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 :

* 8 séances de 2h encadrées
* 4 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

# Formation Matlab : « Onramp »

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

# Génération d’un signal sinusoïdal x(t)

On désire générer et afficher le signaloù x(t) est une tension en Volts.

Matlab étant un logiciel numérique, il va devoir **échantillonner** ce signal. On choisira ***f*=40kHz** comme fréquence d’échantillonnage.

Le début du programme Matlab est le suivant :

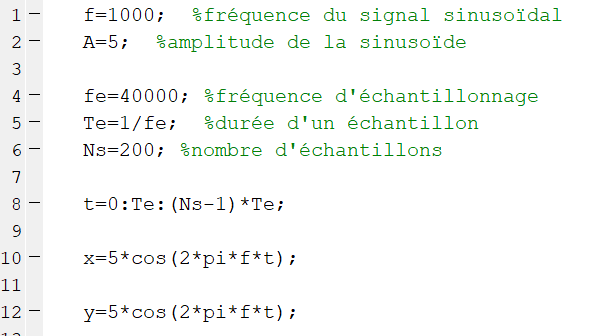


Figure 1: début du programme de génération d'un cosinus

##### Que représente la variable t ? Quelle est sa nature ?

##### Que fait la ligne 10 ? Quelle est la nature de  ?

##### Combien de période du signal seront calculées ?

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

* **Faire la saisie de ce programme** (Il est recommandé de travailler avec des « live script » qui est une forme de programmation qui permet de commenter et de mettre en forme le programme. De cette façon le compte-rendu de cette SAE sera partiellement fait). Nommer par exemple ce programme : P1sinus.mlx.

# 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

Il faut obtenir une figure similaire à la Figure 2 :

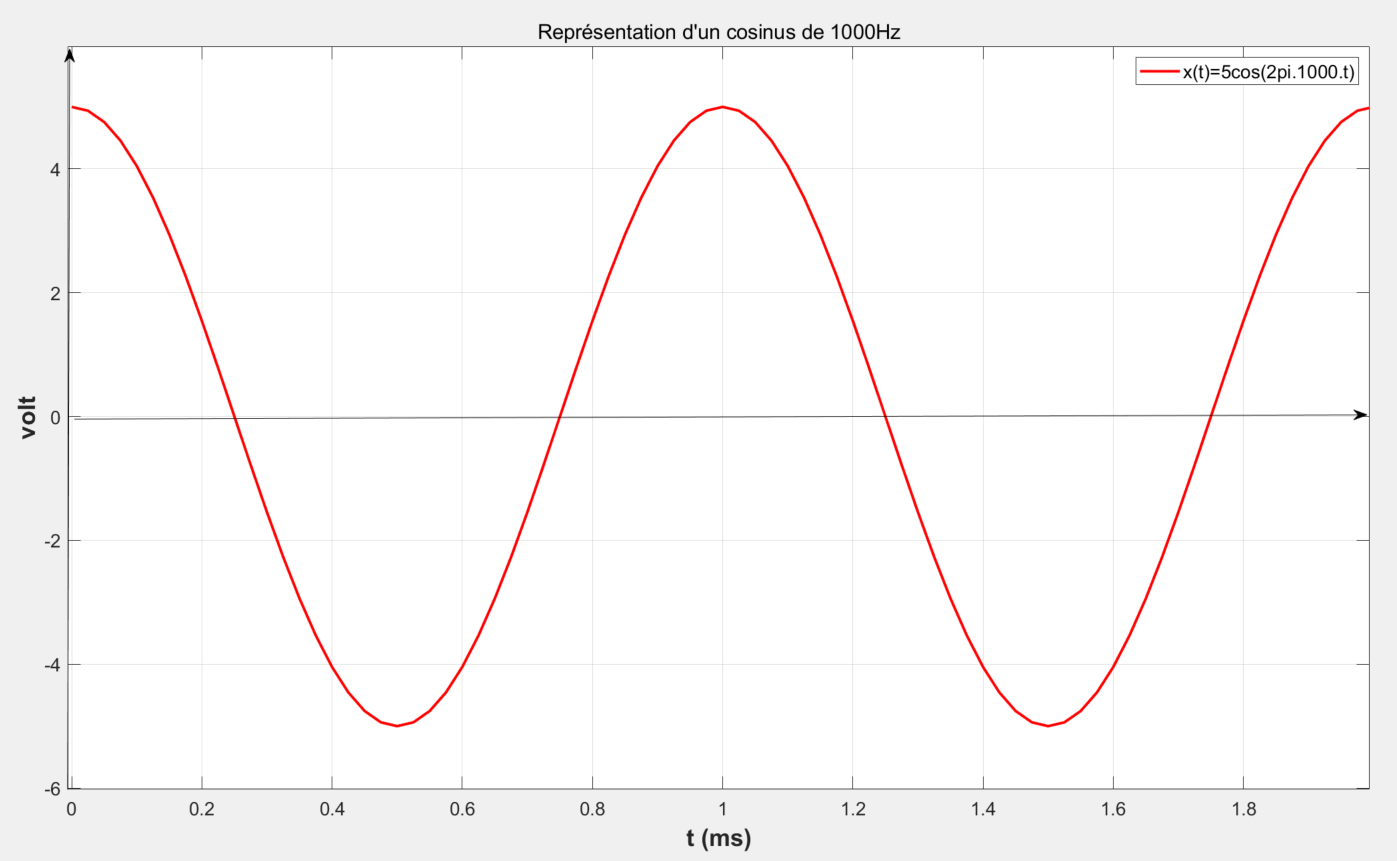


Figure 2 : affichage de la fonction cosinus x(t)

On veut maintenant créer un signal 

* Compléter le programme afin de créer et d’afficher le signal en bleu.
* Etudier la fonction « subplot » au moyen du menu « Help ».
* Modifier le programme afin d’afficher les deux courbes sur deux graphes sur la même figure et obtenir la représentation suivante :

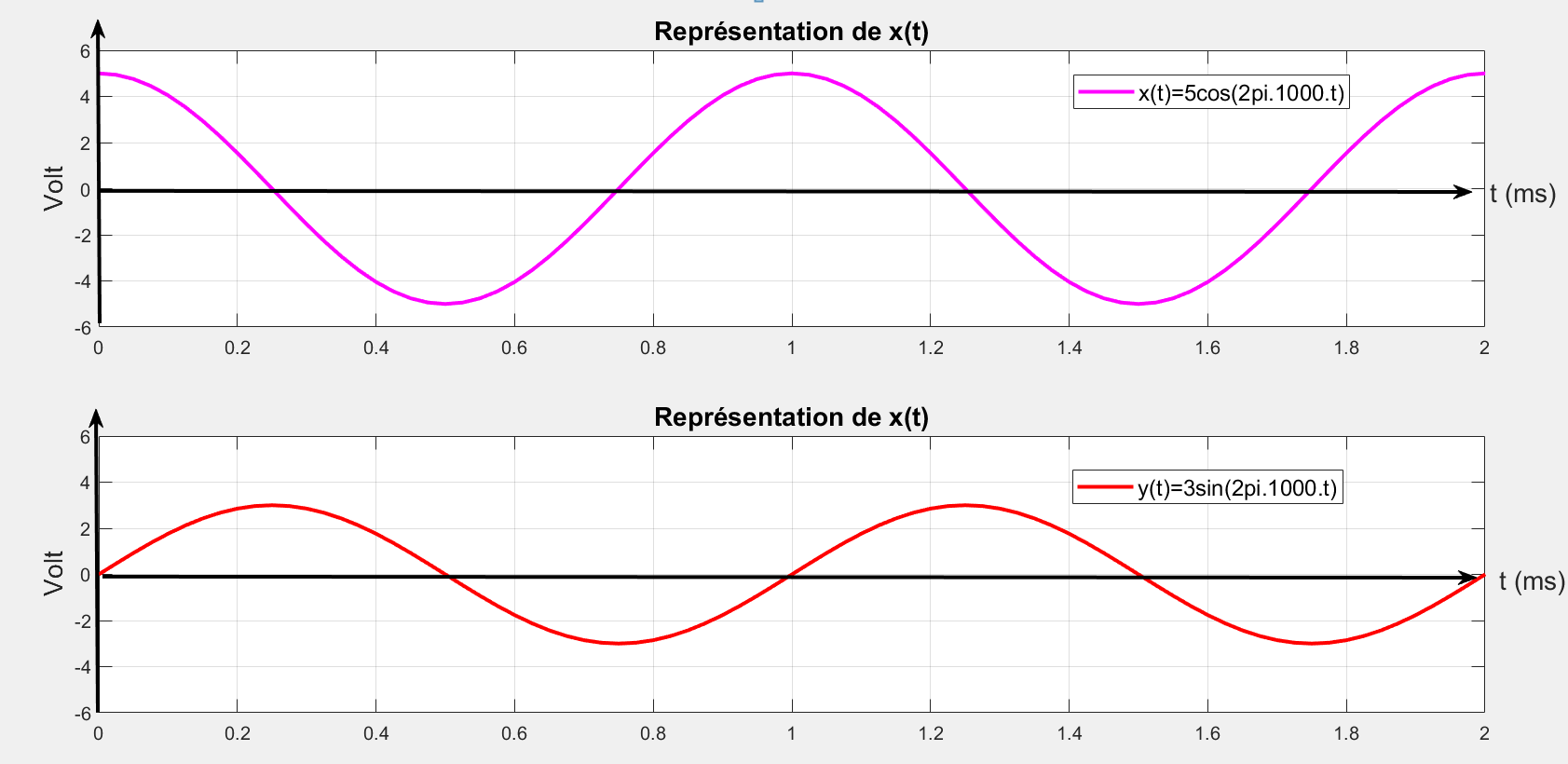


Figure 3 : représentation de deux signaux en utilisant la commande « subplot »

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 ».
* Compléter le programme pour obtenir la représentation suivante :

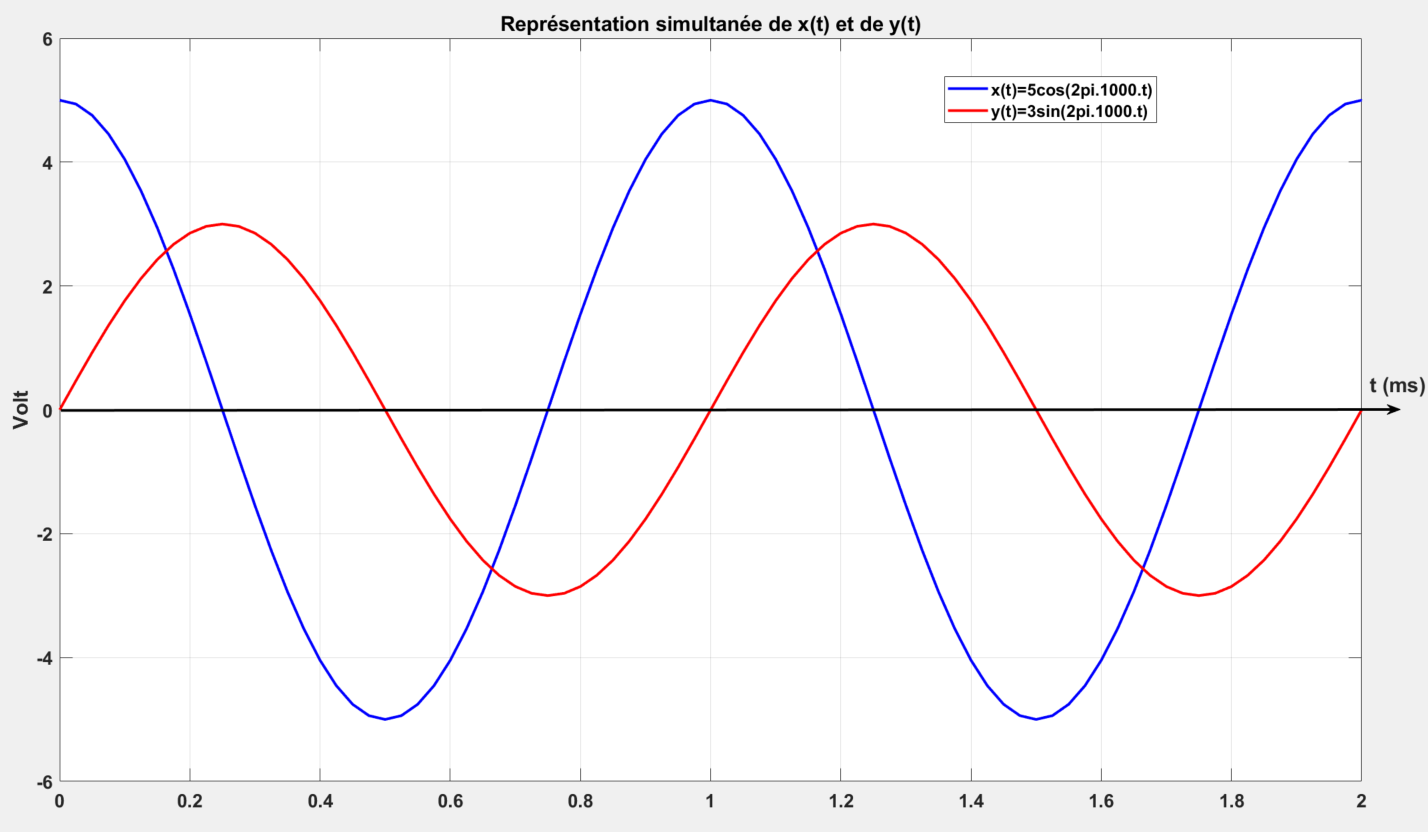


Figure 4 : représentation de deux signaux sur un même graphique

# Formation Matlab : suite

* Finir l’autoformation à Matlab Onramp.

# 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 »

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



##### Exécuter le script et valider le résultat.

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 . On ne calculera pas ici la valeur efficace pour le moment.
* Compléter le programme pour tracer le spectre en dBV.
* Valider le résultat obtenu.
* Réaliser l’affichage comme montré ci-dessous, en mieux (penser aux noms des axes et autres …)

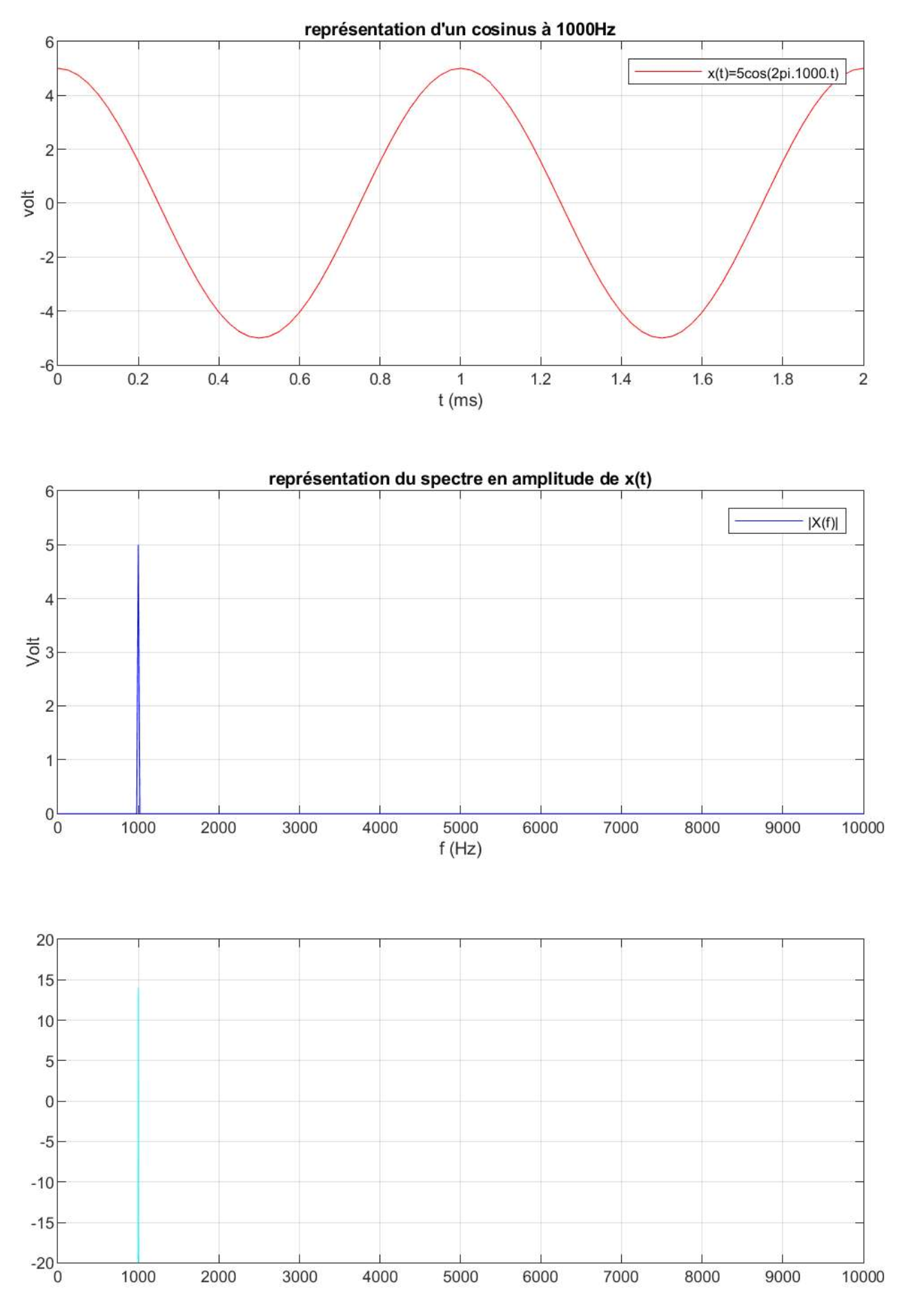


Figure 5 : représentation du signal x(t), de son spectre et de ce spectre en dBV

# 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 : *fech*=40kHz
* Nombre d’échantillons : Ns=2000

Les signaux à multiplier sont les suivants :

*  avec A1=5V et *f1*=1kHz
*  avec A2=3V et *f2*=3kHz
* Générer les deux signaux x1(t) et x2(t).

##### Vérifier la taille des vecteurs x1 et x2 créés. Justifier.

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

##### 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( ??) ….)

##### Calculer le produit de ces deux signaux :

##### Quelle est la dimension du vecteur z ? Justifier.

* Afficher sur une même figure, mais sur trois graphes les chronogrammes de x1(t), x2(t) et z(t).
* Calculer les spectres de ces trois signaux et afficher les sur trois graphes d’une nouvelle figure.

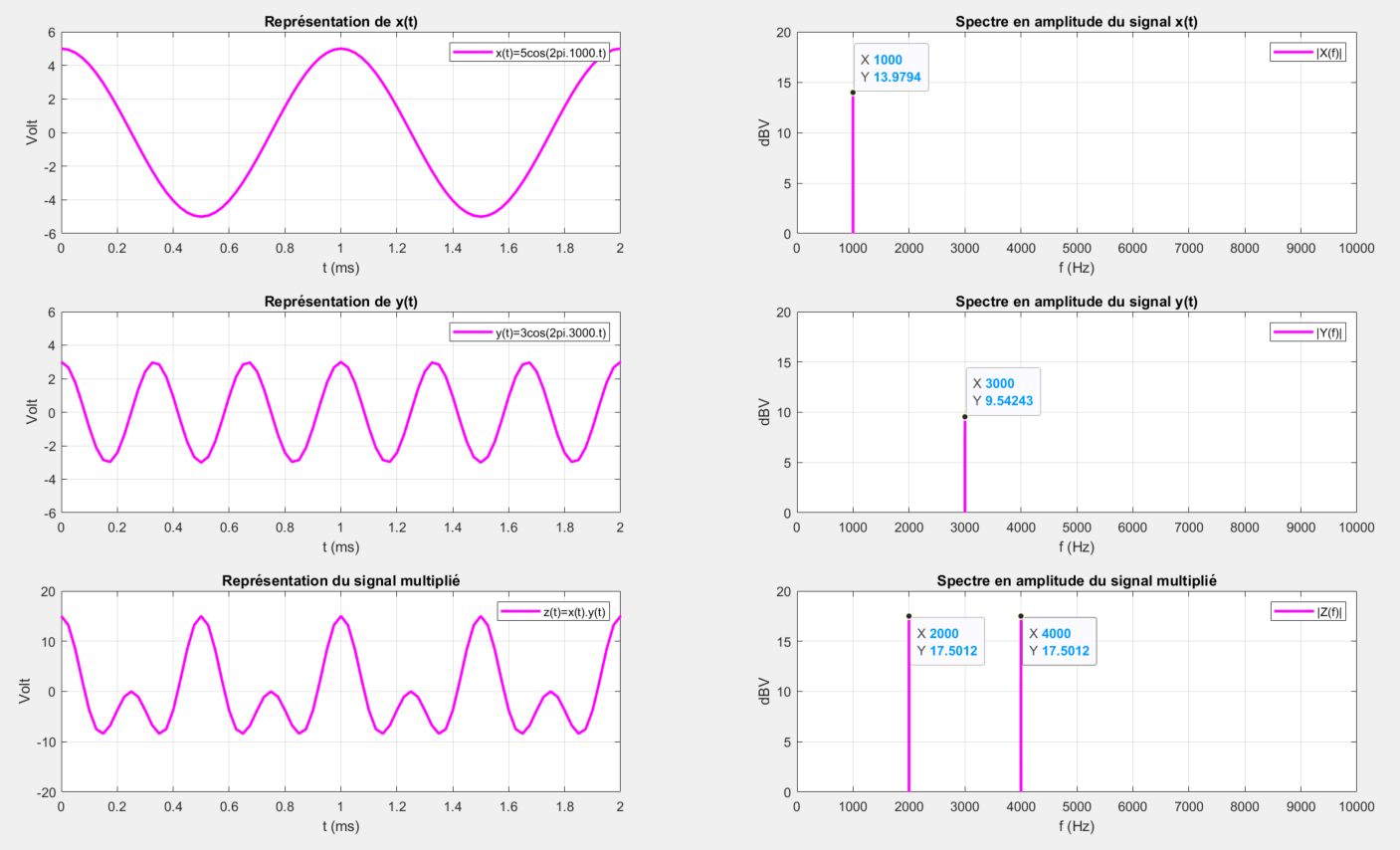
Les résultats obtenus doivent être de la forme suivante :

Figure 6: multiplication de deux signaux sinusoïdaux

##### Analyser les résultats obtenus et comparer avec les spectres attendus théoriquement.

##### Recommencer en prenant maintenant . Analyser. Quelle est la fonction réalisée ?!

Partie 2 : Calcul numérique d’une intégrale

# Présentation du contexte

Soit une fonction continue définie sur un intervalle . On souhaite évaluer numériquement l’intégrale : 

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 est découpé en **n sous-intervalles** de longueur (cf. Figure 1).

Cela revient en fait à échantillonner la fonction à la fréquence .

Le iième échantillon sera ainsi à l’abscisse : où i est un nombre entier compris entre 0 et n , ce qui s’écrit : . 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 par un polynôme.

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

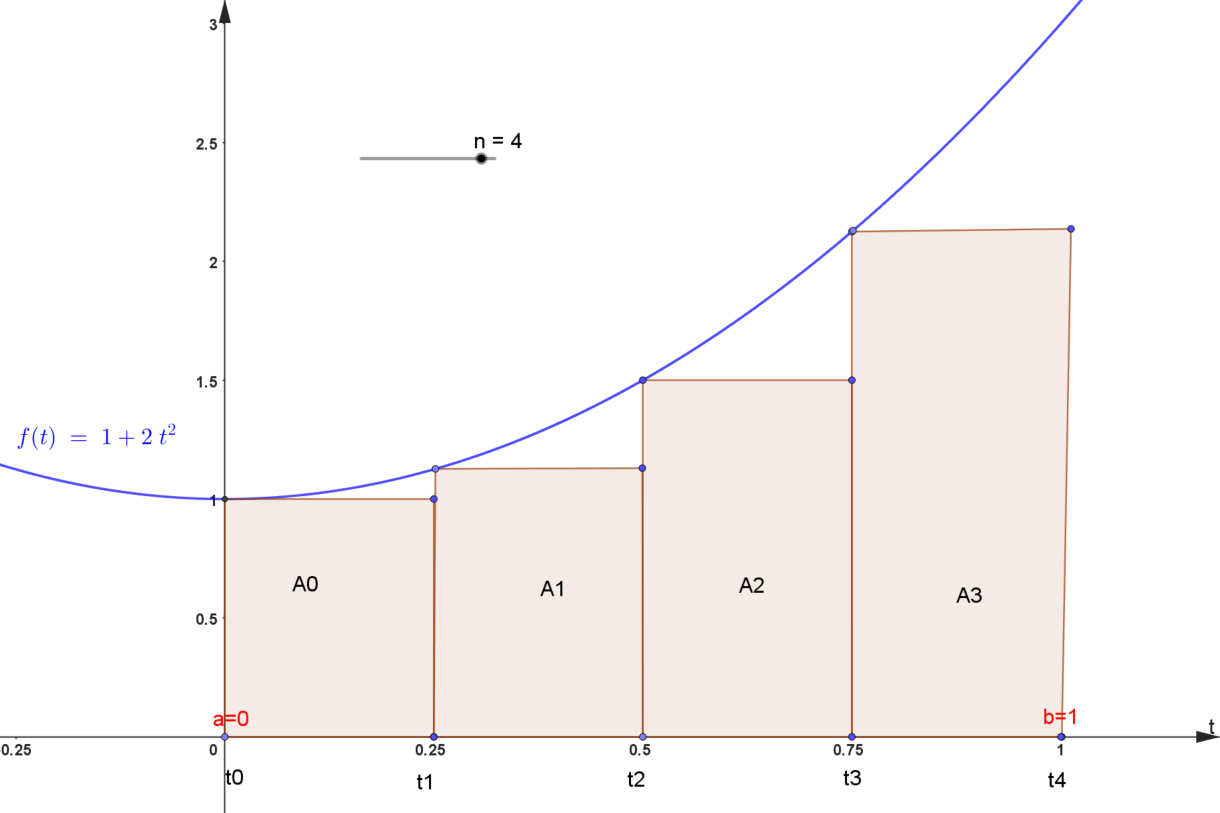


Ceci sera fait par plusieurs méthodes différentes

# Méthode des rectangles

La méthode des rectangles consiste à remplacer sur chaque sous-intervalle , la fonction par .

La Figure 1 illustre le calcul de  par la méthode des rectangles avec **n=4** segments.



***b - a***

***(b – a) /n = dt***

Figure 1: intégration par la méthode des rectangles

Ainsi, selon cette approximation :

Avec :

**Préparation théorique** :

##### Calculer théoriquement la valeur de I. Faire une application numérique à 10-5 près.

##### 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).

##### Montrer que de manière générale la formule du calcul approché de l’intégrale I par la méthode des rectangles sur n intervalles est :.

**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 …*

# Facultatif : vitesse de convergence (\*)

Dans cette partie, on prendra toujours .

##### 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 :

##### 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 :, la valeur maximale de sur l’intervalle .

##### Déterminer la formule de .

##### Montrer que :

##### Vérifier par simulation sur Matlab la formule théorique de majoration d’erreur :



##### Vérifier de même que l’erreur commise par la méthode des trapèzes est majorée selon la formule suivante :

##### 

##### Faire les simulations nécessaires afin de comparer les vitesses de convergences de ces 2 méthodes.

# Méthodes des trapèzes

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

*(b – a) /n*

***b - a***

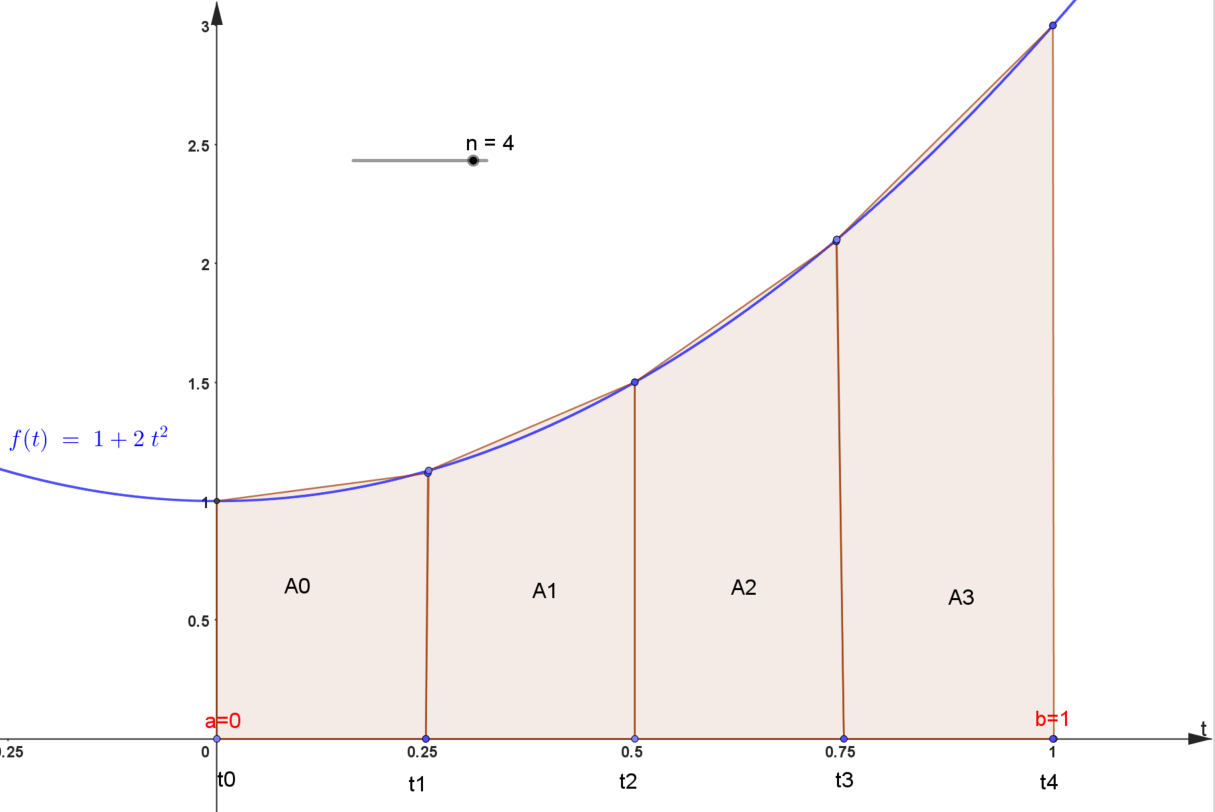


Figure 2: intégration par la méthode des trapèzes

**Préparation théorique** :

##### Calculer l’aire du trapèze A0, sur l’intervalle .

##### Calculer l’aire du trapèze A3, sur l’intervalle .

##### De façon plus générale, que vaut l’aire du trapèze Ai dans l’intervalle pour  ? Exprimer le résultat en fonction de ,

##### Montrer que la formule du calcul approché de l’intégrale I par la méthode des trapèzes est :

 (eq 2)

##### (\*) Montrer que cette formule peut se mettre sous la forme suivante :



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

**Réalisation** :

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

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

# 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 SR 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 n1 ce nombre de points.

L’aire de la surface est approximée par :

Une image contenant texte, capture d’écran, ligne, Tracé

Description générée automatiquement

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.

# Acquisition des signaux

* Générer à l’aide d’un GBF un signal , 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** ».

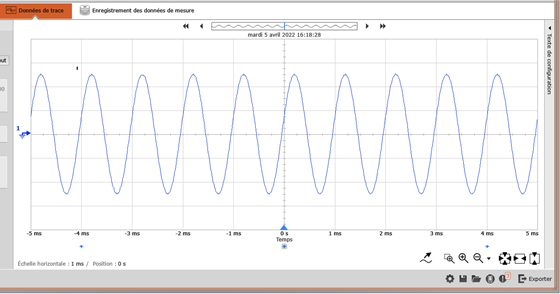
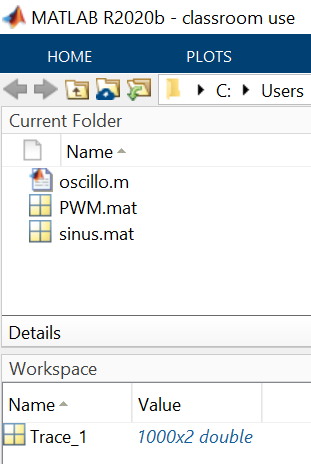


Figure 1 : export du signal de l'oscilloscope

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

# Etude du signal sinusoïdal : x(t)

## 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.
* Récupérer les données avec la procédure suivante :



##### Quel est le rôle de la commande « transpose » ?

* Afficher le signal x(t). (Commande plot).
* Récupérer le nombre d’échantillons, Ns, de ce signal à l’aide de la commande « size ».
* A partir du tableau des temps, calculer la période d’échantillonnage : Te.

*Remarque* : Ns et Te seront utiles pour la suite…

## Exploitation des données du signal x(t)

### Calcul de l’amplitude

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

### 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).
* A l’aide de la fonction « max » déterminer l’amplitude du fondamental de x(t).
* Regarder l’aide de la fonction « max » et retrouver l’indice, dans le tableau de x, où se trouve ce maximum.
* En déduire une détermination de la valeur de la fréquence du fondamental : F0.

### Calcul de la valeur moyenne

La valeur moyenne du signal est donnée par la formule :  où T représente la période de la *composante fondamentale*.

* A partir de la fréquence du fondamental, déterminer la période du signal : 
* En déduire le nombre d’échantillons pour une période de x(t) à l’aide de la fonction « round » :  
* Réaliser un programme Matlab qui calcule la valeur moyenne du signal en utilisant la méthode des rectangles.

### Calcul de la valeur efficace

La valeur efficace du signal est donnée par la formule : 

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

# Etude du signal PWM : y(t)

### Etude théorique du signal PWM

On considère le signal y(t) suivant :

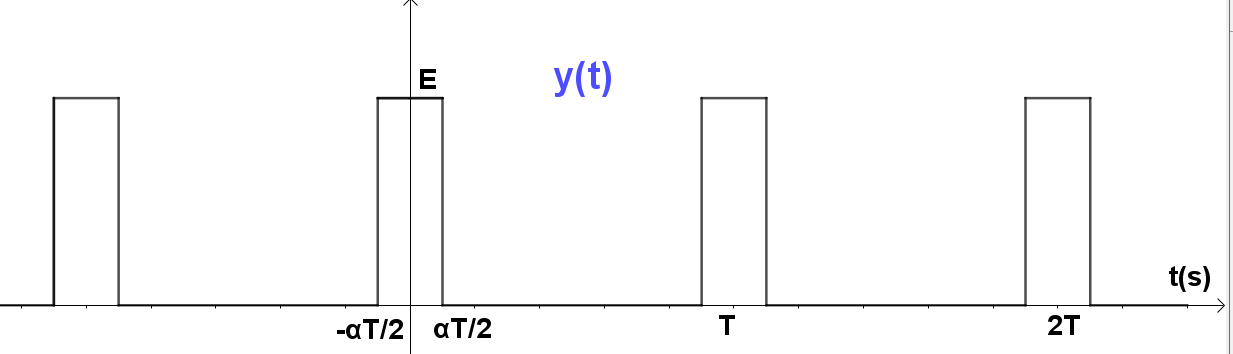


Figure 2: signal PWM

##### 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 :



Soit en utilisant le signe somme : 

Pour les applications numériques, on prendra **E=5V ; α=0,2 ;** .

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

##### Représenter le spectre en amplitude de ce signal jusqu’à .

##### En utilisant Parseval, déterminer la puissance de ce signal (on se limitera à l’harmonique 4).

### 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 fp=27MHz.

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

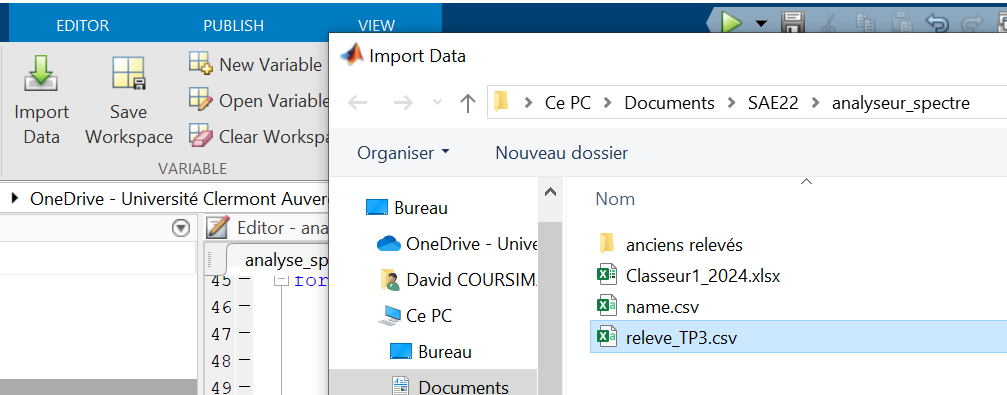
# Récupération des données

## Importation des données sur Matlab

Il faut importer les données sauvegardées pendant la séance de TP de Télécoms sur Matlab.

###### Enregistrer le fichier dans le répertoire de votre projet Matlab.

###### Importer les données du fichier comme indiqué dans la fifure suivante :



###### Sélectionner les à importer puis choisir le type de sortie ‘column vector’

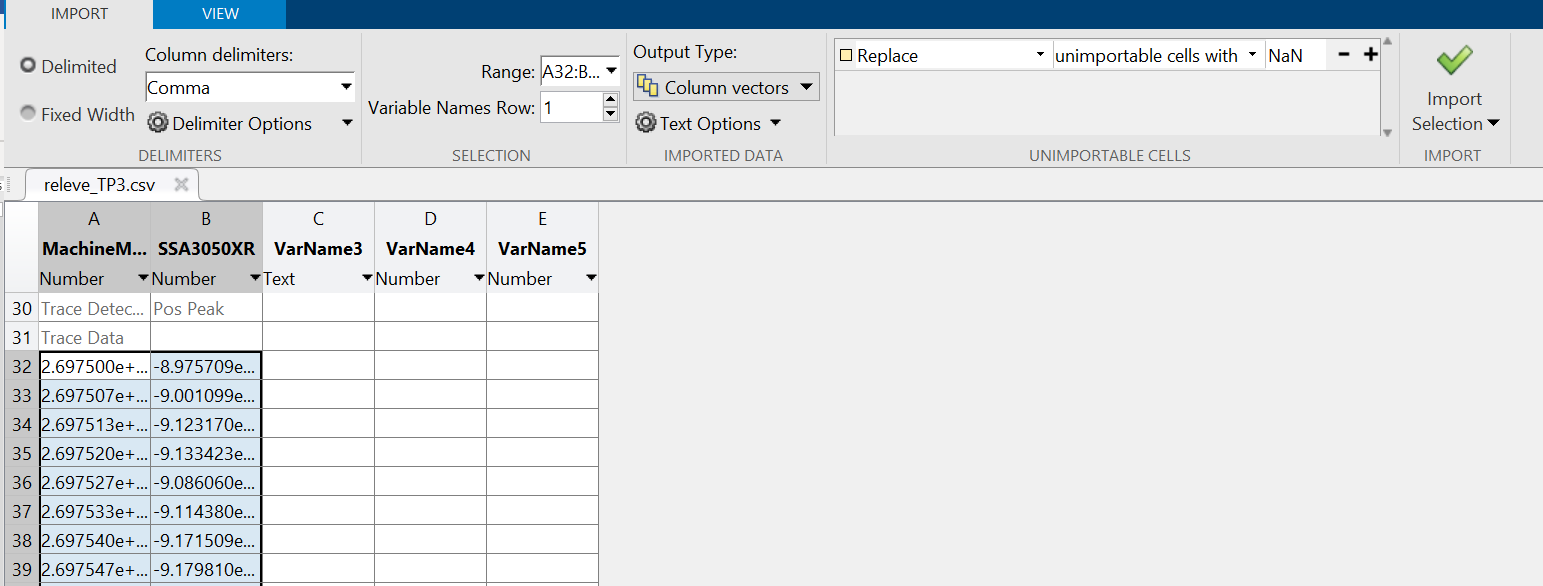


Figure 3: importation des données sur Matlab

###### Importer ensuite les données

###### Deux vecteurs colonnes ont dû être crées dans le Workspace. Cliquer sur ces vecteurs pour visualiser leurs contenus…

## Visualisation des données

* Créer un script permettant d’afficher l’évolution de la puissance en fonction de la fréquence.

# 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 moyenneur. L’algorithme est le suivant :

**Entrées :**

PdBm  % tableau des puissances en dBm.

Ns  % nombre de cases de ce tableau

K =4 % ordre du filtre (on fera la moyenne sur 2K+1 échantillons)

**Initialisation :**

Créer un tableau PL dBm =PdBm % Ce tableau servira à stocker les données lissées.

**Traitement :**

**Pour** n allant de K+1 à Ns-K



**fin**

**Sortie :** PL dBm

* Implanter cet algorithme et afficher le résultat.
* Faire varier l’ordre du filtre de lissage (paramètre K) et visualiser son effet.

# 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.

# Calcul de la puissance du canal

* Convertir le tableau des puissances, PdBm, en un tableau exprimé en Watt : PW
* En déduire un tableau des densités spectrales de puissances :.
* Calculer la puissance du canal comme : 

# 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 4.



Figure 4: résultat final attendu