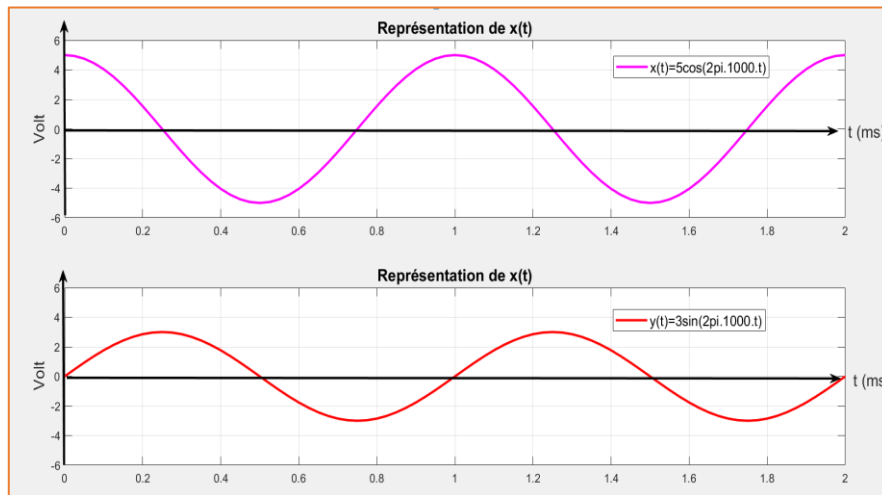


Figure 3 : représentation de deux signaux en utilisant la commande « subplot »

De ce fait, nous obtenons la représentation :

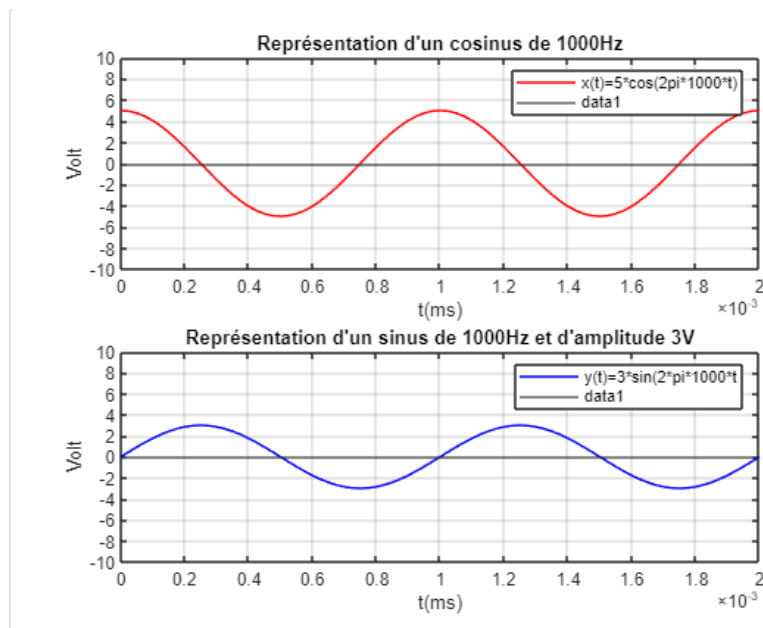


Figure exp 4 : observation des deux signaux $x(t)$ et $y(t)$

Il est parfois intéressant d'avoir les 2 représentations graphiques sur le même graphique.

👉 Rechercher comment afficher 2 courbes sur le même graphique avec la fonction « plot ».

Pour afficher deux courbes sur le même graphique, il faut renseigner les deux courbes dans le plot de cette manière :

```
plot(t,x,"red",t,y,'blue'); %x(t) s'affichera en rouge et y(t) en bleue
```

Figure exp 5 : ligne permettant de mettre $x(t)$ et $y(t)$ sur le même graphique

✎ Compléter le programme pour obtenir la représentation suivante :

Nous obtenons le programme suivant :

```
y=3*sin(2*pi*f*t);
plot(t,x,"red",t,y,'blue');
title("Représentation d'un sinus de 1000Hz et d'amplitude 3V");
legend("y(t)=3*sin(2*pi*1000*t)","x(t)=5*cos(2*pi*f*t)");
ylabel("Volt");
xlabel("t(ms)");
grid on;
axis([0 0.002 -10 10]);
yticks(-10:2:10);
xticks(0:0.0002:0.002);
yline(0);
```

Figure exp 6 : script permettant d'afficher les deux courbes sur le même graphique

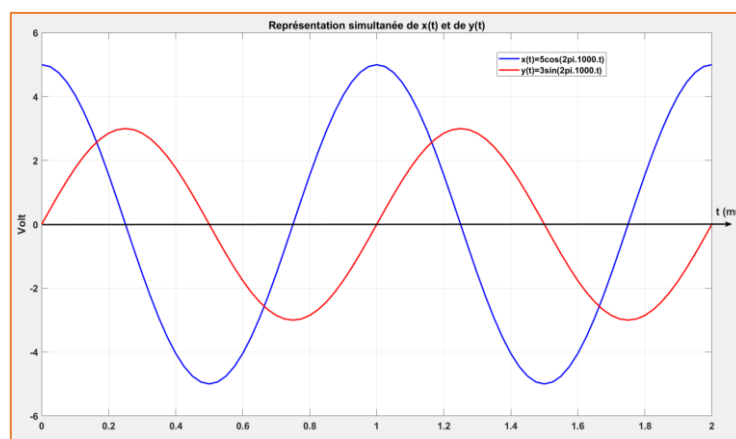


Figure 4 : représentation de deux signaux sur un même graphique

De ce fait, nous obtenons le graphique suivant :

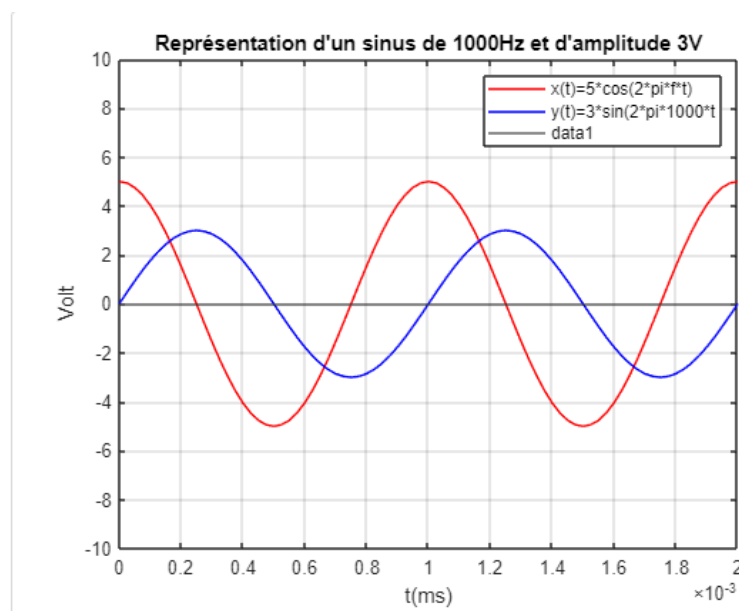


Figure exp 7 : observation des deux courbes de x(t) et y(t) sur le même graphique

IV / Formation Matlab : suite

✎ Finir l'autoformation à Matlab Onramp.

Voici le certificat de fin d'autoformation :



Course Completion Certificate

Maxence DUCREY

has successfully completed **100%** of the self-paced training course

MATLAB Onramp

DIRECTOR, TRAINING SERVICES

15 May 2024

V / Affichage du spectre

Matlab dispose d'une fonction FFT (Fast Fourier Transform) permettant de calculer la transformée de Fourier d'un signal.

Afin de faciliter le travail, une fonction, « spectre.m », permettant d'obtenir le spectre d'un signal, est fournie.

✎ Télécharger sur l'ENT puis décompresser le dossier « TP initiation Matlab.zip » contenant :

- Le programme principal : « affichage_spectre_sinus.m »
- Une fonction pour le calcul des spectres : « spectre.m »

1) Etudier la ligne 28 ainsi que la fonction « spectre.m » à laquelle elle fait appel.

```
27 %calcul de X(f): spectre en amplitude de x(t)
28 [X f]=spectre(x,fe,Ns); % appel à la fonction spectre.
```

Cette ligne fait appel à la fonction spectre qui est dans le même dossier que affichage_spectre_sinus.m Elle utilise comme argument x, le signal temporel, fe, la fréquence d'échantillonnage, Ns, le nombre d'échantillons utilisés. De ce fait cette fonction va renvoyer deux variables X, qui est le spectre en amplitude du signal, et f qui contient les fréquences correspondantes.

2) Exécuter le script et valider le résultat.

Quand nous exécutons le script, nous obtenons :

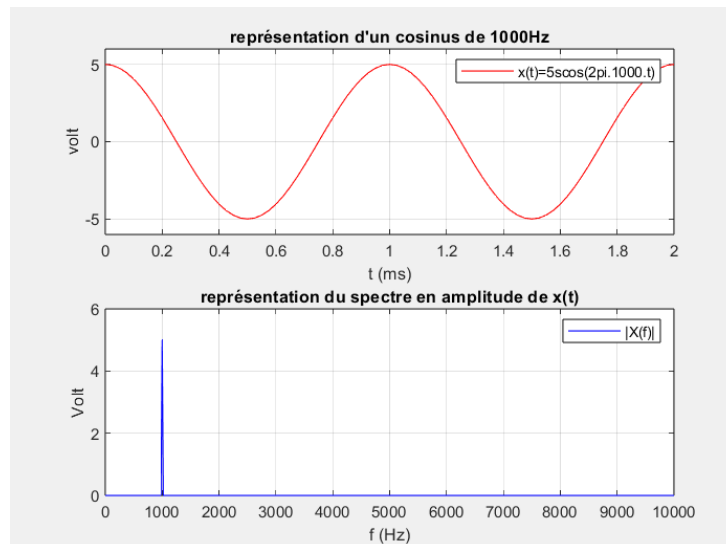


Figure exp 8 : création du spectre de $x(t)$ grâce à la fonction « spectre »

Nous obtenons bien le spectre de $x(t)$, nommé $X(f)$, avec comme amplitude 5V et une fréquence de 1000Hz.

Remarque : par la suite vous pourrez réutiliser cette fonction « spectre.m » pour tracer le spectre de n'importe quel signal.

✎ Créer une fonction « spectredBV.m » afin d'afficher maintenant le spectre en dBV ou dBμV. On utilisera pour cela la fonction $\log_{10}(X)$. On ne calculera pas ici la valeur efficace pour le moment.

Pour implémenter la fonction spectredBV.m, il suffit de copier le code de spectre.m et de rajouter une ligne permettant la conversion en dBV. De ce fait, nous obtenons :

```
function [X_dBV f] = spectredBV(signal, fe, Nech)

df=fe/Nech;          % les échantillons fréquentiels sont espacés de fe/Nech
f=0:df:fe/2-df;      %création d'un vecteur fréquence constitué de Nech/2 points
                    %répartis entre 0 et fe/2

%calcul et affichage du spectre du signal
X=fft(signal)/Nech;
X=[X(1) 2*X(2:Nech/2)]; %passage du spectre bilatéral au spectre unilatéral...

X=abs(X); % calcul du module pour afficher le spectre en amplitude.

%On passe X en dBV :
X_dBV = 20*log10(X);

end;
```

Figure exp 9 : code de la fonction pour afficher un spectre en dBV

Nous avons ajouté la ligne $X_dBV = 20*\log_{10}(X)$ permettant la conversion.

✎ Compléter le programme pour tracer le spectre en dBV.

Pour tracer le spectre en dBV, il faut modifier le programme comme suit :

```
[X_dBV f]=spectredBV(x,fe,Ns);

%affichage du spectre en dBV
subplot(1,1,1)
plot(f,X_dBV,"cyan");
title('représentation du spectre en dBV de x(t)')
xlabel('f (Hz)')
ylabel('dBV')
legend('|X(f)| en dBV')
axis([0 10000 -40 20]) %affichage entre 0 et 10kHz
grid on
```

Figure exp 10 : code pour afficher le spectre en dBV

Avec ce code, nous remplaçons dans le plot X par X_dBV pour afficher le spectre en X_dBV.

✎ Valider le résultat obtenu.

Sur Matlab, nous obtenons le spectre suivant :

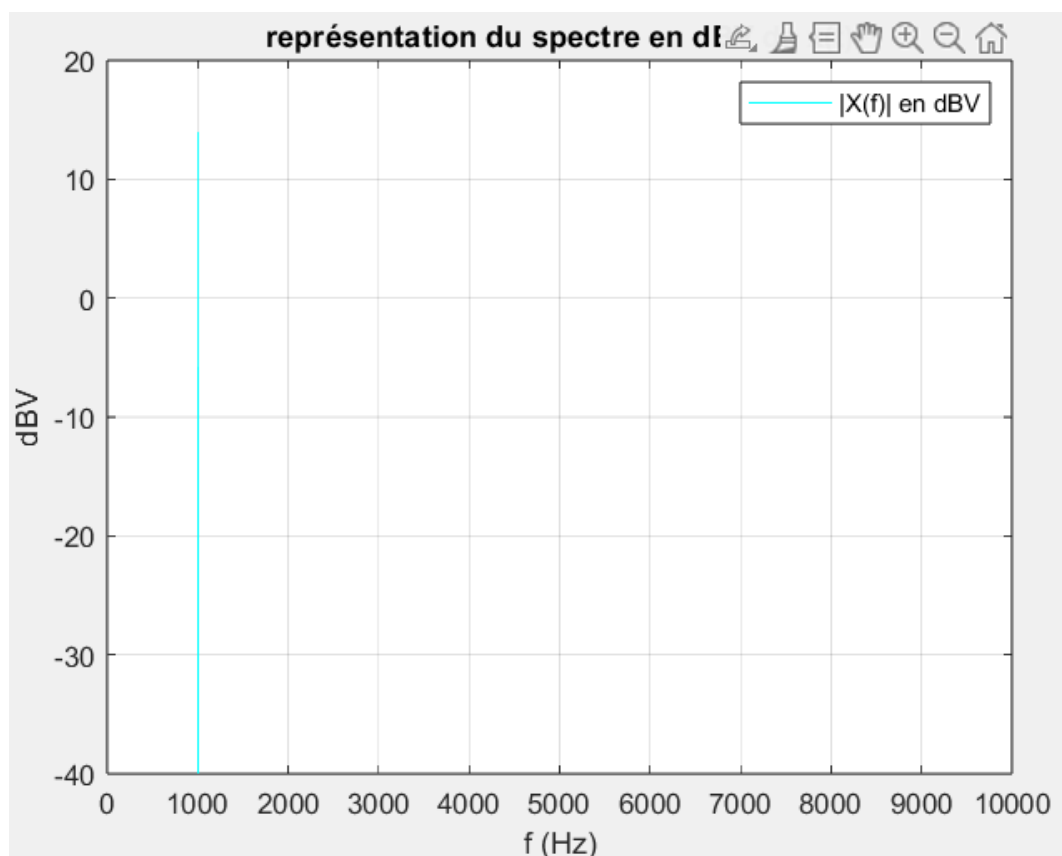


Figure exp 11 : observation du spectre de x(t) en dBV

Nous trouvons une valeur qui est environ égale à 14 dBV.

👉 Réaliser l’affichage comme montré ci-dessous, en mieux (penser aux noms des axes et autres ...)

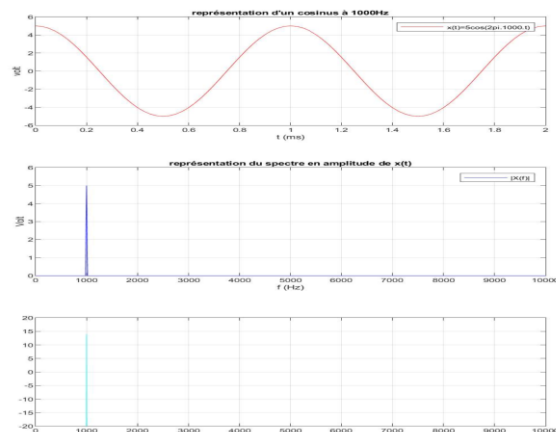


Figure 5 : représentation du signal $x(t)$, de son spectre et de ce spectre en dBV

Pour réaliser cet affichage, il faut changer le nombre de lignes du subplot en le mettant à 3. Nous changeons ensuite les indices des graphiques. Nous obtenons donc :

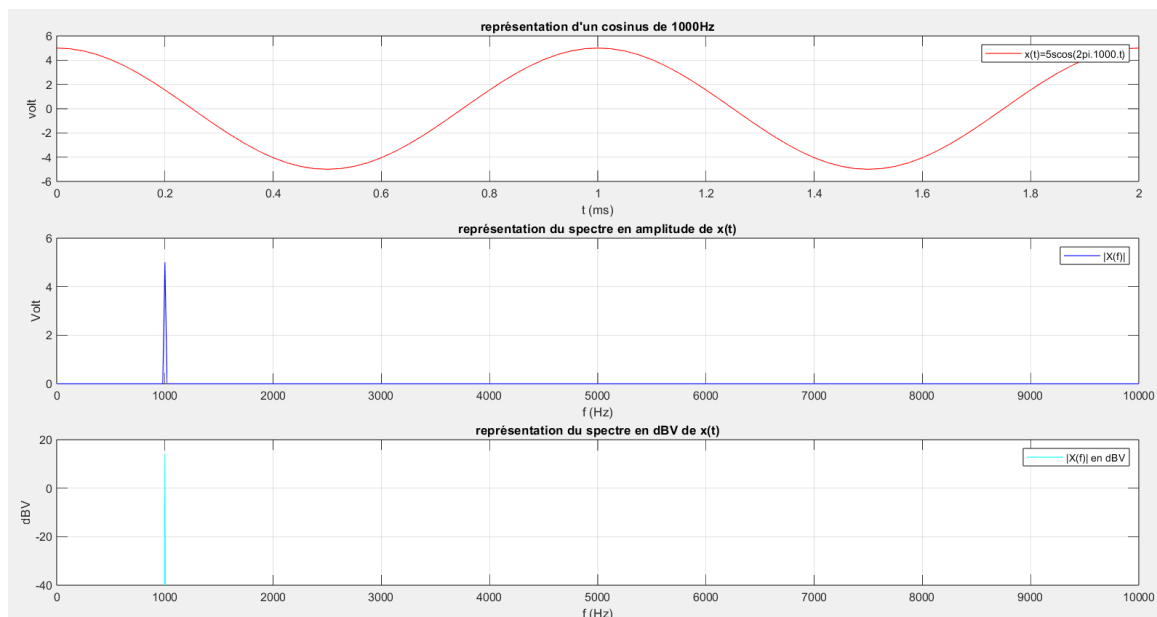


Figure exp 12 : observation du signal $x(t)$ et ses spectres

VI / Multiplication de deux signaux sinusoïdaux

On veut à présent réaliser la multiplication de deux signaux sinusoïdaux. En effet cette étape est fondamentale dans la transmission de signaux car elle réalise une transposition de fréquence. Ceci sera vu à l'aide du spectre de ce produit.

Les caractéristiques de l'échantillonnage seront les suivantes :

- Fréquence d'échantillonnage : $f_{ech}=40\text{kHz}$
- Nombre d'échantillons : $N_s=2000$

Les signaux à multiplier sont les suivants :

- $x_1(t) = A_1 \times \cos(2 \cdot \pi \cdot f_1 \cdot t)$ avec $A_1=5\text{V}$ et $f_1=1\text{kHz}$
- $x_2(t) = A_2 \times \cos(2 \cdot \pi \cdot f_2 \cdot t)$ avec $A_2=3\text{V}$ et $f_2=3\text{kHz}$

👉 Générer les deux signaux $x_1(t)$ et $x_2(t)$.

Pour générer les deux signaux $x_1(t)$ et $x_2(t)$, nous devons créer un nouveau programme et mettre ces lignes :

```
%caractéristiques de l'échantillonnage
fe = 40000; %fréquence d'échantillonnage
Ns = 2000; %nombre d'échantillons
Te = 1/fe; %période d'échantillonnage

%caractéristiques de x1(t) et x2(t)
A1 = 5; %amplitude
f1 = 1000; %fréquence

A2 = 3;
f2 = 3000;

t = 0:Te:(Ns-1)*Te;

%signaux x1 et x2
x1 = A1*cos(2*pi*f1*t);
x2 = A2*cos(2*pi*f2*t);
```

Figure exp 13 : génération des signaux $x_1(t)$ et $x_2(t)$

1) Vérifier la taille des vecteurs x1 et x2 créés. Justifier.

La taille des vecteurs x1 et x2 créés est :

 **x1** *1x2000 double*
 **x2** *1x2000 double*

Ce sont des tableaux de 1 ligne par 2000 colonnes. En effet : $T_e = \frac{1}{f_e} = \frac{1}{40000} = 25 * 10^{-6} s$

Et $(N_s - 1) * T_e = 1999 * 25 * 10^{-6}$

Ce qui signifie que nous commençons à 0 avec un pas de 25 μs et ce sur (Ns-1) points. C'est pourquoi nous avons $1999 + 1 = 2000$ points/colonnes.

2) Quelle est la différence entre les opérations suivantes sous Matlab : « * » et « .* » ? Quel est l'opérateur à utiliser pour la multiplication demandée ?

L'opérateur « * » va simplement faire un produit matriciel. Si A est de taille m*n et B est de taille n*p, alors A*B sera de taille m*p.

Quant à l'opérateur « .* », celui-ci va multiplier les éléments de chaque matrice ayant le même indice. Par exemple : A = [1 2 ; 3 4] et B = [5 6 ; 7 8] alors AB = [1*5 2*6 ; 3*7 4*8].

Dans notre cas, nous souhaitons multiplier les deux signaux. Donc l'opérateur « .* » est plus approprié.

3) Quelle est l'opération réalisée avec l'opérateur « .* » ? C'est-à-dire, en fonction des indices, quels sont les termes multipliés entre eux ? (z(1)=x1(??)*x2(??))

L'opération réalisée est : z(1) = x1(1) * x2(1) puis z2 = x1(2) * x2(2) etc.

4) Calculer z le produit de ces deux signaux : **z = x1.* x2**

Nous rajoutons une ligne au programme pour réaliser l'opération.

5) Quelle est la dimension du vecteur z ? Justifier.

La dimension du vecteur z est 1*2000 ce qui est cohérent car comme les matrices sont de mêmes tailles, tous les indices de x1 sont multipliés avec leur équivalent dans x2 donc les 2000 éléments de x1 et de x2 sont réunis dans z.

👉 Afficher sur une même figure, mais sur trois graphes les chronogrammes de x1(t), x2(t) et z(t).

Le code de x1(t) à répéter pour x2(t) et z(t) en changeant les variables est :

```
%Affichage de x1(t)
subplot(3,1,1);
plot(t, x1, "green");
title("Représentation de x1 en fonction de t");
xlabel("t(ms)");
ylabel("Volt");
legend("x1(t) = 5*cos(2*pi*1000*t)");
axis([0 0.002 -10 10]);
grid on;
yline(0);
xticks(0:0.0002:0.002);
yticks(-10:2:10);
```

Figure exp 14 : code de x1(t) permettant de l'afficher

De ce fait, nous obtenons :

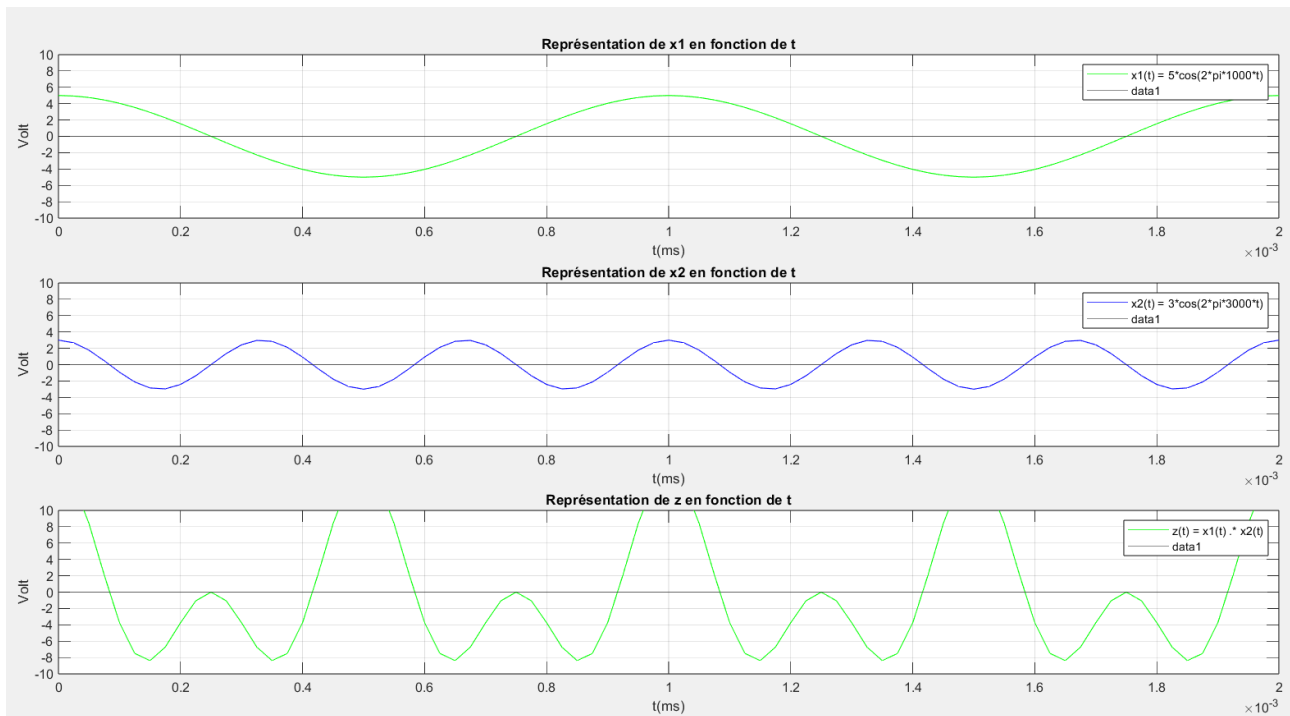


Figure exp 15 : observation des trois signaux en fonction de t

👉 Calculer les spectres de ces trois signaux et afficher les sur trois graphes d'une nouvelle figure.

Le code pour calculer les spectres est :

```
%spectre des signaux x1(t), x2(t), z(t) en dBV
[x1_dBV f1]=spectredBV(x1,fe,Ns);
[x2_dBV f2]=spectredBV(x2,fe,Ns);
[z_dBV fz]=spectredBV(z,fe,Ns);
```

Figure exp 16 : code pour générer les spectres des trois signaux

Les résultats obtenus doivent être de la forme suivante :

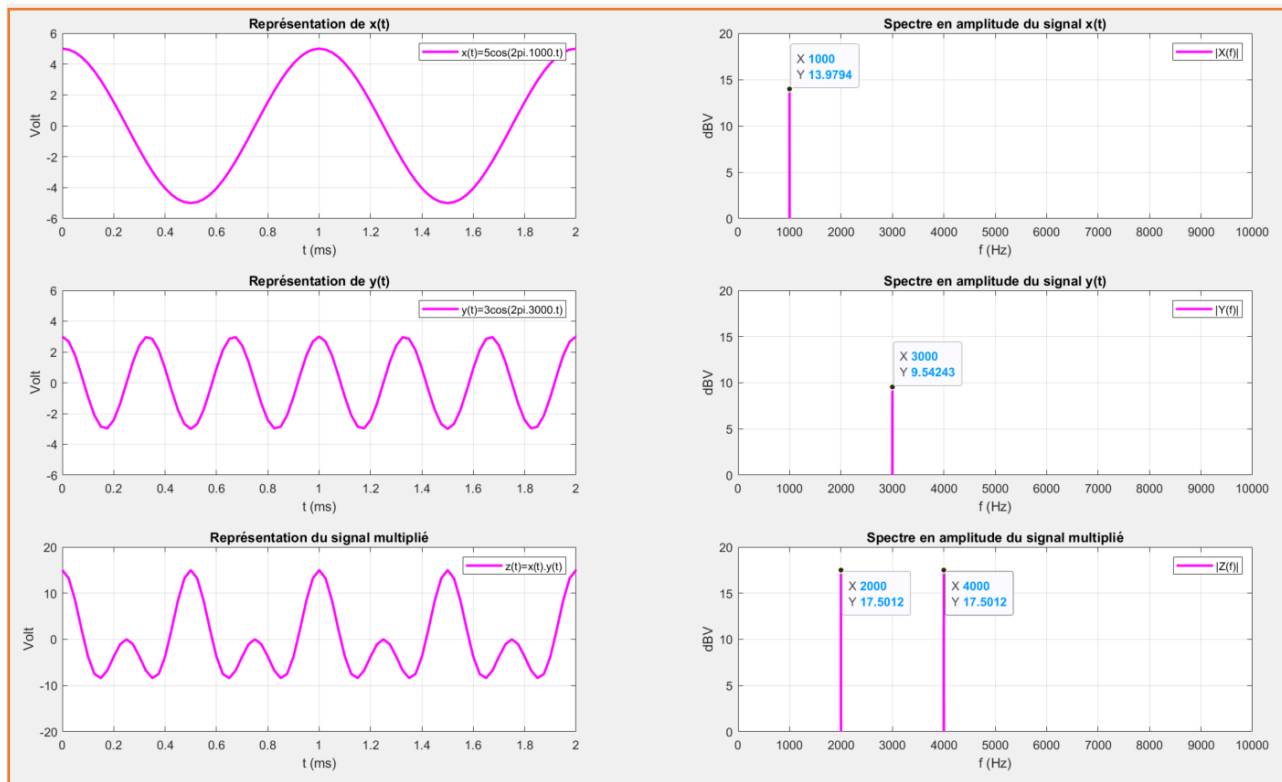


Figure 6: multiplication de deux signaux sinusoïdaux

Sur Matlab, nous obtenons :

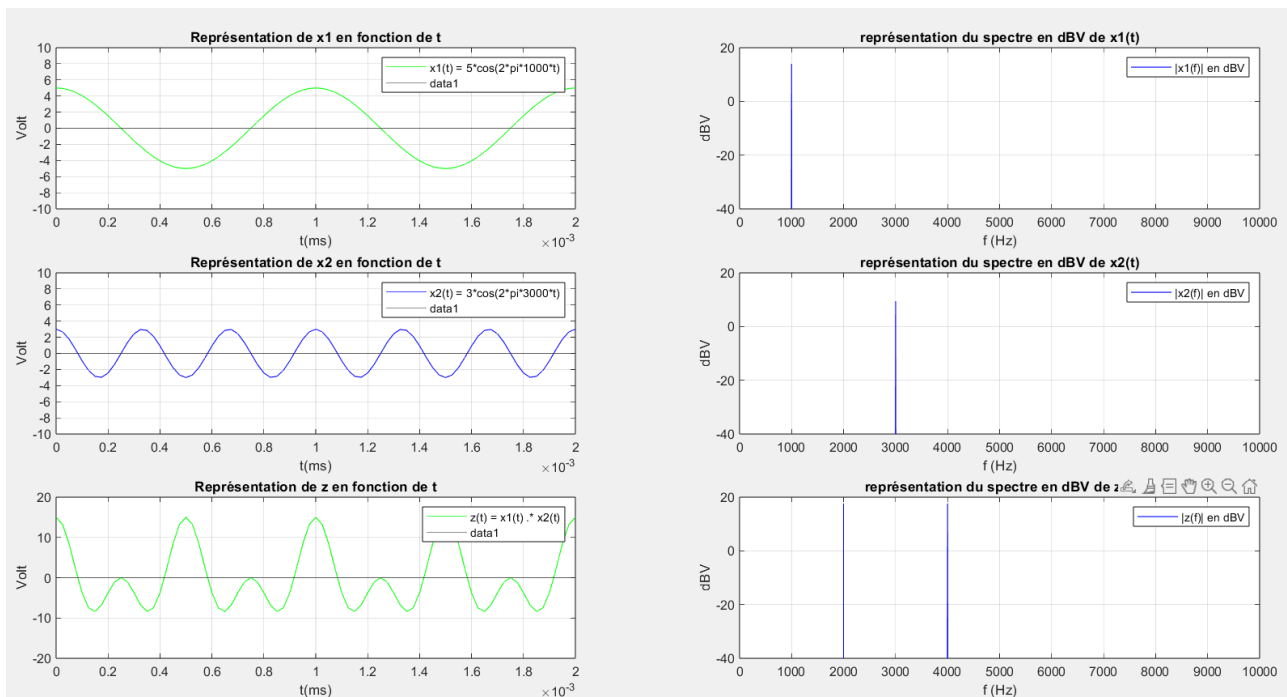


Figure exp 17 : observation des signaux temporels et des spectres en dBV

Nous avons bien les mêmes graphiques.

- 6) Recommencer en prenant maintenant $x_1(t) = x_2(t) = 3 \times \cos(2 \cdot \pi \cdot 1000 \cdot t)$. Analyser. Quelle est la fonction réalisée ?!

Partie 2 : Calcul numérique d'une intégrale

I / Présentation du contexte

Soit $f(t)$ une fonction continue définie sur un intervalle $t \in [a; b]$. On souhaite évaluer numériquement l'intégrale : $I = \int_a^b f(t) \cdot dt$

L'intégrale d'une fonction correspond en fait à son aire algébrique par rapport à l'axe des abscisses.

Pour donner une valeur approchée, mais qui sera simple à calculer en programmation, l'intervalle $[a; b]$ est découpé en **n sous-intervalles** de longueur $dt = \frac{(b-a)}{n}$ (cf. Figure 1).

Cela revient en fait à échantillonner la fonction $f(t)$ à la fréquence $fe = \frac{1}{dt}$.

Le $i^{\text{ème}}$ échantillon sera ainsi à l'abscisse : $t_i = a + (i-1) \cdot dt$ où i est un nombre entier compris entre 0 et n ce qui s'écrit : $i \in \langle 0 ; n \rangle$. On aura donc $(n+1)$ échantillons (pour n intervalles...).

Les méthodes de calcul numérique d'intégrales consistent à remplacer, sur chaque sous-intervalle, la fonction $f(t)$ par un polynôme.

Dans ce qui suit, on va essayer d'évaluer numériquement l'intégrale :

$$I = \int_0^1 (1 + 2t^2) \cdot dt$$

Ceci sera fait par plusieurs méthodes différentes

II / Méthode des rectangles

La méthode des rectangles consiste à remplacer sur chaque sous-intervalle $J_i = [t_i ; t_{i+1}]$, la fonction $f(t)$ par $f(t_i)$.

La Figure 1 illustre le calcul de $I = \int_0^1 (1 + 2t^2) \cdot dt$ par la méthode des rectangles avec **n=4** segments.

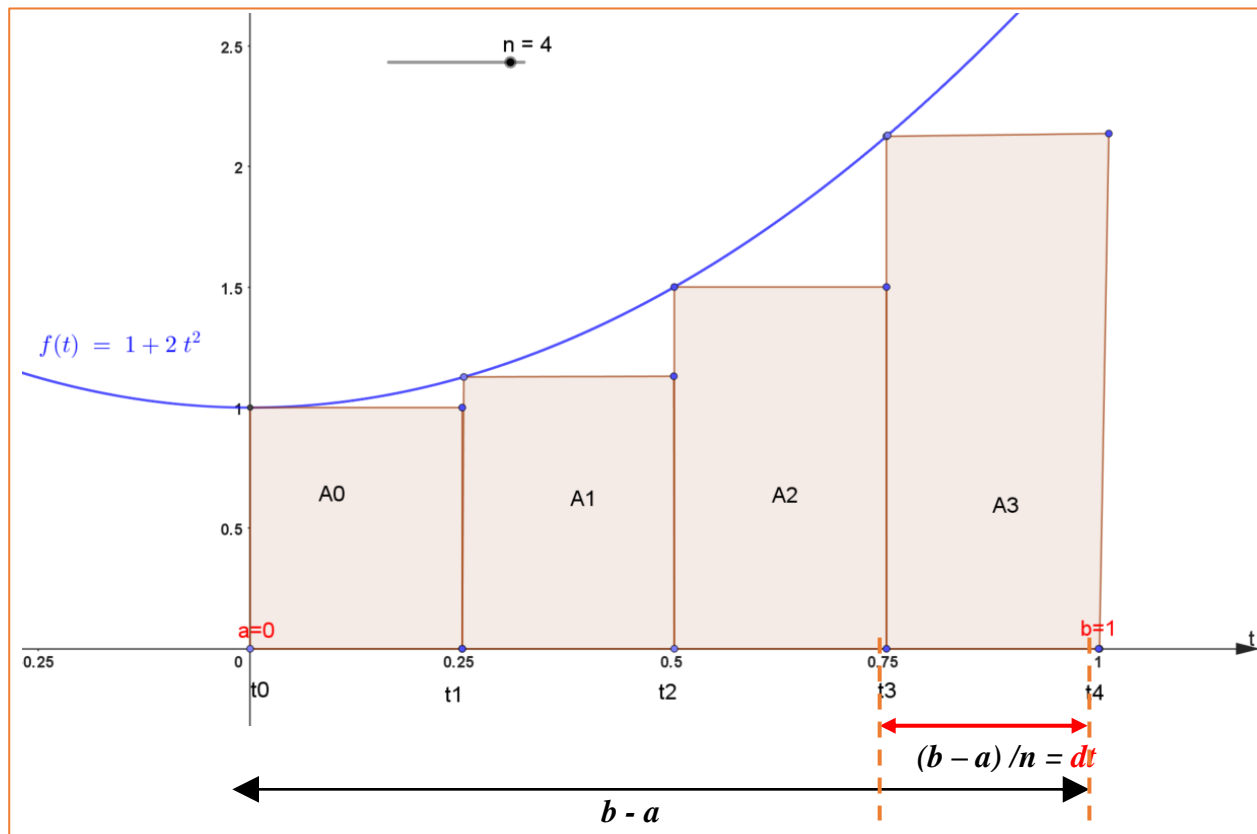


Figure 1: intégration par la méthode des rectangles

Ainsi, selon cette approximation : $I \approx R = A_0 + A_1 + A_2 + A_3$

Avec : $R = f(t_0) \times dt + f(t_1) \times dt + f(t_2) \times dt + f(t_3) \times dt$ (eq 1)

Préparation théorique :

- 1) Calculer théoriquement la valeur de I. Faire une application numérique à 10^{-5} près.

On a : $f(t) = (1 + 2t^2)$. La primitive de cette fonction est $F(t) = t + \frac{1}{3}t^3$. L'intégrale entre 0 et 1 vaut $\int_0^1 1 + 2t^2 = (1 + \frac{2}{3}) - 0 = \frac{5}{3}$.

- 2) Pour $n=4$, calculer à l'aide d'un tableur la valeur approchée de I la méthode des rectangles, c'est-à-dire **R** (eq 1).

Avec Excel, nous trouvons que $R = 1.4375$:

A	B	C	D	E
Valeur de n :	4		f(t0) =	1
Fonction à intégrer :	$1 + 2t^2$		f(t1) =	1,125
b :	1		f(t2) =	1,5
a :	0		f(t3) =	2,125
dt :	0,25			
			I =	1,4375

Figure exp 18 : tableur Excel pour calculer l'intégrale

- 3) Montrer que de manière générale la formule du calcul approché de l'intégrale I par la méthode des rectangles sur n intervalles est : $R = \frac{b-a}{n} \times \sum_{i=0}^{n-1} f(t_i)$.

On sait que : $R = f(t_0) \times dt + f(t_1) \times dt + f(t_2) \times dt + f(t_3) \times dt \quad (\text{eq 1})$

Avec $dt = (b-a)/n$ donc $R = (b-a) / n * \sum_{k=0}^{n-1} f(tk)$

Réalisation :

On se propose maintenant d'implanter le calcul de cette intégrale en utilisant Matlab.

👉 **Compléter le script suivant** afin d'implanter l'approximation de la méthode des rectangles sur Matlab.

```
a=0;
b=1;
n=100;
dt=(b-a)/n;
t=(0:n-1)*dt;

x=1+2*t^2;
plot(t,x);
à terminer ...
```

Le script complété est :

```
close;
clc;
clear;

a=0;
b=1;
n=4;
dt=(b-a)/n;
t=(0:n)*dt;

x=1+2*t.^2;
plot(t,x);

I = sum(x(1:end-1))*dt
```

Figure exp 19 : script complété pour calculer une intégrale par la méthode des rectangles

Sur Matlab, nous trouvons $I = 1.4375$:



1.4375

Les résultats concordent.

III / Facultatif : vitesse de convergence ^(*)

Dans cette partie, on prendra toujours $I = \int_0^1 (1+2t^2)dt$.

- 1) A l'aide de la valeur théorique de I et des résultats de simulations, calculer, pour n=10, l'erreur d'approximation pour la méthode des rectangles : $|I - R|$.
- 2) Représenter l'évolution de ces erreurs en fonction de n pour n variant de 2 à 1000. On pourra prendre une échelle logarithmique. Analyser.

Justification théorique :

On note : $m_1 = \sup_{[a,b]} |f'(t)|$, la valeur maximale de f' sur l'intervalle $[a, b] = [0,1]$.

- 3) Déterminer la formule de $f'(t)$.
- 4) Montrer que : $m_1 = 4$
- 5) Vérifier par simulation sur Matlab la formule théorique de majoration d'erreur :

$$|I - R| \leq \frac{(b-a)^2}{n} m_1$$

- 6) Vérifier de même que l'erreur commise par la méthode des trapèzes est majorée selon la formule suivante :

$$|I - T| \leq \frac{(b-a)^3}{n^2} m_2 \quad \text{avec } m_2 = \sup_{[a,b]} |f''(t)|$$

- 7) Faire les simulations nécessaires afin de comparer les vitesses de convergences de ces 2 méthodes.

IV / Méthodes des trapèzes

On se rend compte que la méthode des rectangles peut être améliorée. La méthode des trapèzes consiste à approximer la fonction $f(t)$ dans chacun des sous intervalles par des segments. La figure 2 illustre cette méthode pour n=4 intervalles.

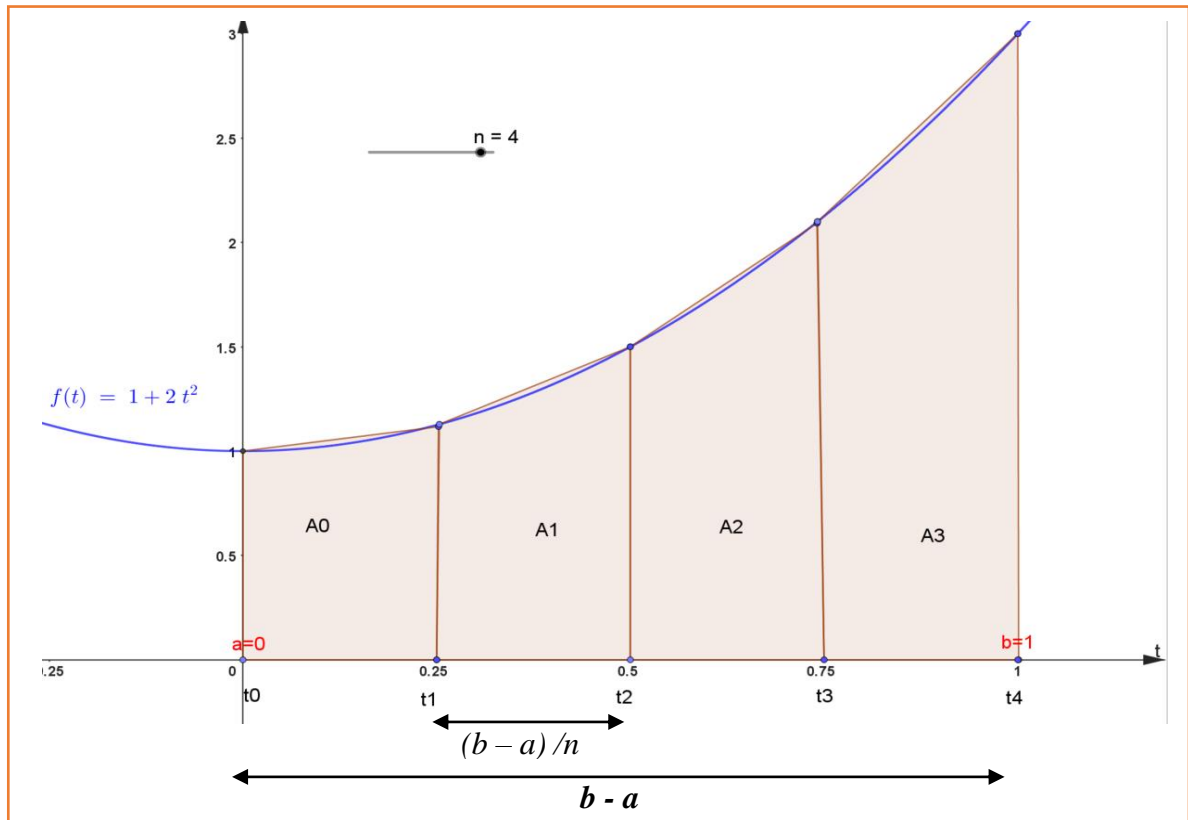


Figure 2: intégration par la méthode des trapèzes

Préparation théorique :

- 1) Calculer l'aire du trapèze A0, sur l'intervalle $J_0 = [t_0 ; t_1]$.

On a $t_0 = 0$ et $t_1 = 0.25$. $f(t_0) = 1$ et $f(t_1) = 1 + 2 \cdot 0.25^2 = 1.125$

Donc $A_0 = ((1 + 1.125) \cdot 0.25) / 2 = 265.625 \cdot 10^{-3}$

- 2) Calculer l'aire du trapèze A3, sur l'intervalle $J_3 = [t_3 ; t_4]$.

On a $t_3 = 0.75$ et $t_4 = 1$. $f(t_3) = 2.125$ et $f(t_4) = 3$.

Donc $A_3 = ((2.125 + 3) \cdot 0.25) / 2 = 640.625 \cdot 10^{-3}$

- 3) De façon plus générale, que vaut l'aire du trapèze A_i dans l'intervalle $J_i = [t_i ; t_{i+1}]$ pour $i \in \langle 0 ; n-1 \rangle$? Exprimer le résultat en fonction de $dt = \frac{b-a}{n}$, $f(t_i)$ et $f(t_{i+1})$

De manière générale, l'aire du trapèze vaut $A_i = \frac{(f(t_i) + f(t_{i+1})) \cdot dt}{2} = \frac{(f(t_i) + f(t_{i+1})) \cdot \frac{b-a}{n}}{2}$

- 4) Montrer que la formule du calcul approché de l'intégrale I par la méthode des trapèzes est :

$$T = \frac{b-a}{n} \times \sum_{i=0}^{n-1} \frac{f(t_i) + f(t_{i+1})}{2} \quad (\text{eq 2})$$

À faire

- 5) (*) Montrer que cette formule peut se mettre sous la forme suivante :

$$T = \frac{b-a}{n} \times \left(\frac{f(t_0) + f(t_n)}{2} + \sum_{i=1}^{n-1} f(t_i) \right)$$

6) Pour $n=4$, calculer la valeur approchée de I , nommée T , par la méthode des trapèzes.

$$T = (1-0)/4 * ((1+3)/2 + 1.125 + 1.5 + 2.125) = 1.6875$$

Réalisation :

👉 **Implanter** la méthode des trapèzes sur Matlab.

À faire

V / Méthode de Simpson

C'est une des méthodes numériques des plus performantes.

👉 En faisant des recherches sur Internet (par exemple sur le site https://fr.wikipedia.org/wiki/M%C3%A9thode_de_Simpson), expliquer le principe de cette méthode.

La méthode de Simpson est une technique d'intégration numérique utilisée pour estimer la valeur d'une intégrale. Elle se base sur l'approximation de la fonction à intégrer par des polynômes. Elle utilise trois points au lieu de deux.

👉 Facultatif : Implanter cet algorithme puis comparer la vitesse de convergence avec celles des algorithmes précédents.

VI / Facultatif : Méthode de Monte-Carlo (*)

Principe de la méthode de Monte Carlo :

Cette méthode consiste dans un premier temps à considérer un rectangle (en bleu sur la figure 3) qui inclue la surface de l'intégrale à calculer. La surface S_R du rectangle est facile à calculer.

On génère aléatoirement n points dans ce rectangle.

Parmi ces n points, on détermine le nombre de points inclus dans la zone d'intégration (en rouge). On note n_1 ce nombre de points.

L'aire de la surface est approximée par : $I_{Monte Carlo} = \frac{n_1}{n} \cdot S_R$

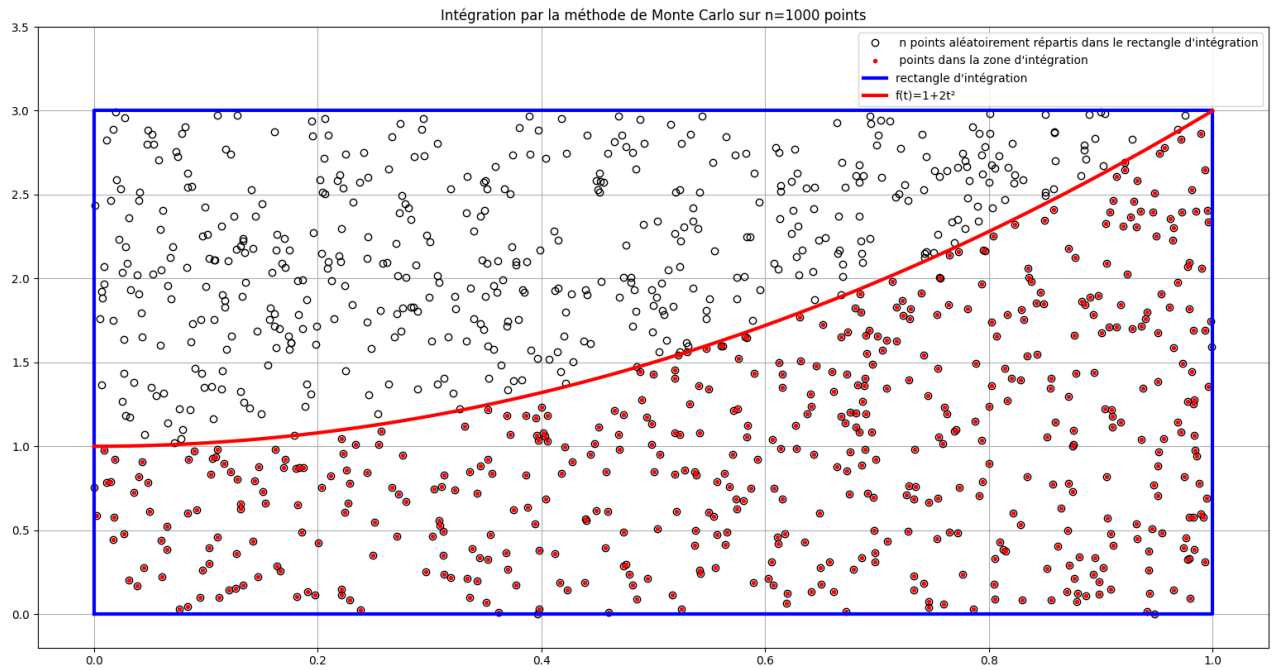


Figure 3: intégration par la méthode de Monte Carlo

Réalisation :

✎ Implanter le calcul de l'intégrale I en utilisant la méthode de Monte-Carlo

Remarque : cette méthode est particulièrement adaptée au calcul d'intégrales multidimensionnelles (volumes...). En 2D, elle a une vitesse de convergence très faible.

Partie 3 : Etude d'un signal temporel

Objectif : on se propose d'analyser sous Matlab un signal acquis à l'aide d'un oscilloscope. On souhaite le caractériser en effectuant les calculs suivants :

- Mesure de l'amplitude de sa tension.
- Calcul de la valeur moyenne et de la valeur efficace de sa tension.
- Détermination du spectre en amplitude.
- Détermination du fondamental et des harmoniques.
- Calcul de la puissance moyenne en utilisant Parseval.

I/ Acquisition des signaux

- ✎ Générer à l'aide d'un GBF un signal $x(t)$, sinusoïdal de fréquence 1000Hz et d'amplitude 5V.
- ✎ Observer ce signal à l'oscilloscope et le capturer à l'aide du logiciel « Keysight Benchvue ».
- ✎ Exporter ce signal en fichier Matlab. Pour cela, aller sur « **Données de trace** » puis « **Exporter** » (cf. Figure 1). On nommera le fichier exporté « **sinus.mat** ».

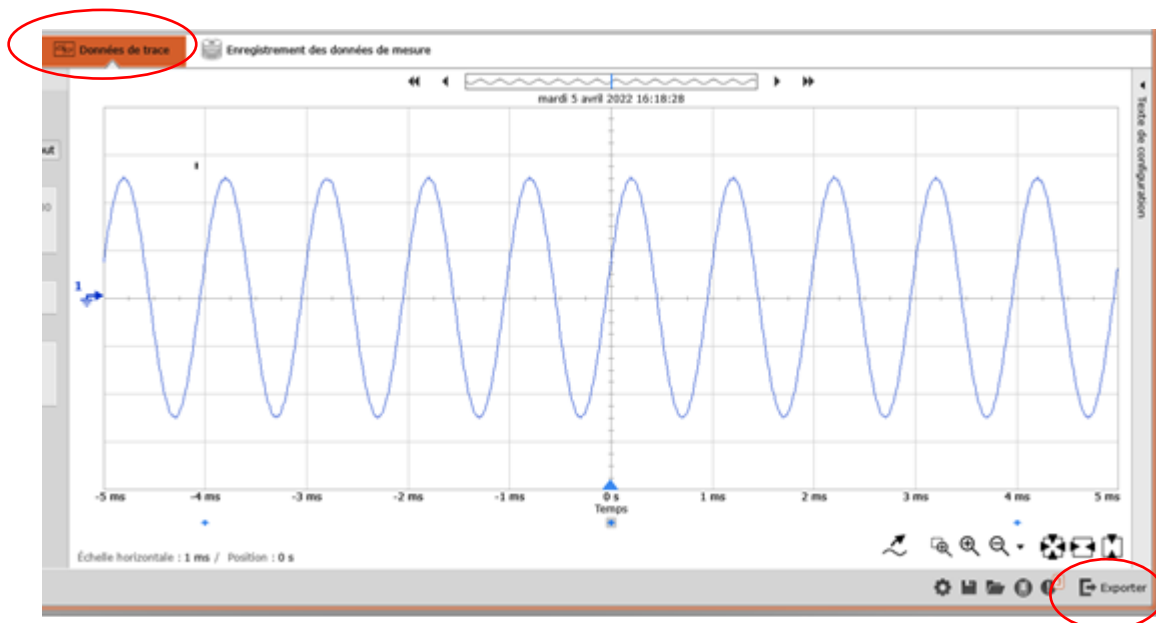


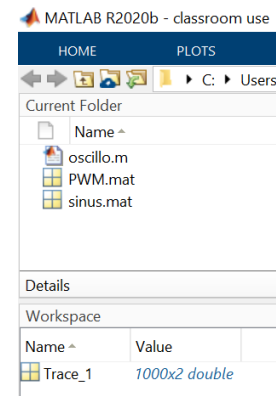
Figure 1 : export du signal de l'oscilloscope

- ✎ Recommencer en générant maintenant $y(t)$: signal carré 0-5V de fréquence 100kHz et de rapport cyclique $R_c = 0,2$. Le fichier importé sera nommé « **PWM.mat** ».

II / Etude du signal sinusoïdal : $x(t)$

II.1 / Extraction et affichage

- ✎ Créer un nouveau script Matlab, nommé « oscillo.m » et l'enregistrer dans un répertoire identifié.
- ✎ Dans ce répertoire ajouter les fichiers « sinus.mat » et « PWM.mat » importés précédemment.
- ✎ Double-cliquer sur le fichier « sinus.mat » afin de créer un tableau 2D, ici nommé : « Trace_1 » (cf. Workspace). (remarque : il existe d'autres méthodes pour réaliser cette opération).
- ✎ Double-cliquer sur « Trace_1 » dans le « Workspace » afin de voir la structure du tableau.



Le tableau Trace_1 a cette structure :

	1	2
1	-0.0050	2.0905
2	-0.0050	2.3317
3	-0.0050	2.6533
4	-0.0050	2.8945
5	-0.0050	3.1357
6	-0.0049	3.3769
7	-0.0049	3.6181
8	-0.0049	3.8593

Figure exp 20 : Structure du tableau « Trace_1 »

- ✎ Récupérer les données avec la procédure suivante :

```
1 - temp=transpose(Trace_1);
2 - t=temp(1,:);
3 - x=temp(2,:);
```

- 1) Quel est le rôle de la commande « transpose » ?

La commande « transpose » permet de prendre la transposée du vecteur Trace_1.

- ✎ Afficher le signal $x(t)$. (Commande plot).

Avec la commande `plot(t,x)` :

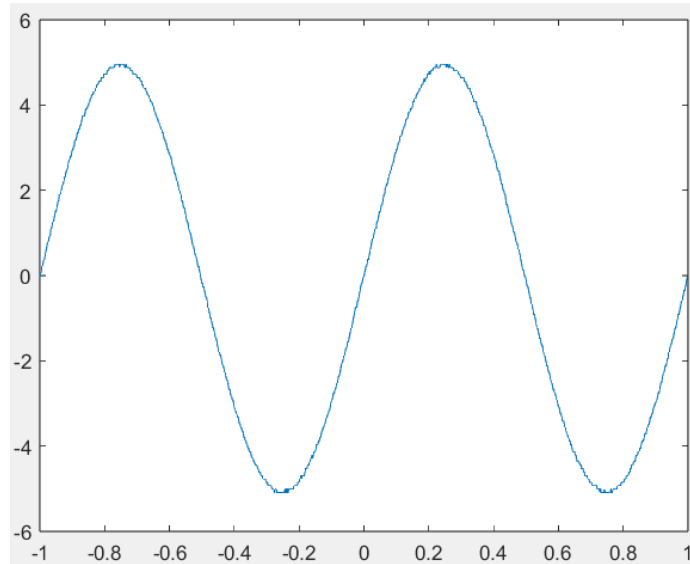


Figure exp 21 : observation du signal sinus.mat sur MATLAB

✎ Récupérer le nombre d'échantillons, N_s , de ce signal à l'aide de la commande « size ».

Avec la commande « size », nous devons faire :

```
Ns=size(x);
```

Figure exp 22 : commande pour avoir la taille du tableau x

✎ A partir du tableau des temps, calculer la période d'échantillonnage : T_e .

Pour calculer la période d'échantillonnage, il faut faire la commande :

```
Te = temp(1,2)-temp(1,1)
```

On trouve donc une période d'échantillonnage de $10\ \mu\text{s}$ ($1 \cdot 10^{-5}\text{ s}$).

Remarque : N_s et T_e seront utiles pour la suite...

II.2 / Exploitation des données du signal $x(t)$

II.2.a / Calcul de l'amplitude

✎ En utilisant les fonctions « max » et « min » calculer l'amplitude de ce signal.

Pour trouver l'amplitude de ce signal, nous devons prendre le max et le min de ce signal. La commande résultante est :

```
amp=(max(x)-min(x))/2;
```

Figure exp 23 : calcul pour trouver l'amplitude de $x(t)$

De ce fait nous trouvons un résultat de : **amp** **5.1055**

Le résultat concorde avec ce que nous avons généré au GBF.

II.2.b / Représentation du spectre en amplitude et détermination du fondamental

✎ En utilisant la fonction « spectre » vue précédemment, calculer puis afficher le spectre en amplitude de ce signal $x(t)$.

Le code à rentrer pour calculer et afficher le spectre du signal $x(t)$ est :

```
[x_spe f]=spectre(x,Fe,Ns);

subplot(2,1,2);
plot(f/1000,x_spe);
xlabel('F(kHz)')
ylabel('Volt')
legend('x(t)=5scos(2pi.1000.t)')
grid on
axis([0 5 0 5])
```

Figure exp 24 : code pour afficher le spectre de $x(t)$

Avec $F_e = 1/T_e$.

En faisant un subplot, nous observons le signal temporel et le spectre en fréquence :

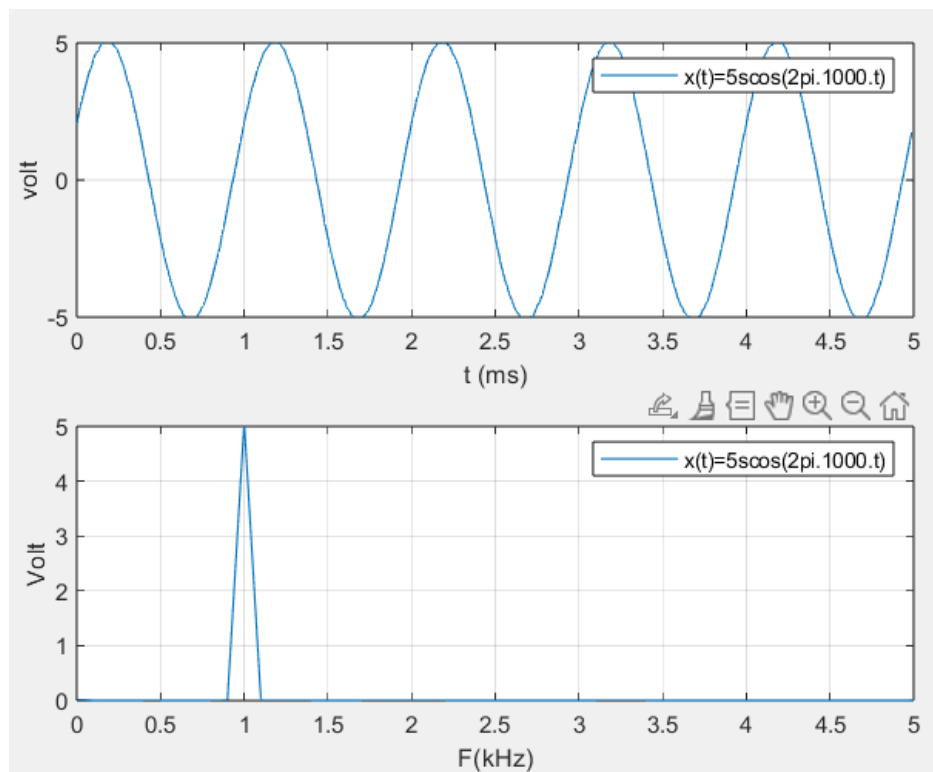


Figure exp 25 : observation du signal et du spectre de $x(t)$

Le spectre concorde avec le signal en temporel.

✎ A l'aide de la fonction « max » déterminer l'amplitude du fondamental de $x(t)$.

Nous utilisons la fonction max sur le spectre et trouvons donc une amplitude de :


 amp_spectre 5.0708

Figure exp 26 : max de $x(t)$ en utilisant le spectre

Les valeurs concordent, elles sont quasiment équivalentes.

✎ Regarder l'aide de la fonction « max » et retrouver l'indice, dans le tableau de x , où se trouve ce maximum.

Pour trouver l'indice dans le tableau de x , il faut faire cette commande :

`[amp_spectre, index]=max(x_spe);`

Figure exp 27 : commande pour trouver l'indice du max dans $x(t)$

Nous trouvons la valeur 11, donc le max est à l'indice 11. Nous vérifions :

10	11	12	13	14
0.0045	5.0708	0.0036	0.0029	0.0031

Figure exp 28 : vérification de l'indice du max de $x(t)$

Donc, la commande précédente était bonne.

✎ En déduire une détermination de la valeur de la fréquence du fondamental : F_0 .

En utilisant le tableau f , nous cherchons la valeur à l'indice « index » : $F_0 = f(\text{index})$, nous obtenons donc :

 F0 1.0001e+03

Nous avons bien la fréquence du max (1kHz).

II.2.c / Calcul de la valeur moyenne

La valeur moyenne du signal est donnée par la formule : $x_{\text{moy}} = \frac{1}{T} \times \int_0^T x(t) \cdot dt$ où T représente la période de la *composante fondamentale*.

✎ A partir de la fréquence du fondamental, déterminer la période du signal : $T = \frac{1}{F_0}$

On trouve $T =$  T 9.9994e-04 = $1 \cdot 10^{-3}$ s.

✎ En déduire le nombre d'échantillons pour une période de $x(t)$ à l'aide de la fonction « round » : $N_{\text{sparperiode}} = M = \text{round}\left(\frac{T}{T_e}\right)$

Nous trouvons :  M 99

De ce fait, nous comprenons qu'il y a 99 échantillons par période de $x(t)$.

- ✎ Réaliser un programme Matlab qui calcule la valeur moyenne du signal en utilisant la méthode des rectangles.

Le calcul à réaliser est :

II.2.d / Calcul de la valeur efficace

La valeur efficace du signal est donnée par la formule : $x_{eff} = \sqrt{\frac{1}{T} \times \int_0^T x^2(t) \cdot dt}$

- ✎ Réaliser un programme Matlab qui calcule la valeur efficace du signal en utilisant la méthode des rectangles.

III / Etude du signal PWM : $y(t)$

III.1.a / Etude théorique du signal PWM

On considère le signal $y(t)$ suivant :

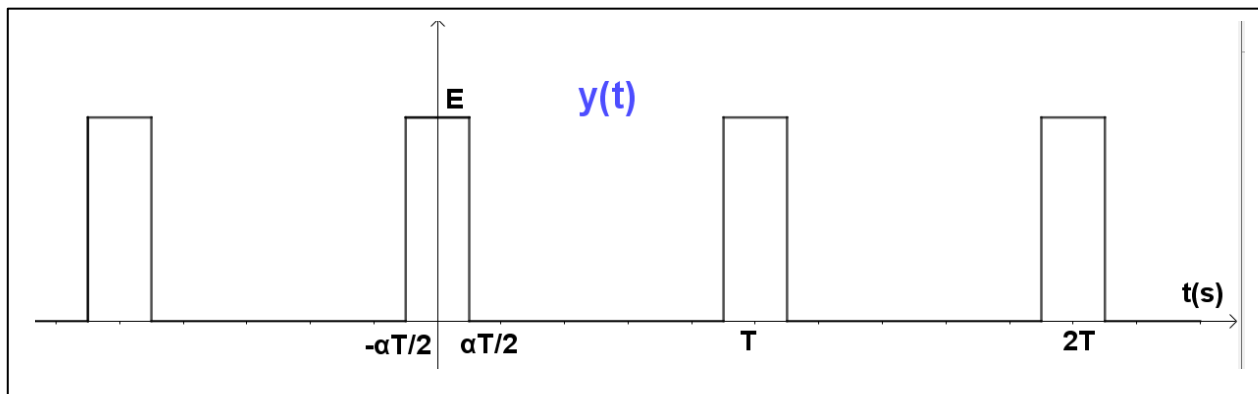


Figure 2: signal PWM

- 1) Calculer théoriquement la valeur moyenne et la valeur efficace de ce signal.

On admet que la décomposition en série de Fourier de ce signal est :

$$y(t) = \alpha E + \frac{2E}{\pi} \sin(\alpha\pi) \cdot \cos(\omega t) + \frac{2E}{2\pi} \sin(2\alpha\pi) \cdot \cos(2\omega t) + \frac{2E}{3\pi} \sin(3\alpha\pi) \cdot \cos(3\omega t) + \dots$$

Soit en utilisant le signe somme : $y(t) = \alpha E + \sum_{n=1}^{+\infty} \frac{2E}{n\pi} \sin(n\alpha\pi) \cdot \cos(n\omega t)$

Pour les applications numériques, on prendra $E=5V$; $\alpha=0,2$; $\omega = 2\pi \cdot 100 \cdot 10^3 \text{ rad/s}$.

- 2) Calculer la valeur numérique de la composante continue ainsi que l'amplitude du fondamental, et des harmoniques 2, 3 et 4.

- 3) Représenter le spectre en amplitude de ce signal jusqu'à $f = 400kHz$.
- 4) En utilisant Parseval, déterminer la puissance de ce signal (on se limitera à l'harmonique 4).

III.1.b / Etude sous Matlab du signal PWM

- ✎ Réaliser un programme Matlab qui calcule la valeur moyenne, la valeur efficace et le spectre du signal $y(t)$ importé de l'oscilloscope.
- ✎ Comparer les résultats avec ceux calculés dans la partie théorique.
- ✎ Retrouver les amplitudes de la composante continue, du fondamental et des N=14 premiers harmoniques.
- ✎ Appliquer le théorème de Parseval afin de déterminer la valeur efficace de ce signal.
- ✎ Conclure.

Partie 4 : Etude d'un canal Radio

Objectif : lors du TPn°3 des TP de Télécoms, vous avez acquis le spectre en réception d'une transmission radio. Cette transmission été effectuée autour d'une fréquence porteuse de $f_p=27\text{MHz}$.

On se propose d'analyser sous Matlab ce spectre et de la caractériser en effectuant les calculs suivants :

- Calcul de la bande occupée à 20dB
- Calcul de la densité spectrale de puissance (DSP)
- Détermination de la puissance totale du canal

I / Récupération des données

I.1 / Mise en forme du fichier Excel

👉 Récupérer le fichier que vous avez déposé sur l'ENT lors du TPn°3 de télécoms et l'enregistrer au format Excel. Le fichier doit ressembler au format ci-contre.

SN	1013013
UNIT_NAME	
TYPE	DATA
DESCR	/card0/usr/tem
DATE	2022-04-01-05-1
SETUP_FILE	1,00,000
PACKAGE_V	V3,02
BASE_VER	V5,89
APP_NAME	SPA
APP_VER	V7,14
APP_MODE	1
# Begin SPA Setup	

👉 Retrouver les informations utiles : plage de fréquence et Résolution BandWidth. L'unité pour les fréquences est le MHz...

Les données (fréquences et puissance en dBm) sont données plus bas dans la page Excel (cf. Figure 3). Ces données se trouvent au format texte concaténées sur la seule colonne B.

Presse-papiers		
E485		
A	B	C
482	# Begin SPA Data	
483		
484	# Begin TRACE A Data	
485	P_0	-73,608000 , 26,950001 ,MHz
486	P_1	-74,008000 , 26,950182 ,MHz
487	P_2	-74,888000 , 26,950364 ,MHz
488	P_3	-72,696000 , 26,950546 ,MHz
489	P_4	-73,548000 , 26,950728 ,MHz
490	P_5	-70,452000 , 26,950910 ,MHz

484	# Begin TRACE A Data	
485	P_0	-73,608000 , 26,950001 ,MHz
486	P_1	-74,008000 , 26,950182 ,MHz
487	P_2	-74,888000 , 26,950364 ,MHz
488	P_3	-72,696000 , 26,950546 ,MHz
489	P_4	-73,548000 , 26,950728 ,MHz
490	P_5	-70,452000 , 26,950910 ,MHz

Figure 3: données récupérées au format Excel

👉 Utiliser les fonctions Excel « GAUCHE » et « DROITE » afin de créer deux colonne (E et F ici) contenant la puissance et la fréquence.

Presse-papiers		
E485		
A	B	C
482	# Begin SPA Data	
483		
484	# Begin TRACE A Data	
485	P_0	-73,608000 , 26,950001 ,MHz
486	P_1	-74,008000 , 26,950182 ,MHz
487	P_2	-74,888000 , 26,950364 ,MHz
488	P_3	-72,696000 , 26,950546 ,MHz
489	P_4	-73,548000 , 26,950728 ,MHz
490	P_5	-70,452000 , 26,950910 ,MHz

				E	F
				PdBm (format texte)	Fréquence (format texte)
485	P_0	-73,608000 , 26,950001 ,MHz		-73,608000	26,950001
486	P_1	-74,008000 , 26,950182 ,MHz		-74,008000	26,950182
487	P_2	-74,888000 , 26,950364 ,MHz		-74,888000	26,950364
488	P_3	-72,696000 , 26,950546 ,MHz		-72,696000	26,950546
489	P_4	-73,548000 , 26,950728 ,MHz		-73,548000	26,950728
490	P_5	-70,452000 , 26,950910 ,MHz		-70,452000	26,950910

Le souci est que les colonnes E et F sont au format texte...

- ✎ Créer deux colonnes (I et J ici) dans lesquelles les données sont converties en nombres : il suffit d'ajouter 0 pour convertir une cellule texte en nombre (cf. création de la cellule I485).

I485										
	A	B	C	D	E	F	G	H	I	J
482	# Begin SPA Data									
483					PdBm (format texte)	Fréquence (format texte)			PdBm (format nombre)	Fréquence (format nombre)
484	# Begin TRACE A Data									
485	P_0	-73,608000 , 26,950001 ,MHz			-73,608000	26,950001			-73,608	26,950001
486	P_1	-74,008000 , 26,950182 ,MHz			-74,008000	26,950182			-74,008	26,950182
487	P_2	-74,888000 , 26,950364 ,MHz			-74,888000	26,950364			-74,888	26,950364

I.2 / Importation des données sur Matlab

Il faut maintenant importer ces données sur Matlab. Le fichier Excel doit se trouver dans le répertoire de votre projet Matlab.

- « Double-cliquer » sur le fichier Excel (ici « relevé_TP3.xls »)
- Sélectionner les cases de la page Excel à importer (ici les colonnes I et J).
- Choisir un type de sortie « Numeric Matrix »
- Cliquer sur import ; un tableau de données 'releveTP3' a dû être créé.
- Cliquer sur ce tableau pour visualiser son contenu...

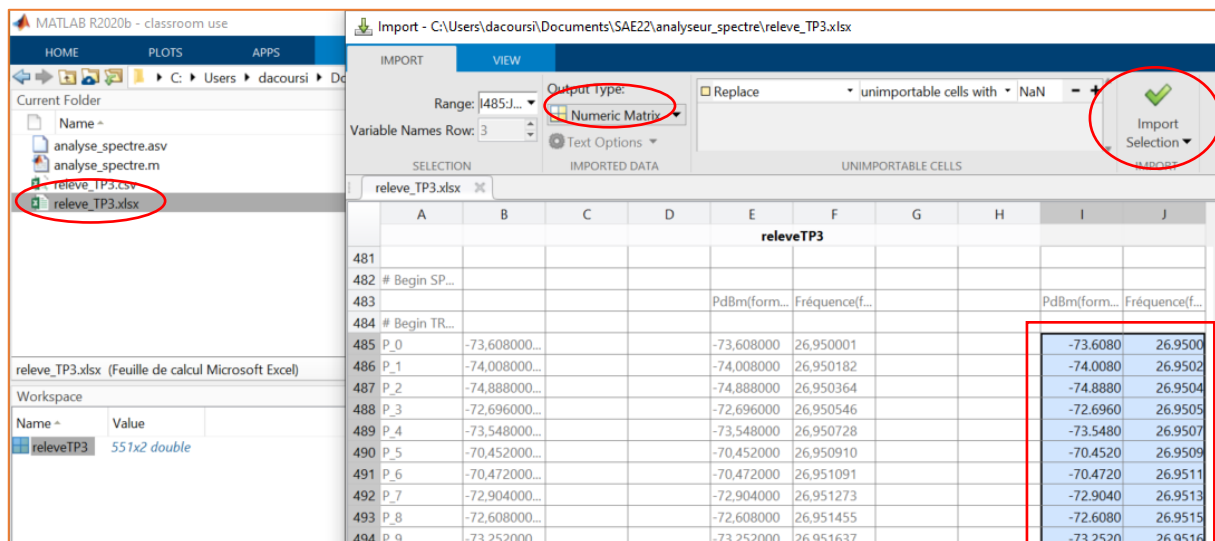


Figure 4: importation des données sur Matlab

I.3 / Visualisation des données

- ✎ Créer un script permettant d'afficher l'évolution de la puissance en fonction de la fréquence.

II / Lissage des données

Les données récupérées sont relativement bruitées. On va essayer de réduire ce bruit en appliquant un filtre appelé filtre de lissage ou filtre moyennneur. L'algorithme est le suivant :

Entrées :

P_{dBm} % tableau des puissances en dBm.
 N_s % nombre de cases de ce tableau
 $K = 4$ % ordre du filtre (on fera la moyenne sur $2K+1$ échantillons)

Initialisation :

Créer un tableau $P_{L\ dBm} = P_{dBm}$ % Ce tableau servira à stocker les données lissées.

Traitement :

Pour n allant de $K+1$ à N_s-K

$$P_{LdBm}(n) = \frac{\sum_{k=-K}^K P_{dBm}(n+k)}{2K+1}$$

fin

Sortie : $P_{L\ dBm}$

✎ Implanter cet algorithme et afficher le résultat.

✎ Faire varier l'ordre du filtre de lissage (paramètre K) et visualiser son effet.

III / Détermination de la bande occupée à -20dB

✎ En utilisant la fonction max, déterminer le niveau maximal de puissance dans ce relevé ainsi que la fréquence correspondante. Aide : la fonction « max » renvoie le maximum d'un tableau mais aussi l'indice correspondant...

✎ Compléter le programme afin de déterminer la bande occupée de ce canal à -20dB.

IV / Calcul de la puissance du canal

✎ Convertir le tableau des puissances, P_{dBm} , en un tableau exprimé en Watt : P_W

✎ En déduire un tableau des densités spectrales de puissances : $DSP = \frac{P_W}{RBW}$.

✎ Calculer la puissance du canal comme : $P_{canal} = \int_{f_{min}}^{f_{max}} DSF(f).df$

V / Présentation des résultats

✎ Utiliser les commandes « strcat » et « text » afin d'afficher les calculs de la bande occupée à 20dB et de la puissance canal.

On devra avoir un résultat analogue à celui de la Figure 5.

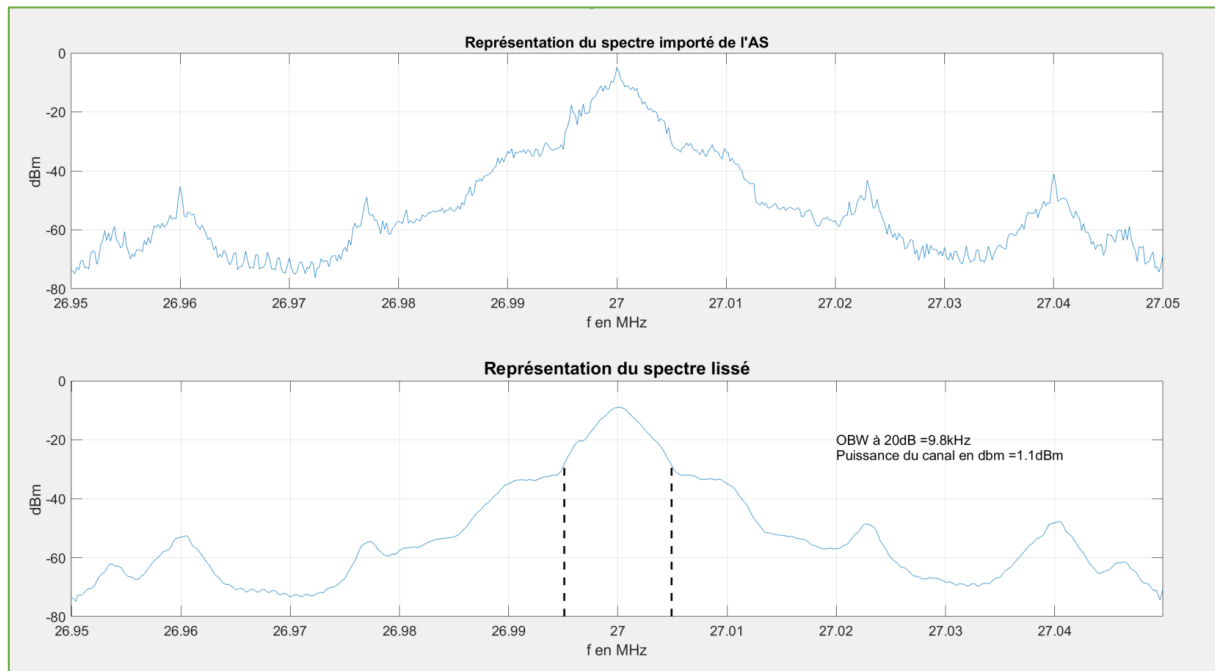


Figure 5: résultat final attendu