**Université Cadi Ayyad**
**Faculté des Sciences Semlalia Marrakech**
**Département de Physique**

**Année universitaire 2019-2020**
**Enseignant : Fouad Sefyani**

# Master PHEAPC

## Traitement du signal

## Polycopié de Travaux Pratiques

# AVANT PROPOS

Les Travaux Pratiques de traitement du signal présentés dans ce manuel constituent une illustration de ce que les méthodes de traitement des signaux numériques sont capables de réaliser. Elles permettent à l'étudiant de mieux comprendre ses implications pratiques. L'outil de base de ces travaux pratiques est le traitement numérique mis en œuvre par le puissant logiciel MATLAB.

Les séances de travaux pratiques sont au nombre de cinq. Il s'agit des manipulations suivantes : **[1]** Outils de base des filtres numériques **[2]** Conception des filtres numériques. **[3]** Reconnaissance des numéros DTMF d'un téléphone fixe**. [4**] Autocorrélation et intercorrélation. **[5]** Vitesse radiale des étoiles par effet Doppler.

L'utilisation des signaux physiques donne une dimension pratique quant à l'utilisation des outils et méthodes de traitement des signaux numériques. La maîtrise du savoir-faire en matière de traitement du signal numérique est l'objectif escompté à travers les différentes manipulations présentées dans ce manuel de Travaux Pratiques.

## I - INTRODUCTION

Ce document constitue un support d'initiation au logiciel MATLAB. Il permet à l'étudiant de se familiariser tout d'abord avec les outils de base dont dispose MATLAB, puis de prendre connaissance des possibilités offertes dans le domaine de traitement du signal numérique facilitant ainsi leur compréhension et leur exploitation.

**MATLAB** (**MAT**rix **LAB**oratory) est un logiciel de calcul matriciel mis au point par l'université d'Albuquerque (Nouveau-Mexique, Etats-Unis) et développé par la société MathWorks. MATLAB est écrit en langage C et conçu à la base pour un être un environnement informatique de calcul scientifique et de visualisation de données. Il peut être considéré comme un langage de programmation au même titre que le langage C, Pascal ou Basic. C'est un langage interprété, c'est-à-dire qu'il n'est pas nécessaire de compiler un programme. Les instructions, écrites dans la fenêtre de commande, sont exécutées immédiatement après avoir été tapées (après la frappe de **return**).

Dans le cadre universitaire, MATLAB constitue un outil standard pour les cours d'algèbre linéaire ainsi que pour des cours plus avancés dans d'autres domaines. Dans l'industrie il est utilisé pour la recherche ainsi pour la résolution des problèmes pratiques de l'ingénieur et des problèmes mathématiques.

MATLAB en tant que logiciel de conception des systèmes de traitement du signal est largement répandu, aujourd'hui, autant dans les universités que dans les centres de recherche et développement, ainsi que dans l'industrie.

MATLAB est un outil puissant qui permet la résolution de nombreux problèmes en beaucoup moins de temps qu'il ne faudrait pour les formuler en C ou en Pascal.

MATLAB est un langage parfaitement adapté au traitement du signal. C'est aussi un excellent outil de traitement numérique qui présente de nombreux avantages pour l'étude et même la conception des systèmes de traitement numériques des signaux. Parmi ces avantages on peut citer :

- La syntaxe MATLAB est très proche du formalisme de l'algèbre linéaire, ce qui permet de transcrire rapidement en code des opérations comme le produit scalaire de deux vecteurs (corrélation) ou encore le produit d'une matrice et d'un vecteur (transformée).
- Toutes les variables sont par définition des matrices de complexes, ce qui signifie qu'on n'a pas à déclarer une variable avant de l'utiliser (exemple la ligne a = 1 crée la variable a puis lui affecte la valeur 1).
- MATLAB comprends de nombreuses librairies, notamment pour le traitement numérique des signaux, ce qui permet de réaliser rapidement des opérations comme le filtrage numérique et les transformées discrètes et même de concevoir des filtres numériques (Butterworth, Chebyshev,…).
- MATLAB comprend également une bonne librairie graphique, ce qui permet d'observer facilement les caractéristiques des signaux, leurs analyses spectrales et le comportement des systèmes numériques (réponses impulsionnelle, réponse en fréquence, fonction de transfert,…).
- MATLAB compte plus de **500** fonctions prédéfinis et près de **80 000** fonctions ! dans ses Toolbox. Leurs natures étant très diversifiées. Même les utilisateurs les plus chevronnés ne prétendent pas connaître parfaitement la totalité des fonctions de MATLAB.

## II - Environnement de travail, scripts et fonctions

### 1 – Répertoire de travail :

- Il est souhaitable de travailler dans votre répertoire et non dans le répertoire de MATLAB. De même vous pouvez organiser vos fichiers dans des sous répertoires.
- Pour que vos fonctions et scripts soient accessibles à partir de la ligne de commande MATLAB il faut, au début de chaque session MATLAB, déclarer vos répertoires de travail. Ceci est réalisé à l'aide de la commande **path**. Cette commande permet de déclarer le chemin à suivre par MATLAB pour exécuter vos fichiers.
- Il faut aussi se placer dans votre répertoire de travail. Pour cela il est possible de se déplacer dans l'arborescence du disque de travail à l'aide des commandes **cd** et **dir** (même syntaxe que MS-DOS).
- Par défaut, les scripts et fonctions sont enregistrés dans le répertoire work sous MATLAB.

### 2 – Sauvegarde et chargement de variables

- Une variable est déclaré en scalaire (a = 5), complexe (a = 5 + 2j) ou format texte (a = 'bonjour').

- Lorsque le nombre de variables déclarés dans une cession MATLAB est élevé, il se peut qu'il soit utile de les sauvegarder dans un fichier d'extension .mat pour une réutilisation ultérieure. Ceci est rendu possible à l'aide de la commande **save** dont la syntaxe est :

  **save** nom du fichier noms des variables

*Exemple :*

**save** toto.mat A, B, C

- Si le nom des variables est omis, tout l'espace de travail est sauvé.
- Si l'extension du fichier est omise elle sera automatiquement .mat
- Si de plus, le nom du fichier est omis, la sauvegarde se fera dans le fichier matlab.mat.
- Pour recharger les variables sauvées dans un fichier .mat, il suffit de taper : **load** nom du fichier.
- Si le nom du fichier est omis, MATLAB chargera le fichier matlab.mat
- Pour effacer plusieurs variables dans l'espace de travail on utilise la commande **clear**.

*Exemple :*

**clear** x → supprime la variable x

**3 – Scripts**

- Il est parfois (souvent) souhaitable, pour ne pas avoir à taper plusieurs fois une même séquence d'instructions, de la stocker dans un fichier. Ainsi on pourra réutiliser cette séquence dans une autre session de travail. Un tel fichier est dénommé script ou fichier de commande.
- Sous windows, Pour créer un nouveau fichier de commande, choisir **New M-file** dans le menu **File**. Une fenêtre **Untitled** apparaît alors, dans laquelle la suite de commande désirées peut être éditée comme dans un traitement de texte et de sauver le fichier avec une extension .m
- Pour ouvrir un fichier script, toto.m par exemple, dans la fenêtre de commande MATLAB il suffit de taper **open** toto.m
- Pour enregistrer le fichier, choisir **Save as…** dans le menu File, introduire le nom désiré, par exemple toto, sélectionner le dossier de destination, puis enregistrer. Le nom du fichier doit comporter l'extension .m
- Par défaut, les fichiers .m sont stockés dans le dossier work.
- En tapant le nom du fichier sous MATLAB, la suite d'instructions s'exécute.
- Si la fenêtre du script est active, par exemple lors de l'édition du fichier, il suffit de choisir la commande **save** dans le menu File pour enregistrer les modifications.
- Les variables définies dans l'espace de travail sont accessibles pour le script. De même les variables définies dans le script sont accessibles dans l'espace de travail. Les fichiers de commandes partagent les variables définis dans la fenêtre de commande. Il s'agit donc de variables globales.
- Dans le fichier script, il est possible, et même recommandé, d'introduire des commentaires. Ceci est réalisé à l'aide du caractère %. Toute commande située après % n'est pas prise en compte par MATLAB jusqu'à la ligne suivante.
- MATLAB ne *voit* pas les espaces blancs entre les différents caractères des instructions de commandes. Par conséquent on peut *aérer* suffisamment le fichier pour qu'il soit facilement lisible.

**4 – Fonctions**

- MATLAB dispose de bibliothèques supplémentaires de fonctions appelées *toolboxes*, ou boîte à outils, qui contiennent des fonctions spécialisées permettant d'utiliser l'environnement MATLAB pour résoudre des problèmes bien spécifiques.
- MATLAB compte plus de 500 fonctions prédéfinies, mais il se peut qu'on ait besoin d'une fonction ne figurant pas dans son catalogue. Alors il est possible de créer de telles fonctions dans un fichier séparé et les appeler de la même façon que les fonctions préexistantes.
- Si la première ligne est une ligne de commentaire alors elle sera affichée à chaque fois que le **help** de cette fonction est sollicité.
- La première ligne (hormis la ligne de commentaires) d'une fonction doit impérativement avoir la syntaxe suivante : **function** [S1, S2,…] = **nom fonction** (E1, E2,…).

  **nom fonction** est une chaîne de caractère qui correspond au nom de la fonction. Ce nom doit être différent de celui des autres fonctions déjà disponibles. Le fichier de commandes doit porter le même nom que la fonction. E1, E2,… sont les variables transmises à la fonction lorsqu'elle est invoquée. S1,

S2,… sont les variables retournées par la fonction après son exécution. Ces dernières variables définies à l'intérieur de la fonction sont locales.

- Par défaut, les fichiers .m contenant les fonctions sont stockés dans le dossier work.

*Exemple :*

```
% la fonction calcul la somme et le produit de deux nombres.
function [a, b] = somprod (x,y)
% Cette fonction retourne la somme et le produit de deux nombres.
a = x + y ;
b = x * y ;

>> [u, v] = somprod (3,4)
>> u =7
   v =12
```

- Le nom de la fonction doit impérativement être le même que le nom du fichier dans lequel elle est stockée, sinon MATLAB ne prendra pas en compte ce nom mais uniquement celui du fichier.
- Les variables de l'espace de travail ne sont pas accessibles à la fonction sauf si elles sont entrées comme variable d'entrée. De même les variables définies dans une fonction ne sont pas accessibles dans l'espace de travail.
- Il est vivement recommandé, avant d'écrire une fonction, de consulter la documentation ou le **help** afin de savoir s'il n'existe pas déjà une fonction similaire et peut être mieux écrite.
- Avant d'utiliser une fonction MATLAB pour la première fois il est souhaitable de consulter la documentation ou le **help** pour vérifier qu'elle réalise bien ce que vous désirez.

**5 – Help**

- MATLAB est pourvu d'une fonction d'aide très utile : **help**. Ainsi si vous tapez **help** sur la ligne de commande apparaissent tous les répertoires accessibles depuis MATLAB ainsi qu'une explication (en anglais) concernant ces répertoires.
- Si vous tapez **help** suivi du nom du répertoire, MATLAB affiche une explication brève sur toutes les fonctions et scripts de ce répertoire.
- Enfin si vous tapez **help** suivi du fichier script ou fonction apparaît une explication détaillée sur l'utilisation de la fonction ou du script.

## III – OUTILS DE BASE

### 1 - Commande générales :

| | | |
|---|---|---|
| **who** | : | Affiche les variables actuellement présentes en mémoire. |
| **whos** | : | Affiche les variables actuellement présentes en mémoire ainsi qu'une série d'informations comme leur nature, leur taille en nombre de lignes et de colonnes pour les matrices, leur taille en bytes, etc... |
| **exist** ('Nom variable') | : | Affiche la valeur 1 si la variable Nom Variable existe, et la valeur 0 si la variable n'existe pas. |
| **clear** Nom variable | : | Efface de la mémoire la variable Nom Variable. |
| **clear** | : | Efface toutes les variables en mémoire. |
| **help** Nom Fonction | : | Affiche la description et la syntaxe de la fonction Nom Fonction |
| **save** Nom Fichier | : | Enregistre dans le fichier Nom Fichier l'ensemble des variables définies et actuellement présentes en mémoire. |
| **Load** Nom Fichier | : | Charge les variables stockées dans le fichier Nom Fichier. |
| X = **input** ('message') | : | Cette commande permet de saisir une donnée, elle produit l'apparition à l'écran du 'message' et attend l'introduction des éléments de X. Essayer avec |

ou sans point virgule.

**fprintf** (‘bonjour’) : Produit l’apparition à l’écran du message : bonjour.

**pause** : provoque un arrête momentané sur l’écran. presser sur n’importe quelle touche pour reprendre la main sur le clavier.

**2 - Caractères spéciaux**

; : Utilisé à la suite d’une commande, ce caractère supprime l’écho à l’écran de cette commande et de son résultat.

... : Ajouté au cours d’une commande, ce caractère permet la continuation de la commande à la ligne suivante.

**3 - Valeur particulières**

ans : Variable créée automatiquement quand le résultat d’une expression n’est pas affecté à une variable.

pi : Constante correspondant au nombre $\pi$.

i, j : Constantes correspondant à $\sqrt{-1}$.

**4 - Définitions de scalaires**

z = 1 +2*i : Définit le nombre z = 1 + 2i.

a = 2 : Définit la variable a = 2.

a : En actionnant la touche RETURN, on visualise la valeur de a.

**5 - Définition de vecteurs**

x = [-1 0 2] ou x = [-1, 0, 2] : Définit le vecteur ligne : x = [-1 0 2].

x = [-1; 0; 2] : Définit le vecteur colonne : $x = \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix}$

x = [-1.3 sqrt(3) 6*4/5] : Définit le vecteur : x = [-1.3 1 .7321 4.8]

x (5) = x(1) : Etend la dimension de x jusqu'à 5 et assigne la valeur prescrite à la dernière composante. L’élément x (4) n’étant pas défini, il vaut 0 par défaut : x = [-1.3 1.7321 4.8 0 -1.3].

y = [$y_0$ : D : $y_{Max}$] : Définit un vecteur ligne dont la première composante est $y_0$ et la dernière $y_{Max}$. Les valeurs intermédiaires sont obtenues par adjonction successive de l’incrément D.

y = [$y_0$ : $y_{Max}$] : Si l’incrément D est omis, il vaut par défaut 1.
Donc y = [$y_0$, $y_0$+1, ..., $y_{Max}$-1, $y_{Max}$].

y = [1 : 5] : Définit le vecteur : y = [1 2 3 4 5].

y = **linspace** ( $y_0$, $y_{Max}$, n): Définit un vecteur ligne dont les n composants sont linéairement espacés entre les valeurs $y_0$ et $y_{Max}$.

y = **linspace** (-pi, pi, 4) : Définit le vecteur y = [-3.1416 -1.0472 1.0472 3.1416]

w = **logspace** (d1, d2, n) : Définit un vecteur ligne dont les n composants sont logarithmiquement espacées entre les valeurs $10^{dl}$ et $10^{d2}$. C’est une fonction adaptée à la création de vecteurs de pulsations utilisées pour visualiser des réponses fréquentielles.

w = **logspace**(-2, 1, 50) : Génère 50 points compris entre 0.01 et 10.

**6 - Définitions de matrices**

A = [1 2 3 ; 4 5 6; 7 8 9] : Définit la matrice : $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

r = [10, 11, 12]

A = [A; r] : Modifie A par adjonction de r comme nouvelle ligne $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$

**ones** (n, m) : Définit une matrice de dimension n x m avec tous les éléments égaux à 1

O = **ones** (2, 3) : Définit la matrice : $O = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

**zeros** (n, m) : Définit une matrice de dimension n x m avec tous les éléments égaux à 0

**size** (A) : Définit un vecteur dont la première composante est le nombre de lignes de A et la seconde son nombre de colonnes.

**eye** (n, m) : Définit une matrice identité de dimension n x m.

E = **eye** (size (A)) : Définit une matrice E de même dimension que A, qui contient des 1 sur la diagonale principale et des 0 partout ailleurs.

$$E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

**diag**(v) : Si v est un vecteur à n composantes, alors **diag**(v) est une matrice carrée de dimension n qui porte sur la diagonale principale les éléments du vecteur v.

D = **diag** ([1 2 3]) : Définit la matrice diagonale : $D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$

V = [ ] : Définit une matrice V de dimension 0 x 0.

**7 - Manipulations de matrices ou de vecteurs**

A (:, J) : Représente la j-ième colonne de A.

A (i, :) : Représente la i-ième ligne de A.

A (:, k : m) : Représente [A(:, k), A(:, k+1), … , A(:, m)]

A (:, 2)= [ ] : Efface la 2ème colonne de la matrice A. On obtient : $A = \begin{bmatrix} 1 & 3 \\ 4 & 6 \\ 7 & 9 \\ 10 & 12 \end{bmatrix}$

A (i, j) : En actionnant la touche RETURN on visualise la valeur de l'élément (i, j) de la matrice A.

**EXEMPLE :**

Soit A = [1 2 3; 4 5 6; 7 8 9; 10 11 12];

A = A (1 : 3, :) : Produit la matrice $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

A = A (3, 2) : Visualise la valeur 8

**8 - Opérations matricielles ou vectorielles**

Soient x = [-1 0 2] , A = [ 1 2 3; 4 5 6; 7 8 0] et Z = [1+5*i, 2+6*i; 3 + 7* i, 4+8*i];

- **Complexe conjugué transposé**

B = A' : Correspond à B = $A^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 0 \end{bmatrix}$

x = x' : Produit le vecteur $x = \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix}$

W = Z' : Produit la matrice $W = \begin{bmatrix} 1-5i & 3-7i \\ 2-6i & 4-8i \end{bmatrix}$

- **Transposé**

W = Z'. : Produit la matrice $W = \begin{bmatrix} 1+5i & 3+7i \\ 2+6i & 4+8i \end{bmatrix}$

- **Addition et soustraction**

Les opérations sont définies uniquement si les variables ont la même dimension ou si une des variables est un scalaire.

C = A+B : Produit la matrice $C = \begin{bmatrix} 2 & 6 & 10 \\ 6 & 10 & 14 \\ 10 & 14 & 0 \end{bmatrix}$

y = x –1 : Produit le vecteur $y = \begin{bmatrix} -2 \\ -1 \\ 1 \end{bmatrix}$

- **Multiplication et division**

Les dimensions des variables doivent être compatibles.

x'*y : Produit le scalaire ans = 4

x*y' : Produit la matrice ans = $\begin{bmatrix} 2 & 1 & -1 \\ 0 & 0 & 0 \\ -4 & -2 & 2 \end{bmatrix}$

b = A*x : Produit le vecteur $b = \begin{bmatrix} 5 \\ 8 \\ -7 \end{bmatrix}$

pi*x : Produit le vecteur ans = $\begin{bmatrix} -3.1416 \\ 0 \\ 6.2832 \end{bmatrix}$

C = A.*B : Correspond à la multiplication, élément par élément, des matrice A et B, $c_{ij} = a_{ij}\, b_{ij}$.

$$C = \begin{bmatrix} 1 & 8 & 21 \\ 8 & 25 & 48 \\ 21 & 48 & 0 \end{bmatrix}$$

B/A : Correspond formellement à $BA^{-1}$, si A est une matrice carrée inversible.

z = x. /y : Correspond à la division, élément par élément, des vecteur x et y.

Donne le vecteur : $z = \begin{bmatrix} 0.5 \\ 0 \\ 2 \end{bmatrix}$

- **Puissance**

A^n : correspond à An. Cette opération est définie si A est une matrice carrée et si n est un scalaire.

Z = y.^x : Correspond à une élévation en puissance élément par élément. Donne le vecteur :

$$z = \begin{bmatrix} -0.5 \\ 1 \\ 1 \end{bmatrix}$$

z = x.^2 : Donne le vecteur $z = \begin{bmatrix} 1 \\ 0 \\ 4 \end{bmatrix}$

**9 - Fonctions**

- **Fonctions matricielles élémentaires**

Ces fonctions sont définies uniquement pour des matrices carrées. Soit A une matrice de dimension n x n ;

**det**(A) : Déterminant de la matrice A.

**inv**(A) : Inverse de la matrice A.

**expm**(A) : Exponentielle de la matrice A.

**logm**(A) : Logarithme naturel de la matrice A.

**sqrtm**(A) : Racine carrée de la matrice A. Donne $A = P^{-1}BP$. Avec P matrice de passage et B matrice diagonale. $A^{1/2} = P^{-1} B^{1/2} P. B^{1/2}$ sera calculé élément par élément.

**poly**(A) : Vecteur **ligne** de dimension n+1 dont les éléments sont les coefficients du polynôme caractéristique det(sI-A) ordonnés en puissances décroissantes. .

**eig**(A) : Vecteur **colonne** contenant les valeurs propres de la matrice A.

[V, D] = **eig**(A) : Calcule les vecteurs propres et les valeurs propres de la matrice A et les retourne respectivement dans les matrices V et D. Chaque colonne de la matrice V correspond à un vecteur propre associé à la valeur propre se trouvant dans la colonne correspondante de la matrice D. Ceci peut être écrit de la manière suivante : AV= VD

*Exemple :*

Considérons la matrice A tels que A = [2 0 4 ; 3 –4 12 ; 1 –2 5]

[V, D] = eig(A) donne $V = \begin{pmatrix} -0.8944 & -0.9701 & 0.7428 \\ -0.4472 & 0 & -0.5571 \\ 0 & 0.2425 & -0.3714 \end{pmatrix}$ et $D = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

- **Fonctions Mathématiques élémentaires**

Il existe un ensemble de fonctions mathématiques élémentaires qui s'appliquent aux éléments de vecteurs ou de matrices.

**sin** (A) : Sinus des éléments de la matrice A.

**cos** (A) : Cosinus des éléments de la matrice A.

**tan** (A) : tangente des éléments de la matrice A.

**asin** (A) : Arc sinus des éléments de la matrice A.

**acos**(A) : Arc cosinus des éléments de la matrice A.

**atan** (A) : Arc tangente des éléments de la matrice A.

**abs** (A) : Valeur absolue des éléments de la matrice A.

**sqrt** (A) : Racine carré des éléments de la matrice A.

**real** (A) : Partie réelle des éléments de la matrice A.

**imag** (A) : Partie imaginaire des éléments de la matrice A.

**conj** (A) : Complexe conjugué des éléments de la matrice A.

**exp** (A) : Exponentielle des éléments de la matrice A.

**log** (A) : Logarithme naturel des éléments de la matrice A.

**log10** (A) : Logarithme décimal à base de 10 des éléments de la matrice A.

- **Fonctions pour l'analyse des données**

Les fonctions suivantes opèrent sur les colonnes d'une matrice ou sur les éléments d'un vecteur ligne ou colonne.

Soient x = [ 6 4 1 2 5 3 ] et A = [ 9 8 4; 1 6 5; 3 2 7 ]

mx = **max** (x) : Fournit l'élément le plus grand du vecteur x, mx = 6.

mA = **max** (A) : Définit un vecteur contenant la valeur maximale de chaque colonne de la matrice A, mA = [ 9 8 7].

mvx = **mean** (x) : Calcule la valeur moyenne du vecteur x, mvx = 3.5.

mvA = **mean** (A) : Définit un vecteur contenant la valeur moyenne calculée sur chaque colonne de la matrice A, mvA = [ 4.333 5.333 5.333].

sx = **sort**(A) : Classe les éléments de chaque colonne de la matrice A selon l'ordre croissant, sx = [1 2 3 4 5 6 ]

sA = **sort**(A) : Classe les éléments de chaque colonne de la matrice A selon l'ordre croissant

$$sA = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 6 & 5 \\ 9 & 8 & 7 \end{bmatrix}$$

sux = **sum**(x) : Calcule la somme de tous les éléments du vecteur x, sux = 21.

suA = **sum**(A) : Définit le vecteur contenant la somme calculée sur chaque colonne de la matrice A, sux = [13 16 16]

- **Fonctions opérant sur des polynômes**

MATLAB dispose également de fonctions qui opèrent sur des polynômes représentés par des vecteurs de coefficients.

Soient p = [1 –6 –72 –27], a = [1 2 3] et b = [4 5 6]

r = **roots** (p) : Définit un vecteur colonne dont les composantes sont les racines du polynôme $s^3 - 6s^2 - 72s - 27$,

$$r = \begin{bmatrix} 12.1229 \\ -5.7345 \\ -0.3884 \end{bmatrix}$$

p2 = **poly** (r) : Calcule les coefficients d'un polynôme définit par ses racines, p2 = [1 –6 –72 –27].

v = **polyval** (p, x): Retourne, dans v, les valeurs calculées du polynôme p en chaque point de x.

c = **conv** (a, b) : Effectue le produit des polynômes a(s) = $s^2 + 2s + 3$ et b(s) = $4s^2 + 5s + 6$ puis fournit un vecteur contenant les coefficients du polynôme résultant, c = [4 13 28 27 18].

[q, r] = **deconv** (B, A) : Réalise la division polynomiale B/A et retourne dans q les coefficients du polynôme quotient et dans r ceux du reste. C'est-à-dire B = A q + r

*Exemple :*
[q, r] = **deconv**(c, a)

ans q = 4 5 6
r = 0 0 0 0 0

Ainsi c = q a + r $4x^4 + 13x^3 + 28x^2 + 27x + 18 = (4s^2 + 5s + 6)(s^2 + 2s + 3) + 0$

On peut vérifier ce résultat en utilisant la commande **conv** entre les polynômes q et a pour voir si le résultat donne bien le polynôme c.

- **Fonctions opérant sur les fractions rationnelles.**

Soient les polynômes A = [1 5 6] et B = [1],

[r, p, k] = **residue** (B, A) : effectue la division des polynômes B par A et la retourne sous la forme suivante : $\frac{B(x)}{A(x)} = \frac{r(1)}{x - p(1)} + \frac{r(2)}{x - p(2)} + ... + \frac{r(n)}{x - p(n)} + k(x)$

Ainsi : $\frac{1}{x^2 + 5x + 6} = \frac{-1}{x+3} + \frac{1}{x+2}$

[B, A] = **residue** (r, p, k) : Effectue l'opération inverse et retourne dans B les coefficients de B(x) et dans A ceux de A(x).

**10 - graphiques**

Avec MATLAB les données peuvent être examinées graphiquement. Pour la représentation d'un graphe à l'écran il est possible de choisir le type d'échelle désiré pour les axes x et y.

**Plot** (x, y) : Graphe linéaire de y en fonction de x. x et y doivent être deux vecteurs de même dimension.

**Plot** (x, y, 's1s2s3') : Donne la courbe y(x). s1 et s2 et s3 sont trois paramètres qui précisent la couleur, le type de point et la nature de lignes d'interpolation choisies.

| s1 (couleur) | | s2 (type de point) | | s3 (nature de lignes) | |
|---|---|---|---|---|---|
| y | jaune | . | point | - | Solide |
| m | magenta | o | cercle | : | pointillé |
| c | bleu ciel | x | marque x | -. | tiré point |
| r | rouge | + | plus | -- | 2 tiré |
| g | vert | * | étoile | | |
| b | bleu | s | carreau | | |
| w | blanc | d | diamant | | |
| k | noir | ^ | triangle (haut) | | |
| | | v | triangle (bas) | | |
| | | < | triangle (gauche) | | |
| | | > | triangle (droite) | | |
| | | p | pentagone | | |
| | | h | hexagone | | |

**Plot** (x, y, t, z) : Donne la courbe y(x) et la courbe z(t) sur le même graphique. Eventuellement on peut ajouter les données concernant la couleur, le type point et la nature d'interpolation

**stem** (n, y) : analogue à la commande **plot** sauf qu'elle trace la courbe du signal numérique y en fonction du pas d'échantillonnage n.

**subplot** (nml) : subdivise la fenêtre du graphe en n lignes et m colonnes et choisie la $l^{ème}$ pour être active. Les zones graphiques sont numérotées de la gauche vers la droite et de haut vers le bas.

Une fois le graphe est visualisé à l'écran, il est possible d'introduire un titre et des noms indicatifs pour les axes x et y.

**title** ('texte') : Ajoute la phrase écrite entre apostrophe au sommet du graphe.

**gtext** ('texte') : Place le message entre apostrophe sur le graphique en cours. Une fois la position choisie est indiquée par le témoin de la souris, il suffit d'y cliquer ou d'appuyer sur une touche du clavier.

**xlabel** ('texte') : Ajoute le texte écrit entre apostrophe sur le graphe pour la légende de l'axe des x.

**ylabel** ('texte') : Ajoute le texte écrit entre apostrophe sur le graphe pour la légende de l'axe des y.

**clf** : Efface toutes les courbes du graphe.

**grid** ou **grid on** : Place les lignes du quadrillage sur la courbe courante.

**grid off** : Supprime les quadrillages placés sur la courbe.

**axis** ([$x_{min}$ $x_{max}$ $y_{min}$ $y_{max}$]): Définit les limites des coordonnées des axes x et y en tenant compte des limites mentionnées par $x_{min}$, $x_{max}$, $y_{min}$ et $y_{max}$

*Exemple :*

```
t = 0 : 0.05 : 4*pi ;
x = cos (t) ; y = sin(t) ;
plot ( t, x, 'r.', t, y, 'g*')
xlabel ('t en radian'), ylabel ('cos(t) en rouge et sin(t) en vert')
```

# TP N° 1 OUTILS DE BASE DES FILTRES NUMERIQUES

## I - BUT

Le but de cette manipulation est l'étude des outils de base des filtres numériques. Il s'agit de mettre en évidence, au niveau temporel et fréquentiel, les caractéristiques d'un filtre, d'étudier son régime transitoire et permanent et de réaliser l'opération de filtrage de deux signaux de fréquences différentes.

## II - INTRODUCTION

Si un système numérique, linéaire et invariant dans le temps est utilisé pour modifier la distribution fréquentielle des composantes d'un signal selon des spécifications données, en s'appuyant sur des opérations arithmétiques, alors il s'appelle filtre numérique. L'opération de traitement qui consiste à modifier cette distribution à l'aide d'un système numérique s'appelle filtrage numérique.

Le principal traitement opéré par des systèmes numériques SLTI est le filtrage. Historiquement, ils ont été développés et étudiés dans le but de pouvoir simuler les filtres analogiques sur ordinateur. Ceci a permis de vérifier les performances et d'optimiser les paramètres de ces filtres avant leurs éventuelles réalisations. Ceci est rendu possible vu les progrès actuels de la technologie des circuits intégrés numériques. Ainsi, des méthodes propres pour la synthèse et la conception des filtres numériques ont été également développées.

### 1 - Présentation d'un filtre avec MATLAB

La fonction **filter** dans MATLAB permet de réaliser le filtre numérique décrit par :

$$a_0 y(k) + \sum_{n=1}^{N} a_n y(k-n) = \sum_{m=0}^{M} b_m x(k-m)$$

La syntaxe de la commande **filter** est la suivante : y = **filter** (b, a, x)

| | | |
|---|---|---|
| $b = [b_0, b_1, ...b_M]$ | : | vecteur ligne contenant les coefficients de la partie non récursive du filtre. |
| $a = [a_0, a_1, ...a_N]$ | : | vecteur ligne contenant les coefficients de la partie récursive du filtre. |
| x | : | vecteur ligne contenant le signal d'entrée. |

### 2 - Présentation de la réponse en fréquence avec MATLAB

La fonction **freqz** dans MATLAB permet de calculer la réponse en fréquence d'un filtre numérique à partir des coefficients, a et b, de l'équation récurrente qui régit son état. La syntaxe de cette fonction est :

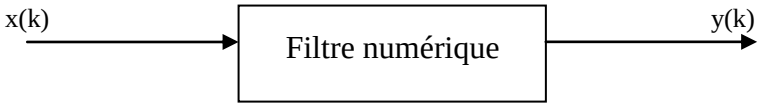
[H, f] = **freqz** (b, a, N, $F_e$)

Retourne dans H le vecteur contenant le réponse en fréquence du filtre, en fonction de f,

| | | |
|---|---|---|
| f | : | échelle des pulsations en rad/échantillons (0 et π) par défaut ou entre 0 et $F_e/2$ |
| N | : | dimension des vecteurs H et f : nombres de points calculés. ***Prenez N = 512*** |
| $F_e$ | : | fréquence d'échantillonnage. |

## III - MANIPULATION

De façon générale, un filtre numérique peut être vu comme une boîte fonctionnelle, qui transforme un signal d'entrée x(k) en un signal de sortie y(k). Les signaux étant bien sûr discrets.




Tous les filtres que nous allons étudier sont linéaires et invariants dans le temps. Un tel filtre doit obéir au principe de superposition. La relation d'entrée-sortie est régie par l'équation récurrente suivante :

$$y(k) + \sum_{n=1}^{N} a_n y(k-n) = \sum_{m=0}^{M} b_m x(k-m)$$

L'échantillon y(k) de la sortie est une combinaison linéaire des M+1 échantillons du signal d'entrée et des N derniers échantillons du signal de sortie.

***Fréquence normalisée :***

Un système de traitement numérique interagit avec des signaux numériques. La fréquence d'échantillonnage des signaux d'entrée et de sortie est prise égale à l'unité. Si ces signaux proviennent de l'échantillonnage des signaux analogique alors il faudra tenir compte de la fréquence d'échantillonnage en remplaçant k par $kT_e$. Une autre manière d'en tenir compte est de définir ce qu'on appelle la fréquence normalisée. La fréquence normalisée $f_0^n$ est donnée par la relation suivante : $f_0^n = \frac{f_0}{F_e}$ où $F_e$ est la fréquence d'échantillonnage. Pour réaliser un filtrage des signaux périodiques, leurs pulsations doivent être divisées par $F_e$ :

$$\cos(2\pi f_0 k) \quad \rightarrow \quad \cos(\frac{2\pi f_0}{F_e} k)$$

**1ère étape :**

Nous allons déterminer la réponse en fréquence d'un filtre numérique. Prenons le filtre causal suivant :

y(k) = 0.0422 x(k) + 0.0843 x(k-1) + 0.0422 x(k-2) + 1.3993 y(k-1) - 0.5779 y(k-2).

- Utiliser la commande **freqz**, sous sa forme standard, pour déterminer la réponse en fréquence du filtre en fonction de la fréquence en Hz.
- Tracer la courbe du module de la réponse en fréquence H(f).

**2ème étape :**

Ce filtre sera excité par un signal d'entrée x(k) = cos(2πf$_0$k***T$_e$**). Avec $f_0$ = 500 Hz. $F_e$ = 8000Hz.

**1)** Peut-on prévoir le signal de sortie en s'appuyant juste sur la courbe de la réponse en fréquence H(f) du filtre ?
- Définir le vecteur k allant de 0 jusqu'à 199 avec un pas unité.
- Définir le signal x(k) et déterminer le signal de sortie y(k) à travers ce filtre (Attention à la fréquence normalisée).
- Tracer sur un graphe, en utilisant la commande **plot** plutôt que **stem**, les deux signaux x(k) et y(k).

**2)** Déterminer, sur la courbe de y(k), le gain obtenu sur le signal d'entrée.

**3)** Retrouver cette valeur en utilisant la courbe du module de la réponse en fréquence. L'accord est-il satisfaisant ?

**3ème étape :**

Nous allons modifier, maintenant, la fréquence du signal d'entrée x1(k) à 2500 Hz plutôt que 500 Hz.

**4)** Avant d'effectuer le filtrage de ce signal, peut-on prévoir le signal de sortie ?
- Définir le nouveau signal d'entrée x1(k) et déterminer le signal de sortie y1(k) à travers ce filtre.
- Tracer sur un graphe, en utilisant la commande **plot** plutôt que **stem**, les deux signaux x1(k) et y1(k)..

**5)** Déterminer sur la courbe de la réponse en fréquence le facteur d'amplification du signal de sortie à la fréquence de 2500 Hz.

**6)** Que pouvez vous en conclure à propos du filtrage de ce 2ème signal d'entrée.

**Remarque : (Régime permanent et régime transitoire)**

Le signal de sortie peut être décomposé en deux partie : la 1ère partie est située sur les 15 premiers échantillons et la 2ème partie se trouve au delà de l'échantillon 15.

- Utiliser la commande $[h,k] = impz(b,a);$ pour calculer la réponse impulsionnelle.

**7)** Donner une interprétation sur le comportement de y1(k) en le comparant à la réponse impulsionnelle sur la même zone temporelle.

# TP N° 2 CONCEPTION DES FILTRES NUMERIQUES

## I - BUT

Cette manipulation a pour but d'étudier les méthodes de conception, par simulation, des filtres numériques non récursifs RIF et récursive RII.

## II - CONCEPTION DES FILTRES RIF

### 1 - Introduction

Nous avons vu, jusqu'à maintenant, que le comportement d'un filtre numérique dépend des valeurs des coefficients $a_n$ et $b_m$ contenus dans l'équation récurrente qui régit sa relation d'entrée sortie. La réponse en fréquence des filtres idéals est définie par des formes carrées. Néanmoins nous pouvons réaliser ces filtres en passant par des approximations. Suivant les pentes stipulées dans le cahier des charges, et suivant les ondulations admises dans la bande passante et dans la bande atténuée, on peut réaliser le filtre correspondant.

### 2 - Fréquence normalisée :

Les fonctions de conception de filtre utilisent la fréquence normalisée cette fois-ci sur l'intervalle [0, fe/2]. Par conséquent, la fréquence normalisée utilisée sera donnée par la relation suivante : $f_0^n = \frac{2 f_0}{F_e}$ où $F_e$ est la fréquence d'échantillonnage, elle est rapportée à l'intervalle [0 1]. Etant donnée que la fréquence est une quantité positive elle ne peut dépasser $F_e/2$.

### 3 - Filtre RIF passe bas :

Nous voulons réaliser un filtre numérique RIF à partir du gabarit ci dessous selon le cahier de charges du filtre numérique à réaliser : Commande $b = fir1(N, f_c)$;

- Filtre passe bas.
- Fréquence d'échantillonnage $f_e$ = 8000 Hz.
- Fréquence de coupure $f_c$ = 300 Hz. Atténuation de 10 dB au minimum à la fréquence 1500 Hz

**1)** Optimiser l'ordre N du filtre
**2)** Exprimer les coefficients an et bm du filtre
**3)** Calculer la nouvelle atténuation du filtre en dB

### 4 – Choix de la fenêtre de pondération

Nous allons utiliser plusieurs fenêtres de pondérations afin de bien observer l'influence de ces fenêtres sur la pente et la bande atténuée du filtre

Commande $b = fir1(N, f_c, win)$; $win = fenetre(N+1)$;

$fenêtre = boxcar, ham\min g, hanning, blackman,...$

- Choisir une à une ces fenêtres et tracer la réponse en fréquence pour l'ordre déjà établie dans la question précédente.
- Commenter et discuter la pente et les ondulations en dB de chaque filtre synthétisé.

## III – CONCEPTION DES FILTRES RII

### 1 - Introduction

Historiquement, les premiers filtres étaient évidemment analogiques. Avec l'avènement des systèmes numériques, une classe importante des filtres numériques, à réponse impulsionnelle infinie dite récursifs, RII, a été issue directement de la numérisation de ces filtres analogiques par le biais de la transformation bilinéaire.

Nous allons étudier aussi la conception des filtres numériques sans partir d'un filtre analogique connu, mais en se basant uniquement sur le gabarit de la réponse en fréquence qui exprime des données spectrales concernant la forme, le gain, l'argument et les fréquences de coupure du filtre numérique. Les filtres analogiques de Butterworth et de Chebyshev, les plus couramment utilisés, possèdent des réponses en fréquence et des fonctions de transfert entièrement définis par des expressions analytiques. La réalisation pratique de ces filtres analogiques est assurée par l'utilisation des composantes électroniques telles les multiplieurs, les additionneurs

et les retardateurs. La numérisation de ces filtres est réalisée directement sous MATLAB par des commandes bien spécifiées (**butter**, **cheby1**,…) qui calculent les coefficients $a_n$ et $b_m$ du filtre à partir du gabarit de la réponse en fréquence de filtres analogiques.

Nous allons d'abord définir l'expression analytique des réponses en fréquence des filtres analogiques. Ensuite on présentera les commandes **butter** et **cheby1** à travers leurs syntaxes pour entamer l'étude de la conception des filtres passe bas et passe bande numériques.

**2 - Le filtre de Butterworth :**

Le filtre de Butterworth d'ordre N est défini pour avoir une réponse fréquentielle plate au maximum dans la bande passante avec une atténuation de – 3dB pour f = $f_c$. Il satisfait l'équation suivante : $H(f) = \frac{1}{\sqrt{1+(\frac{f}{f_c})^{2N}}}$. La syntaxe de la commande du filtre de Butterworth **numérique** est :

[b, a] = **butter** (N, $f_c$) : N est l'ordre du filtre (il y' en a plusieurs, chaque ordre défini une forme différente de la réponse en fréquence.)
$f_c$ est la fréquence de coupure normalisée (rapporté entre 0 et 1) du filtre. Par défaut ce filtre est passe bas.

[b, a] = **butter** (N, $f_c$, 'high') : $f_c$ est dans ce cas défini par un vecteur contenant les fréquences de coupure normalisée $f_c = [f_1, f_2]$
'high' pour filtre passe haut.
'bandpass' pour filtre passe bande.
'stop' pour filtre coupe bande.

La syntaxe de la commande du filtre de Butterworth **analogique** est :

[b, a] = **butter** (N, $\omega_c$,'s') : N est l'ordre du filtre. $w_c$ est la pulsation de coupure en Hz. Par défaut ce filtre est passe bas.

**3 - Le filtre de Chebyshev :**

Le filtre de Chebyshev d'ordre N est définit pour avoir une amplitude constante ε dans la bande passante. Il satisfait l'équation suivante : $H(f) = \frac{1}{\sqrt{1+\varepsilon^2 P_N^2(f)}}$ où $P_N$ (f) est le polynôme de Chebyshev d'ordre N. On constate que plus ε est grand, meilleur est la pente dans la bande de transition et inversement. La syntaxe de la commande du filtre de Chebyshev **numérique** est :

[b, a] = **cheby1** (N, R, $f_c$) : N est l'ordre du filtre (il est l'ordre du polynôme de Chebyshev). Il y' en a plusieurs, chaque ordre défini une forme différente de la réponse en fréquence. R est le paramètre qui détermine l'amplitude des oscillations, en décibel, dans la bande passante du filtre. La valeur la plus utilisée est 0.5. $f_c$ est la fréquence de coupure normalisée du filtre. Par défaut ce filtre est passe bas.

[b, a] = **cheby1** (N, R, $f_c$, 'high') : $f_c$ est dans ce cas défini par un vecteur contenant les fréquences de coupure normalisée $f_c = [f_1, f_2]$ 'high' pour filtre passe haut. 'bandpass' pour filtre passe bande. 'stop' pour filtre coupe bande.

La syntaxe de la commande du filtre de Chebyshev **analogique** est :

[b, a] = **cheby1** (N, R, $\omega_c$,'s') : N est l'ordre du filtre. $\omega_c$ est la fréquence de coupure en Hz. rad.

***Remarque :***

La commande [H, ω] = **freqs** (b, a) Permet de calculer sur 200 points l'amplitude de la réponse fréquentielle H en fonction de la pulsation ω.

**4 - Numérisation des filtres analogiques :**

La numérisation des filtres analogiques, une fois le cahier de charge défini, est réalisée par la méthode de la transformation bilinéaire, la plus couramment utilisée. Cette méthode consiste à trouver une correspondance entre le domaine de stabilité en p et le domaine de stabilité en z. Il ressort de cette méthode la correspondante suivante : $p \Rightarrow \frac{2}{T_e}\frac{1-z^{-1}}{1+z^{-1}}$ et $z \Rightarrow \frac{1+\frac{pT_e}{2}}{1-\frac{pT_e}{2}}$

Cependant, la transformation bilinéaire entraîne une relation non linéaire entre les fréquences du domaine analogique et celles du domaine numérique (frequency warping) : $f_n = \frac{f_e}{\pi} a\tan(\pi T_e f_a)$

La méthode de numérisation d'un filtre analogique permet de déterminer, théoriquement, la fonction de transfert du filtre numérique correspondante à la fonction de transfert du filtre analogique. Cependant les commandes **butter** et **cheby1** permettent de réaliser cette correspondance directement.

**5 - MANIPULATION**

**a – Numérisation d'un filtre passe bas analogique CR :**

***Filtre analogique CR :***

- Calculer la fonction de transfert d'un circuit CR passe haut de constante de temps RC = $10^{-3}$s.
- En déduire les coefficients a et b de H(p).
- Calculer la réponse en fréquence H(f) avec la commande **freqs**.
- Tracer sur un graphe le module de H(f).
- Mesurer la fréquence de coupure de ce filtre. Vérifier qu'elle correspond bien à la valeur théorique.

***Numérisation :***

- Transformation de p vers $z^{-1}$. Déduction de H(z) théoriquement. On peut utiliser la commande : $[b,a] = bilinear(b_a, a_a, f_e)$;
- Calculer la réponse en fréquence H(f) avec la commande **freqz**.
- Tracer sur un le même graphe le module de H(f).
- Mesurer la fréquence de coupure numérique du filtre
- Vérifier que cette fréquence de coupure est bien celle donnée par la Transformation bilinéaire de la fréquence de coupure analogique : $f_n = \frac{f_e}{\pi} a\tan(\pi f_a T_e)$.
- **Conclusion.**

**b – Filtrage d'un signal composite par deux filtres numériques de Chebyshev :**

Nous allons utiliser maintenant deux filtres de Chebyshev, l'un passe bas et l'autre passe haut, pour filtrer un signal contenant deux composantes périodiques de fréquences différentes. Le but est de séparer ces deux composantes.

Cahier de charges des filtres numériques à réaliser (gabarit) :

- Filtre passe bas et passe haut.
- Ordre N = 4.
- Fréquence d'échantillonnage $f_e$ = 8000 Hz.
- Ondulation maximum de 0.5 dB dans la bande passante.
- Fréquences de coupure respectivement de $f_{c1}$ = 216 Hz et de $f_{c2}$ = 293 Hz.

Le signal composite est défini par : x = 10*cos(2π(128Hz)k*$\mathbf{T_e}$) + 10*sin(2π(382Hz)k*$\mathbf{T_e}$).

- Définir la variable k discrète de 0 à 149 avec un pas unité.
- Définir le signal x.
- Définir les deux filtres passe bas et passe haut
- Définir la réponse en fréquence des deux filtres $H_1(f)$ et $H_2(f)$ et tracer leurs modules.
- Calculer les signaux y1 et y2 de sortie des deux filtres passe bas et passe haut correspondant et les tracer.

# TP N$^{0}$ 3 RECONNAISSANCE DES NUMEROS DTMF D'UN TELEPHONE FIXE

## I - BUT

Cette manipulation a pour but de détecter les numéros d'un téléphone fixe à partir de l'enregistrement des sons des tonalités de ces numéros.

## II – METHODE DE MESURE

### 1 – Signal DTMF (Dual-tone multi-frequency)

Le signal de numérotation émis par un téléphone fixe s'appelle signal DTMF (Dual Tone Multi Frequency), il est produit par des oscillateurs qui superposent deux fréquences sinusoïdales correspondant à la paire de fréquence de la touche appuyée. Par exemple la '4' correspond à la superposition des fréquences 770Hz et 1209Hz.

Voici le tableau avec les fréquences normalisées sur le plan international :

| | | f5 | f6 | f7 | f8 |
|---|---|---|---|---|---|
| | | 1209 Hz | 1336 Hz | 1477 Hz | 1633 Hz |
| f1 | 697 Hz | 1 | 2 | 3 | A |
| f2 | 770 Hz | 4 | 5 | 6 | B |
| f3 | 852 | 7 | 8 | 9 | C |
| f4 | 941 Hz | * | 0 | # | D |

***Tableau : Fréquences de codage DTMF des différentes touches d'un téléphone fixe.***

Le choix de ces fréquences a été dicté par le fait que les distorsions harmoniques et d'intermodulation ne doivent pas perturber le bon fonctionnement du processus de reconnaissance de ces numéros. En effet, aucune fréquence n'est un multiple des autres fréquences DTMF, la différence entre deux fréquences quelconques n'égale pas celle d'aucune de ces fréquences DTMF, et la somme de deux fréquences quelconques n'égale pas, également, celle d'aucune de ces fréquences DTMF. L'erreur tolérée sur la valeur de ces fréquences est de 1.5%. L'erreur tolérée sur l'amplitude de ces fréquences est de 4 dB.

### 2 – Détection par filtrage passe bande.

La méthode que nous allons utiliser pour reconnaître les numéros DTMF repose sur l'utilisation des filtres numériques passes bandes. Pour cela, nous allons d'abord observer l'allure du spectre des signaux DTMF enregistrés au préalable. Ensuite, on va calculer les paramètres $\{a_n\}$ et $\{b_m\}$ des équations récurrentes des filtre passes bandes centrés sur l'une des fréquences DTMF. Il suffit alors de filtrer tous les signaux DTMF enregistrés par ce banc de 7 filtres passe bande et de mesurer la puissance sortante. Il est évidemment clair que deux puissances parmi les 7 mesurées seront largement supérieures aux 5 puissances restantes.

**A – Mesure des fréquences DTMF**

Ouvrir un script contenant les commandes suivantes :

- Utiliser la commande $[x, fe, Nbits] = wavrread('DTMF\ 1.wav')$; pour extraire le signal correspondant à un numéro 1 du dossier DTMF1 par exemple.
- Tracer le signal x avec la commande plot.
- Eliminer la voix gauche par la commande : $x = x(1:end/2)$;
- Eliminer la valeur moyenne du signal par la commande : $x = x - mean(x)$;
- Calculer le nombre d'échantillons du signal par la commande : $N = length(x)$;
- Définir l'axe fréquentiel $f = [0:\frac{f_e}{N}:f_e - \frac{f_e}{N}]$;
- Calculer le spectre X du signal x par la commande : $X = fft(x)$;
- Tracer le spectre d'amplitude en fonction de f.
- Mesurer, en Hz, les fréquences dans l'intervalle $[0:\frac{f_e}{2}]$. Ces fréquences sont elles dans le catalogue des fréquences DTMF.
- Refaites la même chose en choisissant le même numéro enregistré dans le dossier DTMF3. Interpréter la nature de ces fréquences parasites qui apparaissent dans ce spectre.

**B – Conception des filtres passes bande**

- Utiliser la commande : $[b, a] = butter(4, [677, 717]*2/fe, 'bandpass')$; pour concevoir un filtre passe bande de Butterworth, de 4ème ordre, centré sur la fréquence DTMF 697 Hz et de bande passante égale à 40 Hz.
- Utiliser la commande $[H, f] = freqz(b, a, 512, fe)$; pour calculer H(f) et tracer le module de H(f). Vérifier que la courbe obtenue est correcte.
- Refaite la même chose pour les 6 autres fréquences.

**C – Détection du numéro**

- Filtrer le signal enregistré par l'ensemble des 7 filtres passes bandes conçus par la commande : $y_1 = filter(b, a, x_1)$;
- Mesurer les 7 puissances de sortie à travers chaque filtre par la commande $p_1 = \mathrm{var}(y_1)$;
- Placer toutes les puissances calculées sur un vecteur $P = [p_1, p_2, p_3, p_4, p_5, vp_6, p_7]$; et le tracer par la commande *stem*.
- Conclusion.

# TP N° 4 AUTOCORRELATION ET INTERCORRELATION

## I - But

Le but de cette manipulation est de maîtriser la fonction d'autocorrélation et d'intercorrélation pour la mesure de la corrélation entre deux signaux. C'est une fonction importante pour la détermination de la vitesse radiale des étoiles par effet Doppler.

## II – PRINCIPES

### 1 – Autocorrélation

On définit la fonction d'autocorrélation $R_x(\tau)$ d'un signal x(t) d'énergie finie par : $R_x(\tau) = x(\tau) * x^*(-\tau) = \int_{-\infty}^{+\infty} x(t)x^*(t+\tau)dt$ . Cette fonction est paire pour les signaux réels. La fonction d'autocorrélation d'un signal périodique **est périodique et de même période $T_0$ que le signal**.

### 2 – Intercorrélation

La fonction d'intercorrélation mesure la similitude entre deux signaux décalés dans le temps. Elle permet aussi de mesurer le retard entre deux signaux identiques. On définit la fonction d'intercorrélation $R_{xy}(\tau)$ d'un signal x(t) d'énergie finie avec un autre signal y(t) d'énergie finie par : $R_{xy}(\tau) = x(\tau) * y^*(-\tau) = \int_{-\infty}^{+\infty} x(t)y^*(t+\tau)dt$ . Si la fonction d'inercorrélation est appliquée entre un signal x(t) lui-même mais décalé de T, alors **le maximum de cette fonction d'inercorrélation correspond au décalage T-N** avec N le nombre d'échantillons. Par conséquent on peut mesurer le retard entre un signal et lui-même avec une grande précision.

## III – Manipulation

### 1 – Etude de la fonction d'autocorrélation

Le but de cette première partie de cette manipulation est d'utiliser l'autocorrélation pour mesurer la fréquence fondamentale de la parole voisée considérée comme un signal périodique.

Le signal de la parole est émis par l'appareil phonatoire. Le processus de génération du son est composé des étapes suivantes :

- Génération d'un flux d'air par les poumons.
- Vibration des cordes vocales.
- Réalisation d'une disposition articulatoire dans les cavités vocales et nasales.

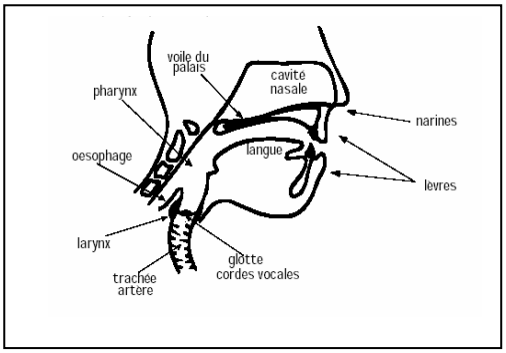
Le son est caractérisé par le pitch et le formant :

- Le **pitch** est la fréquence fondamentale de vibration des cordes vocales.
- Le son émis par les cordes vocales est articulé par le larynx, le pharynx et les cavités buccales, subit une modulation par la fonction de transfert de ce système. Elle comporte plusieurs fréquences de résonance. Il en résulte que tout son voisé possède plusieurs **formant**s tandis que le son non voisé n'en comporte pas.

Le signal de la parole est caractérisé au niveau acoustique par :

- L'intensité ou énergie.
- La fréquence fondamentale ou pitch (fréquence fondamentale des cordes vocales (les harmoniques → empreinte sonore).
- Le spectre : enveloppe spectrale de la parole (spectre de raies (voisé)).
- Le timbre : évolution temporelle des résonances (formants du conduit vocal et de leur largeur de bande).

| | Paramètre acoustique | Qualité perceptive |
|---|---|---|
| Hauteur | Fréquence | Grave – Aigu |
| Intensité | Amplitude | Faible – Fort |
| Timbre | Composantes spectrales | Sombre – Clair |
| Durée | Temps | Bref – Long |

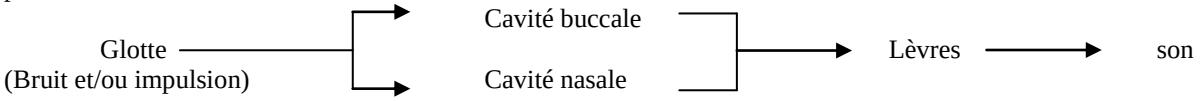



**Exemple de pitch :**
Homme : 70 – 250 Hz
Femme : 150 – 400 Hz
Enfant : 200 – 600 Hz

Le signal de la parole est considéré comme un signal quasi stationnaire. Il est stationnaire jusqu'à une durée de 100 ms, constitué par $\frac{1}{3}$ du bruit et par $\frac{2}{3}$ de pseudo périodique. Son spectre est étalé sur la bande comprise entre 20 Hz et 20 KHz. L'oreille est capable d'entendre des sons dont le spectre s'étale jusqu'à 20 KHz.

Le signal de la parole est produit par deux processus différents :
- Vibration des cordes vocales → son voisé : voyelles.
- Turbulence crée par l'air (bruit) → son non voisé : consonnes, chuchotement, respiration.

Le système acoustique est défini par l'excitateur : cordes vocales ou air propulsé + résonateur (cavité nasale et/ou buccal). La dynamique de la parole est la différence entre le signal le plus faible et le plus fort admissible par notre oreille.




***Modèle de production de la parole :***

La glotte est l'ouverture déterminée par les cordes vocales.

***Manipulation :***

- Enregistrer votre son voisé (la lettre « A ») sur une durée T de 1 s avec une fréquence d'échantillonnage $f_e$ = 8000Hz avec la commande : $x = wavrecord(N, f_e);\quad N = T * f_e$
- Définir l'axe temporel $t = [0 : T_e : T - T_e]$;
- Définir l'axe fréquentiel $f = [0 : \frac{f_e}{N} : f_e - \frac{f_e}{N}]$;
- Calculer le spectre X du signal x par la commande : $X = fft(x)$;
- Tracer le spectre d'amplitude en fonction de f. Pourquoi le spectre n'est pas décroissant ?
- Refaire le même calcul, cette fois ci, sur le signal $x_1 = x(3000 : 4000)$; . Commenter le résultat.
- Calculer l'autocorrélation du signal $x_1$ $R = xcorr(x, x)$;
- Tracer cette autocorrélation. Commentaire.
- Refaire la même chose en choisissant le signal x sur 1000 échantillons. Observer la position du premier pic juste après le pic maximum pour les deux durées du signal. Commentaire
- Utiliser juste la moitié droite de cette autocorrélation, c-à-d dépendante du temps positive : $[R_{\max}, k_{\max}] = \max(R)$; et $R_1 = R(k_{\max} : end)$; puis mesurer la période $T_0$ du signal.
- Localiser le premier pic en partant de l'origine et relever sa valeur $k_0$ correspondante. La période du pitch est la distance entre cette valeur de $k_0$ est celle de l'origine multiplié par $T_e$ : $T_0 = (k_0 - 1) * T_e$ et par conséquent la fréquence fondamentale est : $f_0 = \frac{1}{T_0}$. Commentaire.

**2 – Intercorrélation**

Le but de cette manipulation est de mesurer l'intercorrélation entre deux signaux identiques mais décalés dans le temps.

- Définir la durée T = 1s.
- Définir la fréquence d'échantillonnage fe=8000Hz. Avec Te=1/fe et N=fe*T.
- Définir l'axe temporel $t = [0 : T_e : T - T_e]$;
- Définir l'axe fréquentiel $f = [0 : \frac{f_e}{N} : f_e - \frac{f_e}{N}]$;
- Définir le signal porte suivant : ***x=[zeros(1,3975),0.1*ones(1,50),zeros(1,3975)]***;
- Ajouter du bruit au signal : x1=x+b ; avec ***b=a*randn(1,N)***; on donne à a les valeurs allant de 0.01 à 1.
- Définir le signal y qui est le signal x mais décalé de 100 échantillons par : ***y=[b,x1(1:end-100)]***;
- Tracer les deux signaux ave ***hold on***.
- Calculer le rapport signal sur bruit. ***Ksi = 10log10(var(x)/var(b))***.
- Calculer l'intercorrélation du signal y par rapport à x1 : ***R=xcorr(y,x1)***;
- Déterminer le maximum de cette intercorrélation en utilisant la commande ***[Rmax,kmax]=max(R)***;
- Tracer cette autocorrélation ***R***. Commentaire.
- Modifier l'amplitude de b. A partir de quel rapport signal sur bruit la détermination de ce décalage est parfaite ?
- Conclusion

# TP N$^{0}$ 5 VITESSE RADIALE DES ETOILES PAR EFFET DOPPLER

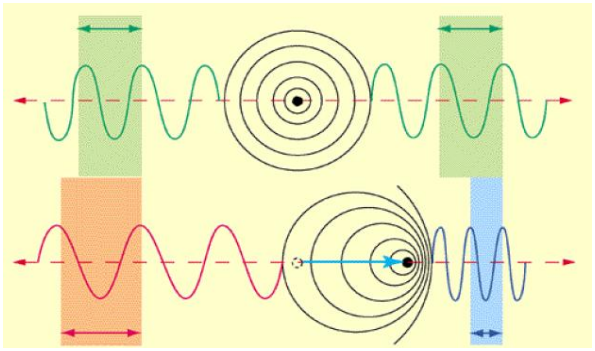
## I - But

Le but de cette manipulation est de maîtriser la procédure de mesure de la vitesse radiale des étoiles par effet Doppler en utilisant le spectre d'étoile mesuré à l'observatoire astronomique d'Oukaïmeden.

## II – PRINCIPES

### 1 – Effet Doppler

Lorsqu'une étoile se déplace dans l'espace, alors la lumière émise par cette étoile subit une variation par rapport à sa fréquence ou sa longueur d'onde perçue par l'observateur. Ce phénomène a été découvert par Christian Doppler en 1842 sur les ondes acoustiques et par Hypollyte Fizeau en 1848 sur les ondes électromagnétiques.



**Variation de la longueur d'onde apparente le long de l'axe de visée**

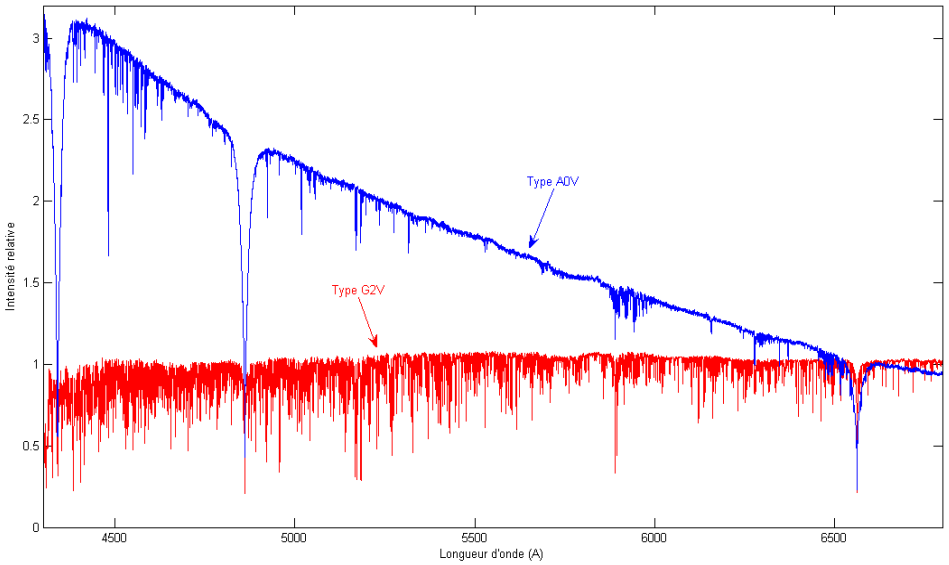
L'effet Doppler est très utilisé en astronomie pour la mesure de la vitesse de l'étoile par rapport à l'observateur et en même temps il permet de détecter le mouvement de la matière par rapport à l'étoile : mouvement de la photosphère dans le cas des étoiles pulsantes. La mesure de la vitesse radiale de l'étoile se fait en détectant le décalage des raies spectrales émises par l'étoile en comparant leurs longueurs d'ondes par rapport à celles mesurées en laboratoire. Il s'ensuit que les longueurs d'ondes des raies spectrales décalées vers le rouge (longueurs d'ondes plus grandes) indiquent que l'étoile s'éloigne de l'observateur et un décalage des raies spectrales vers le bleu (longueurs d'ondes plus petites) indiquent que l'étoile se rapproche de l'observateur. La formule qui relie le décalage de la longueur d'onde $\Delta\lambda$ par rapport à la vitesse radiale v de l'étoile est donné par :

$$\frac{(\Delta\lambda)}{\lambda} = \frac{v}{c}$$

Le principe de calcul de cette vitesse est basé sur la technique de **Cross Correlated Function** (CCF) ou intercorrélation. Il consiste à mesurer l'intercorrélation entre le spectre mesuré de l'étoile et le spectre synthétique correspondant à la même classe d'étoile appelé masque numérique. Ce masque numérique est réalisé sur la base de spectre d'étoile correspondant à sa classe avec un rapport signal sur bruit très élevé. Les raies spectrales d'absorption de ce spectre synthétique ont toutes la même profondeur et des largeurs spectrales différentes. La fonction CCF donne une meilleure précision de calcul du décalage $\Delta\lambda$. On doit exclure de cette corrélation les zones d'absorption de l'atmosphère (bandes telluriques $H_2O$ et $O_2$) et les raies de l'hydrogène qui ont une largeur trop grande.

**Remarque :**

Etant donné que le décalage Δλ qu'on veut mesurer dépend de la longueur d'onde λ cela signifie que les raies spectrales ne sont pas décalées de la même valeur Δλ ! Par conséquent une linéarisation du spectre est obligatoire pour pouvoir calculer le décalage Δλ par cette fonction de CCF.




En bleu : spectre mesuré d'une étoile
En rouge : spectre synthétique correspondant à la même classe d'étoile

**2 – Linéarisation**

Le principe de linéarisation en vitesse consiste à effectuer un changement de variable en remplaçant la longueur d'onde λ par une grandeur n appelée bin selon la formule suivante : $n = ALn(\lambda) + B$ avec A et B sont des constantes dépendantes des valeurs minimales et maximales de longueur d'onde λ.

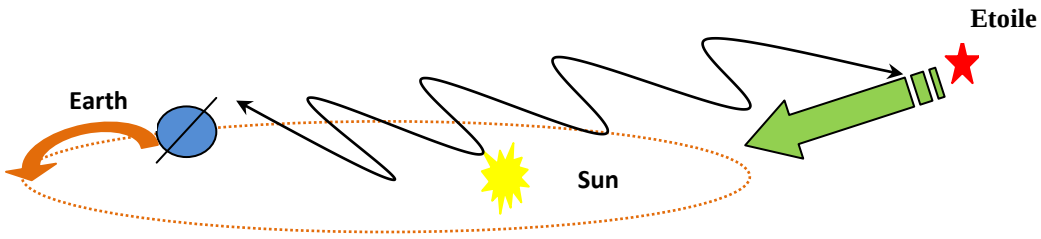
Soit N le nombre total de bin et $[\lambda_1, \lambda_2]$ est l'intervalle spectral analysé. Sachant que le bin n commence par la valeur 0 et s'étend jusqu'à la valeur maximale N, alors on trouve :

$$\begin{cases} A.Ln(\lambda_1) + B = 0 \\ A.Ln(\lambda_2) + B = N \end{cases} \quad \Rightarrow \quad A = \frac{N}{Ln(\lambda_2/\lambda_1)} \quad \text{et} \quad B = -\frac{NLn(\lambda_1)}{Ln(\lambda_2/\lambda_1)}.$$

Une fois la linéarisation réalisée sur les deux spectres, on applique l'intercorrélation, on en déduit la position du maximum de cette fonction CCF : Elle correspond à la valeur N+Δn comme nous avons pu le montrer durant la manipulation précédente. De cette valeur Δn on peut déduire la vitesse radiale par la formule suivante : $v = \Delta n \frac{C}{A}$ où C est la vitesse de la lumière. C = 299 792 458 m/s

**3 – Correction héliocentrique**

La vitesse radiale déduite de la formule de Doppler correspond à la vitesse de déplacement par rapport à l'observateur qui se trouve sur la terre. Or la vitesse radiale de l'étoile ne doit pas dépendre du mouvement de la terre au moment de l'observation. La terre peur être en rapprochement ou en éloignement au moment de l'observation du spectre de l'étoile. C'est pour cette raison qu'on mesure la vitesse radiale par rapport au repère héliocentrique pour s'affranchir de cette difficulté.




**Mouvement de l'observateur au moment d'enregistrement du spectre d'une étoile (correction héliocentrique)**

## III – Manipulation

Le but de cette manipulation est de mettre en œuvre la procédure de linéarisation et intercorrélation afin de mesurer la vitesse radiale de l'étoile Deneb prise comme exemple. Le spectre de cette étoile a été mesuré à l'observatoire d'Oukaïmeden en 2013 avec un spectrographe de résolution 12000 et sur une durée de 150s.

- Télécharger les deux fichiers des spectres de l'étoile Deneb et de son masque numérique.
- Tracer les deux spectres superposés. Mettre en évidence lé décalage des raies spectrales d'hélium D3 (λ5875.66Å) et de sodium D1 (λ5889.950Å) et D2 (λ5895.924Å).
- Calculer les valeurs de A et B correspondant à la linéarisation des deux spectres.
- Réaliser la linéarisation des deux spectres.
- Calculer l'intercorrélation entre le spectre de Deneb et le spectre du masque numérique : **R=xcorr(m,D)**;
- Extraire le décalage Δn. En déduire la vitesse radiale corrigée de la vitesse héliocentrique : **+ 9.8 Km/s**. Le mouvement de la terre nous rapproche de l'étoile à la vitesse 9.8 Km/s.
- Comparer à la valeur mesurée de la vitesse radiale de l'étoile Deneb : **- 4.9 Km/s**. (http://simbad.u-strasbg.fr/)
- Conclusion.