# TP n° 1

# Introduction à Matlab

## Introduction

Matlab est un environement de calcul et de visualisation pout l'Ingénieur. Il offre la posibilité de visualiser graphiquement des données et d'effectuer sur celles-ci des opérations d'analyse et traitement du signal. D'autre part, ce logiciel permet de réaliser simplement des simulations numériques grâce à son langage de programmation très simple d'accès. Le but de ce T.P. est d'étudier, dans un premier temps, les principes de base de Matlab et dans un deuxième temps d'étudier l'incidence de la fréquence d'échantillonnage sur l'analyse d'un signal périodique (thérème de Shannon).

## 1.1 Initiation

### 1.1.1 Les commandes en ligne

A l'ouverture de l'application Matlab, on a accès directement à une application console, c'est à dire acceptant directement des commandes (commandes en ligne).

– La commande help

  Matlab est in logiciel autocommandé: les informations relatives à une commande peuvent être obtenues en composant **help** suivi du nom de la commande.
  $>>$ *help*$\hookleftarrow$ Affiche sur l'écran les bibliothèques disponibles.
  $>>$ *help signal* $\backslash$*signal* $\hookleftarrow$ Affiche la boîte à outils signal disponibles.
  $>>$ *help conv* $\hookleftarrow$ Affiche les renseignements sur la fonction de convolution.
  $>>$ *help fft* $\hookleftarrow$ Affiche les renseignements sur la fonction Transformée de Fourier discrète.

– Quelques opérations élémentaires

  $>>$ $A = 2$ $\hookleftarrow$ Affectation de la valeur 2 à la variable A avec ècho.
  $>>$ $B = 3$ $\hookleftarrow$ Affectation de la valeur 3 à la variable B avec ècho.
  $>>$ $C = 5;$ $\hookleftarrow$ Affectation de la valeur 5 à la variable C sans ècho.
  $>>$ $d = A * B + C$ $\hookleftarrow$ Opérations arithmètiques élémentaires $+, *, /, ...$

– Création de Vecteurs et Matrices

  $>>$ $V1 = [1\,2\,3\,4]$ $\hookleftarrow$ Construction du vecteur V1.
  $>>$ $V2 = V1'$ $\hookleftarrow$ V2 est le transposé de V1.
  $>>$ $V3 = V1. * V2$ $\hookleftarrow$ Donne le produit élément par élément.
  $>>$ $V4 = V2 * V1$ $\hookleftarrow$ Donne le produit matriciel.
  $>>$ *who* $\hookleftarrow$ Liste les variables courantes.
  $>>$ *whos* $\hookleftarrow$ Donne la liste des variables courantes et spécifie leur taille.
  $>>$ *clear all* $\hookleftarrow$ Efface toutes les varibales courantes.
  **Attention:** Matlab distingue les majuscules des minuscules ($A \neq$ a).

  Le logiciel Matlab est conçu pour la manipulation de matrices et de vecteurs (un vecteur de longeur $N$ est une matrice de dimension $1 \times N$, c'est à dire comportant une ligne et $N$ colonnes).

  On produit une matrice en écrivent ses éléments entre crochets, chaque ligne étant délimitée par un point virgule.
  $>>$ $A = [1\,2\,3;\ 4\,5\,6;\ 7\,8\,9]$ $\hookleftarrow$.

La transposée s'obtient par $B = A'$ et la déterminant par $C = det(A)$.
Il est possible d'extraire un élément ou un groupe d'éléments d'une matrice, par exemple:
$>> D = A(2, 1) \hookleftarrow$.
$>> E = A([1\,2], [2,\ 3]) \hookleftarrow$.
$>> F = A(:, 2) \hookleftarrow$.
$>> G = A(, :) \hookleftarrow$.
La matrice $D$ contiendra l'élément situé à l'intersection de la $2^e$ ligne er de la $1^{ere}$ colonne de $A$.
La matrice $E$ contiendra l'intersection des ligned 1 et 2 et des colonned 2 et 3 de $A$.
La matrice $F$ de dimension $3 \times 1$ sera formée de la $2^e$ colonne de $A$ (le symbol : dans **A(2,:)** désigme toutes les solonnes).
Que contiendra la matrice : **H=A([3 1], :)** ?

- Gémération automatique de matrices et de vecteurs :

  $>> D = A_i : dA : A_f \hookleftarrow$. Crée un vecteur de valeur initiale $A_i$, de valeur finale $A_f$ et d'incrément $dA$ entre deux élements voisind du vecteur. Si $dA$ est omis, l'incrément par défaut est 1. Exemple:
  $>> A = 1 : 0.25 : 2 \hookleftarrow$, produit [1 1.25 1.5 1.75 2].
  $>> A = 10 : -1 : 5 \hookleftarrow$, produit [10 9 8 7 6 5].

  Les matrices prédéfinies par Matlab les plus utilisées sont:
  **eye(n)** matrice identité $n \times n$
  **ones(m,n)** matrice $m \times n$ dont tous les éléments valent 1
  **zeros(m,n)** matrice $m \times n$ dont tous les éléments valent 0
  **rand(m,n)** matrice $m \times n$ dont les éléments sont produits de façon aléatoire (distribution unforme sur [0,1])
  **randn(m,n)** matrice $m \times n$ dont les éléments sont produits de façon aléatoire (distribution gaussienne de variance 1)
  Si $(m = n)$, il suffit de spécifier une seule valeur de dimension. Exemple **ones(n)** est la matrice carrée de dimension $n \times n$ ne renfermant que des 1.

  **Remarque:** il est possible de rappeler toutes les commandes antérieures en utilisant les flêches du pavé numérique.

- Opération sur les vecteurs et matrices:

  Dans Matlab, les opérations mathématiques peuvent être effectués sur des matrices ou vecteurs entiers à l'aide d'une seule commande, sans boucle **for**. Par exemple:
  $>> A = [0 : 0.1 : 2 * pi] \hookleftarrow$
  $>> B = cos(A) \hookleftarrow$ crée un vecteur renfermant le cosiunus de chaque élément de $A$
  $>> C = sin(A) \hookleftarrow$ crée un vecteur renfermant le sinus de chaque élément de $A$
  $>> plot(A, B) \hookleftarrow$ trace $B$ en fonction de $A$ (cosinus sur une période de $2\pi$). $>> D = A + B \hookleftarrow$ somme de deux matrices
  $>> E = A^2 \hookleftarrow$ carré des éléments de $A$
  $>> F = A. * B \hookleftarrow$ produit élément par élément des matrices $A$ et $B$
  Une multiplication de matrices sera réalisée par la commande $A = V * C$, où $B$ et $C$ sont les matrices à multiplier.
  **Attention:** Il ne faur pas confondre les commandes $B * C$ et $B. * C$. La première est le produit matriciel des deux matrices $B$ et $C$ dont les dimensions doivent être compatibles (par exemple $m \times n$ et $n \times p$ respectivement).
  La seconde est le produiy élément par élément des deux matrices $A$ et $B$ dont les dimensions doivent être identiques. Exemples d'illustration:
  $>> A = [1\,2\,3] \hookleftarrow >> B = [2\,2\,2] \hookleftarrow >> C = A. * B \hookleftarrow$ qui donne [2 4 6]
  $>> D = A * B \hookleftarrow$ est impossible. Par contre, $D = A * B'$ est possible.

- Nombre Complexes

  Matb=lab offre la possibilité de travailler avec des nombres complexes. $j$ représente le nombre complexe dont le carré vaut -1.
  $>> z = 1 + j \hookleftarrow$ $z$ a pour partie réelle 1 et pour partie imaginaire 1.
  $>> real(z) \hookleftarrow$, $>> imag(z) \hookleftarrow$ retournent respectivement la partie réelle et imaginaire de $z$.
  $>> conj(z) \hookleftarrow$ donne le complexe conjugué de $z$.
  $>> a = abs(z) \hookleftarrow$ donne le module de $z$ soit $\sqrt{2}$.

$>> b = angle(z) \hookleftarrow$ donne l'argument de $z$ soit $\pi/4$.
$>> abs(z) * exp(j * angle(z)) \hookleftarrow$ redonne le nombre complexe $z$ (transformation de la forme exponentielle à la forme algèbrique).

– Les fenêtres graphiques:

Grâce à ses fonctions graphiques, Matlab permet la représentation de courbes $2D$ et $3D$. La commande **figure(num)**, vous permet de créer une nouvelle fenêtre graphique numérotée **num**.La commande **clf** permet l'effacement du contenu de cette fenêtre. La commande **close(num)** permet quant à elle de fermer la fenêtre numéro **num**. Enfin, pour diviser la fenêtre graphique en plusieurs sous fenêtres, on utilise la commande **subplot(i,j,n)**, où $i, j, n$ représentent respectivement: Le nombre de graphique(s) selon l'horizontale, le nombre graphique(s) selon la verticale et enfin le numéro du graphique que l'on désire construire.
Pour mettre les inititulés en abscisse, en ordonnée et en titre de graphique on utilise les commandes suivantes:
$>>$ *title*('ceci est la titre de la figure') $\hookleftarrow$ pour afficher le titre.
$>>$ *xlabel*('ceci est le titre de l'axe abscisses') $\hookleftarrow$
$>>$ *ylabel*('ceci est le titre de l'axe des ordonnée') $\hookleftarrow$
Il est aussi possible d'afficher plusiers courbes sur un même graphique. Pour cela, il y a deux possibilités:

– $>> plot(x_1, y_1,' k', x_2, y_2,' b') \hookleftarrow$ permet d'afficher les deux courbes $y_1(x_1)$ en noir et $y_2(x_2)$ en bleu.
– $>> plot(x_1, y_1) \hookleftarrow$ permet d'afficher la courbe $y_1(x_1)$ ensuite
  $>>$ *hold on* $\hookleftarrow$ conserve les tracés existants sur la fenêtre active
  $>> plot(x_e, y_2) \hookleftarrow$ affice la courbe $y_2(x_2)$.

Pour avoir plus de détails sur les options de *plot* (par exemple la couleu des tracés, le style etc...) faîtes un $>>$ *help plot* $\hookleftarrow$.
Finalement, on peut aussi mettre une légende au graphique, pour cela taper:
$>> legend('graphe1',' graphe2') \hookleftarrow$ pour faire apparaîte une fenêtre de légende dans la fenêtre graphique.
Noter que les noms donnés doivent correspondre à l'ordre d'affichage des courbes.

**En guise d'exercice:**

Construire les fonctions $y_1 = cos(x)$ and $y_2 = sin(x)$ en définisant un vecteur $x$ allant de 0 à $2pi$ par incrément de 0.1.

```
x = [0 : 0.1 : 2*pi];
y1 = cos(x);
y2 = sin(x);
```

Tracer $y_1$ en fonction de $x$ dans une fenêtre graphique numérotée 1.

```
plot(x, y1)
```

Tracer $y_2$ en fonction de $x$ dans une fenêtre graphique numérotée 2.

```
plot(x,y2)
```

Intituler l'axe des ordonnées **cosinus** et l'axe des abscisses **x** pour la figure 1. Mettre un titre de votre choix.

---

**Exercise 1**

Reprendre l'exercice en traçant les trois périodes des $y_1$ et $y_2$ dans la même fenêtre graphique et insérer une légende, penser à différencier les deux tracés.
Intituler les axes et mettre un titre.
Tracer les graphes et **appeler l'enseignante pour verifier.**

---

La commande $>>$ *grid* $\hookleftarrow$ permet d'afficher une grille de référence dans la fenêtre active.
La commande $>> ginput(2) \hookleftarrow$ permet f'afficher les coordonnées de deux points visés dans la fenêtre

graphique active (les coordonnéed apparaissent dans la fenêtre de commande).

Pour obtenir des échelles semilog en abscisses, on utilisera la commande **semilogx(X,Y1)**, ou **semilogy(X,Y1)** (semilog en ordonnée) ou encore **loglog(X,Y1)**.

### 1.1.2 Création d'un fichier .m

Les fichiers **nomfich.m**, sont des fichiers permettant de conserver en mémoire un ensemble de commandes (c'est un programme Matlab).

Pour créer un tel fichier, il suffit de cliquer sur lq petite feuille blanche se trouvant au coin supérieur gauche de l'écran, ou de cliquer sur **File**, puis **New** et enfin sur **M File**. Après cette opération, apparaît à l'écran uun éditeur de texte **Matlab Editor/Debugger**. C'est dans cet éditeur que l'on écrit les commandes que l'on d'ésire que Matlab exécute. Il suffit, une fois le programme écrit, de sauvegarder le fichier sous un nom avec l'extension **.m**.

Tapez dans cette fenêtre le texte

```
% programme d'essai
clear all;
a=2;
b=3;
c=5;
```

Sauver le texte que vous venez taper sous le nom *essai.m*. Pour éxécuter cette suite d'instructions, taper dans la fenêtre de commande de Matlab:
$>>$ *essai* $\hookleftarrow$
$>>$ *whos* $\hookleftarrow$ (pour voir les variables).

#### Création de fonction

Il est également possible de créer vos propres fonctions Matlab aux quelles on doit transmettre un ou plisieurs paramètres.

Exemple: on crée un fichier nommé **sa.m** comprenant les lignes suivantes:

```
function Y=sa(X)
if X==0 Y=1;
else
Y=sin(X)./X
end
```

A chaque fois que l'on demandera **sa(V)**, in obtiendra $sin(V)/V$. Avant l'éxécution de la fonction, la valeur du paramètre transmis est placée dans $X$; le résultat retourné est la valeur de $Y$ après que toutes les commandes aient été effectuée.
En guise d'exercice, tracer $Y(X) = sin(X)/X$ pour $X$ variant de $-5\pi$ à $5\pi$ par incrément de 0.1.

### 1.1.3 Sauvegarde de données .mat

On peut visualiser toutes les données stockées en mémoire en cliquant sur l'onglet workspace sur la denêtre de gauche. On a accés à la taille des données ainsi que la classes des données. Il est possible de sauvegarder toutes ces données afin de les ré-utiliser ultériurement. Pour cela, il suffit de taper la commande:
$>>$ *save* $nomfichier.mat$ $\hookleftarrow$

La taile du fichier de sauvegardepeut-être trés importante dans certains cas, il ne faut donc sauvegarder que les données utiles. On peut supprimer les données redondantes en faisant soit:

– une sélection de ces données et en faisant un click droit et supprimer.
– soit en tapant sur la fenêtre de commande directement la commande:
  *clear nomdonnée* $\hookleftarrow$

Pour charge ces données à nouveau, on tape la commande: *load* $nomfichier.mat$ $\hookleftarrow$

## 1.2 Théorème de Shannon

**Exercice 2**

On veut synthétiser le signal $s(t) = cos(2\pi ft)$ avec $f = 10$ Hz sur une durée d'observation de 2 périodes.

On définit pour cela une variable $dt$ qui représentera le pas (ou période) d'échantillonnage. Ecrire un programme Matlab permettant d'obtenir le signal $s(t)$ pout $t$ allant de 0 à 2 périodes par pas de $dt$. Pour tracer le signal $s$ en mode point par point, on utilisera la commande $plot(t, s,'.')$.

Exécuter votre programme pour $dt = 0.001$, $dt = 0.01$, et $dt = 0.1$. Tracer les graphes. Relever pour chaque valeur de $dt$ le nombre de points par période et conclure.

## 1.3 Etude de filtres analogiques

### 1.3.1 Etude d'un filtre passe-bas du premier ordre

**Exercice 3**

Soit la fonction de transfert du filtre passe bas RC de la figure ci-dessous qui peut se mettre sous la forme: $A = \frac{s}{e} = \frac{1}{1+j\frac{f}{f_0}}$ avec $f_0 = \frac{1}{2\pi RC}$.

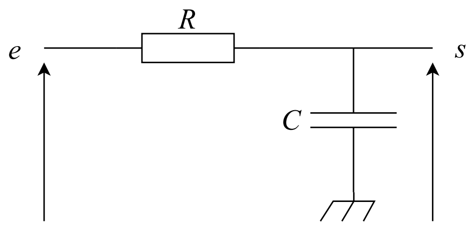



Figure 1.1: Filtre passe-bas

Créer un fichier **Pbas.m** permettant de:

1. Synthétiser la fonction de transfert $A$ avec $f_0 = 10$ de 0 Hz à 1000 Hz.
2. Calculer le Gain en dB défini par $G = 20log_{10} \parallel A \parallel$ (la fonction $log_{10}$ est **log10** sous Matlab).
3. Calculer la phase.

Tracer les courbes et déterminer les suivantes:

1. La pente en dB/decade dans la bande atténuée.
2. Le gain at la phase à la fréquence de coupure.

### 1.3.2 Etude d'un filtre passe-haut du premier ordre

**Exercice 4**

La fonction de transfert du filtre passe haut **CR** de la Figure 1.2 peut se mettre sous la forme $A = \frac{s}{e} = \frac{1}{1-j\frac{f_0}{f}}$ avec $f_0 = \frac{1}{2\pi RC}$. Reprendre l'étude avec cette nouvelle fonction de transfert.

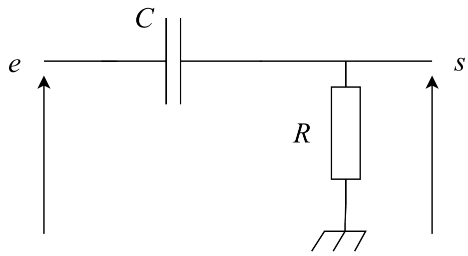



Figure 1.2: Filtre passe-haut

### 1.3.3 Etude du filtre passe-bande de Wien

**Exercice 5**

A l'aide de Matlab, tracer la fonction de transfert du filtre passe-bande: $A = \frac{s}{e} = \frac{A_0}{1+jQ\left(\frac{f}{f_0}-\frac{f_0}{f}\right)}$ avec $A_0 = Q = \frac{1}{3}$ et $f_0 = \frac{1}{2\pi RC} = 10$.

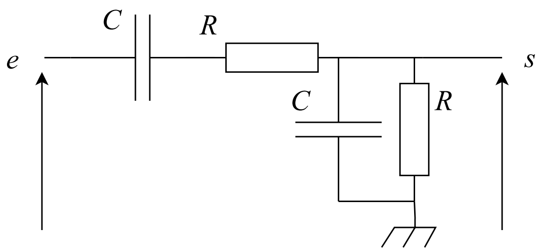
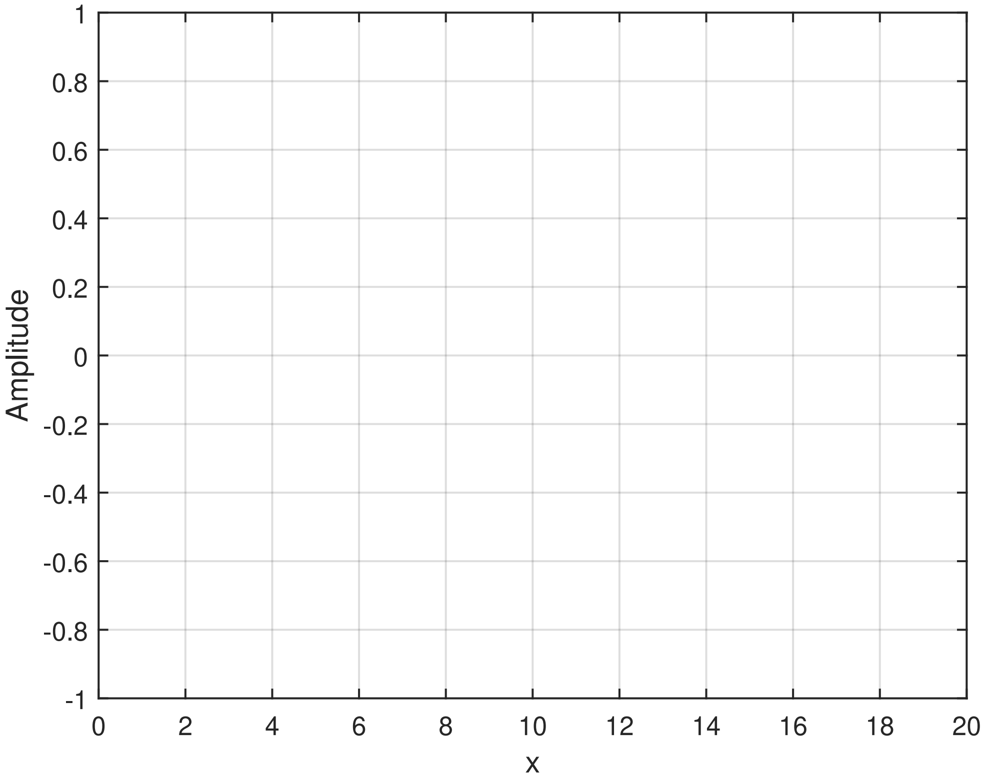



Figure 1.3: Filtre passe-bande de Wien

NOM: ______________________________ DATE: ______________
______________________________

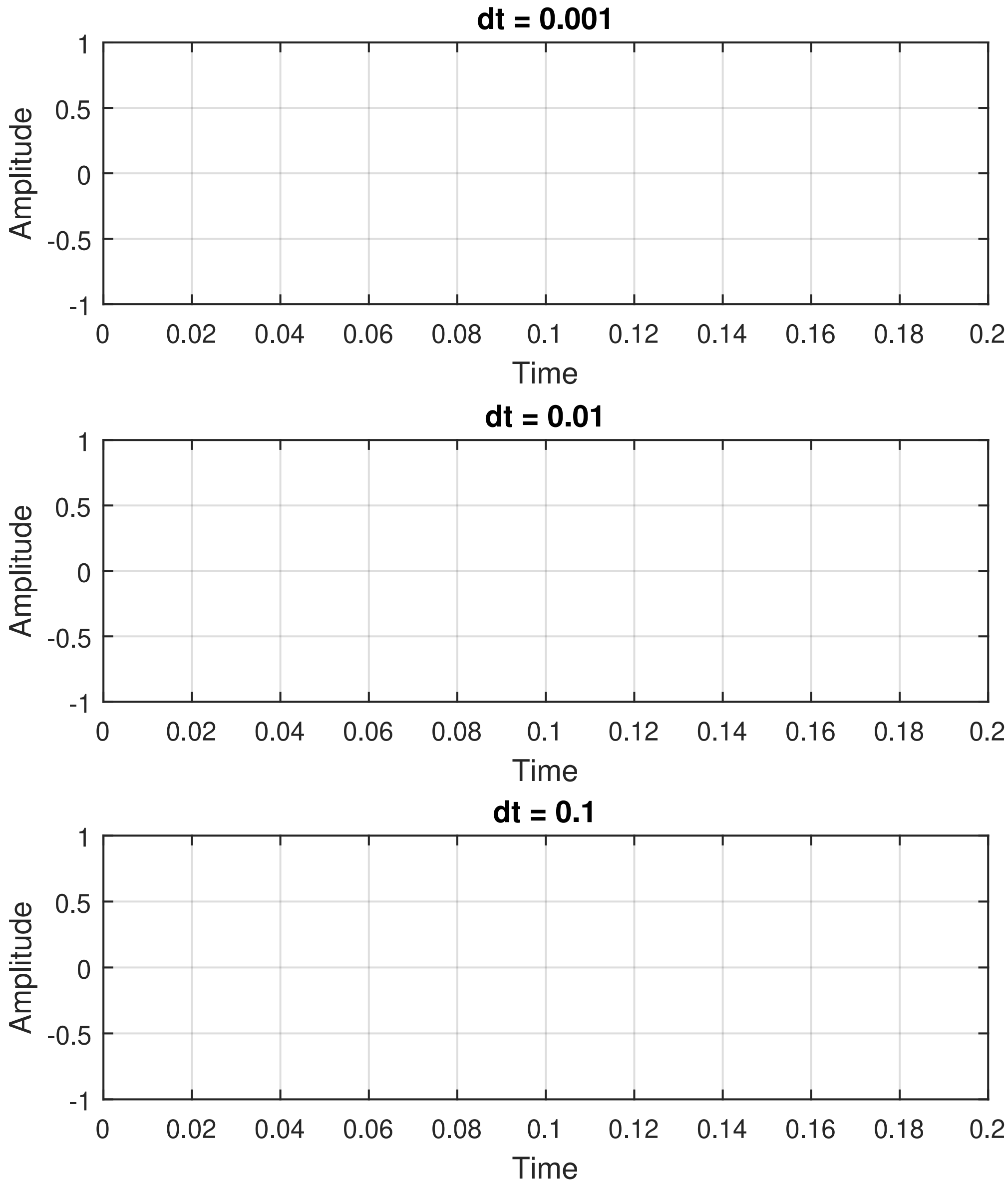
# Traitement du Signal
## Travaux Pratiques 1

**Exercice 1**




Trois périodes des signaux

**Exercice 2**

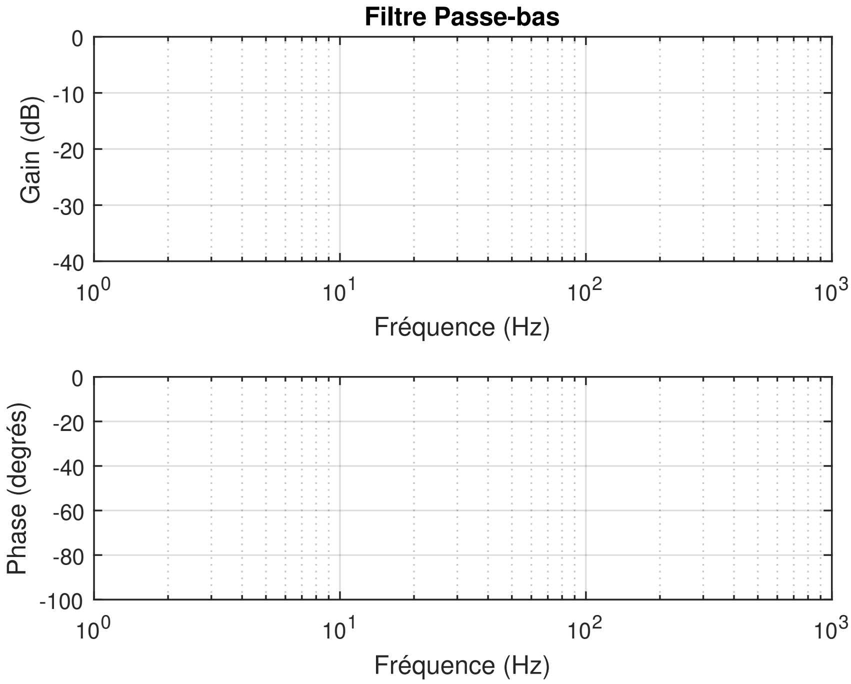



Théorème de Shannon

Nombre de points par période:
$dt = 0.001$:______________
$dt = 0.01$ :______________
$dt = 0.1$ :______________

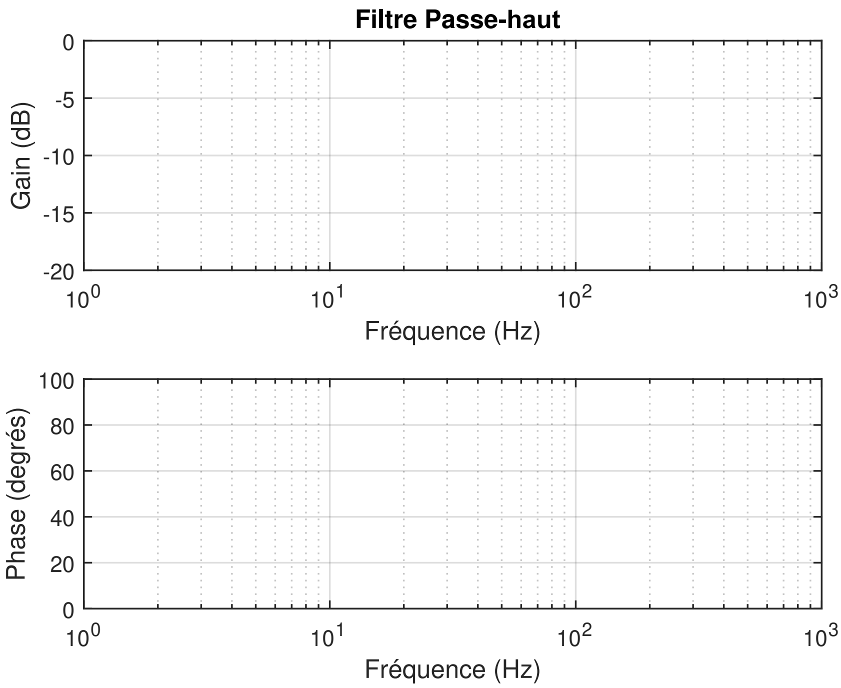
Conclusion:

**Exercice 3**



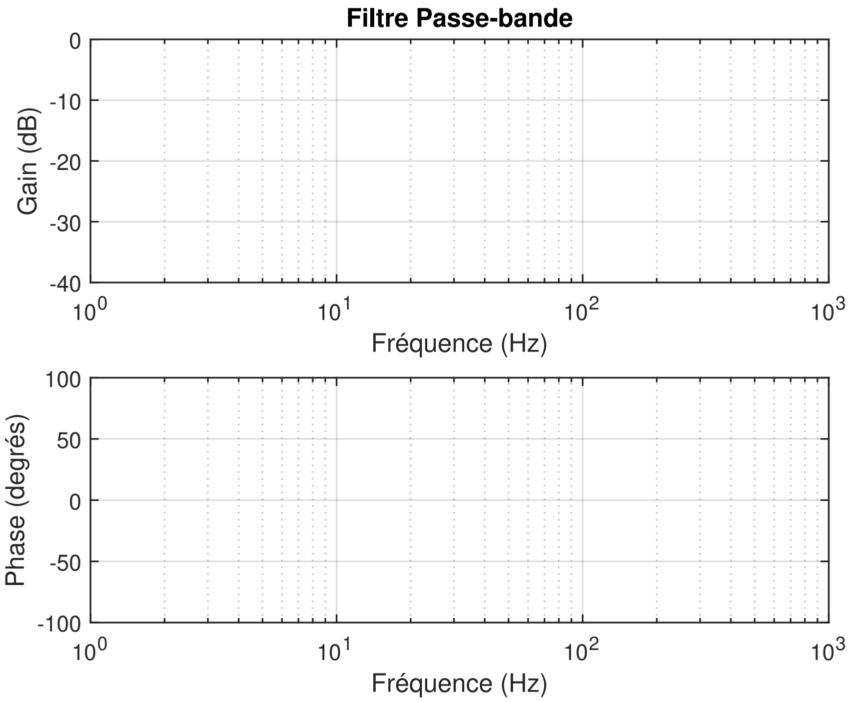

La pente: ______________ Le gain à $f_0$: ______________ La phase à $f_0$: ______________

**Exercice 4**




La pente: ______________ Le gain à $f_0$: ______________ La phase à $f_0$: ______________

**Exercice 5**




Bande-passante : ________________
Fréquence de coupure basse $f_{CB}$: ________________
Fréquence de coupure haute $f_{CH}$: ________________
Gain à $f_{CB}$: ________________
Gain à $f_{CH}$: ________________
Phase à $f_{CB}$: ________________
Phase à $f_{CH}$: ________________
Pente dans la bande atténuée: ________________