Licence STS Mention E.E.A.

parcours Fondamental

Bureau d'Etude Matlab

rev 4.02

H. Carfantan - S. Hosseini - P. Castelan

Table des Matières

I - Calculs Scalaires et Matriciels **1**
- Introduction 1
- Premiers contacts avec Octave/Matlab 1

II - Entrées / Sorties et Graphiques **6**
- Objectifs 6
- Entrées/Sorties simplifiées 6
- Les scripts 6
- Tracé de graphiques 7

III - Programmer avec Matlab/Octave : le langage **10**
- Objectifs 10
- Les Instructions Octave de Contrôle de Flux : 10
- Lecture de fichiers 11
- Les Fonctions 12
- Diagramme de Bode 12

IV - Transformée de Fourier **15**
- Rappels sur les séries de Fourier et la Transformée de Fourier Discrète (TFD) 15
- Expressions de la Transformée Discrète 17
- TP Fourier : signal sinusoïdal 17

V - Etude de la Transformée de Fourier de signaux simples **19**
- Signaux "sinusoïdaux" 19
- Signaux "carrés" 20
- Utilisation de la Transformée de Fourier incluse dans Octave 20
- Conservation de l'énergie 20
- Nombres de points transformés. 21

VI - Résolution Fréquentielle du Spectre **23**
- Introduction 23
- Signaux non périodiques : influence sur le spectre 24
- Méthodes utilisées pour corriger le problème de la Transformée Asynchrone 24

VII - Présentation succincte des filtres RII **27**
- Présentation 27
- Présentation succincte des filtres numériques 27
- Application au filtre passe-bas du premier ordre 28
- Filtre passe-bas du second ordre 29

VIII - Débruitage **31**
- Objectifs 31
- Identification de la réponse impulsionnelle d'un filtre 31
- Débruitage d'un signal avec référence de bruit 32
- Application Audio 32
- Application biomédicale 33
- Annexe: Fonctions Matlab xcorrel et filter 33

I - Calculs Scalaires et Matriciels

I. Introduction

I.1. Préliminaire

Le but de ce cours est de vous familiariser avec l'environnement Octave que vous utiliserez dans la suite de vos études et dans votre vie professionnelle.

Le Langage Octave est, en effet, largement utilisé en laboratoire et en entreprise. Dans le cadre de cette formation, vous utiliserez Octave un clone gratuit de Matlab.

Le message que vous devrez retenir, si vous devez en retenir un, est le recul nécessaire que doit prendre l'utilisateur de tels langages vis à vis des résultats retournés. Par définition un calcul est attaché à un modèle de la réalité, celui-ci n'est pas forcément une représentation exacte de celle-ci. Il faut donc toujours regarder les résultats obtenus avec un oeil critique.

I.2. Objectifs

Prise de contact avec le logiciel (savoir naviguer dans les menus)

- Utilisation de l'aide en ligne
- Calculs élémentaires
- Opérateurs matriciels

II. Premiers contacts avec Octave/Matlab

Dans tout ce qui suit, il est demandé d'utiliser des fonctions/commandes d'Octave que vous ne connaissez pas. Toutefois, il vous sera systématiquement donné un point de départ, utilisez alors l'aide en ligne pour connaître la syntaxe des commandes.

Dans Octave, il existe 3 types de fenêtres :

- la fenêtre de commande,
- les fenêtres graphique,
- et les fenêtres d'édition dans lesquelles on définit les fichiers de programmation (M-File) contenant une série d'instructions.

"**>**" est le début d'une ligne de commande. On peut écrire plusieurs commandes par ligne séparées par ";" mais pour des questions de lisibilité, il vaut mieux éviter cette démarche.

Il est possible (et même conseillé) de commenter ses programmes. En langage Octave, les commentaires sont précédés du signe "**%**". Cela sera surtout utile lorsque vous programmerez sous Octave.

II.1. Historique des commandes

Octave maintient un "fichier historique" des commandes saisies depuis son démarrage. Il est donc possible de ré-utiliser en modifiant ou non les dernières commandes en utilisant les touches du clavier un peu à la façon du "doskey" (cf. l'aide-mémoire).

II.2. Calculs élémentaires scalaires

Le langage Octave connaît les opérateurs classiques (**+**, **-**, *****, **/**, **sqrt**, **exp**, **log**, **log10**, **^** - puissance entière ou réelle-, etc.), trigonométriques (**sin**, **cos**, **tan**, **cot**), hyperboliques (**sinh**, **cosh**, **asinh**, etc.).

➡ *Attention : les angles sont exprimés en radians !*

Le respect des priorités se fait en utilisant les parenthèses.

Certaines constantes sont déjà définies comme "**pi**", "**inf**" (l'infini), "**NaN**" signifiant 'Not a Number' qui exprime une indétermination comme une division par 0.

Pour calculer 4^2 il suffit de saisir : "4^2" et de valider.

```
> 4^2
ans = 16
```

Si le nom de la variable n'est pas spécifié, le résultat du calcul est stocké dans la variable "**ans**" (pour answer). Le contenu est conservé jusqu'à ce qu'il soit remplacé.

Il est possible de spécifier le format d'affichage des calculs. Plusieurs formats sont disponibles :

short, **short e**, **long**, **long e**, **hex**, **rat**

Le sens de ces mots clés est leur sens évident, **hex** affiche en hexadécimal, **rat** en format rationnel.

Ceci doit être spécifié avant de demander le contenu de la variable. Le type de format choisi est conservé ensuite, quelle que soit la variable jusqu'à un changement du format.

```
> pi
ans = 3.1416
> format long
> pi
ans = 3.14159265358979
> format long e
> pi
ans = 3.141592653589793e+00
```

Pour en savoir plus sur les formats de sortie (=affichage), on peut demander l'aide en ligne par :

```
>help format
```

ou

```
>doc format
```

Pour supprimer l'affichage de l'exécution d'une instruction, la faire suivre par un "**;**".

Octave sait gérer les nombres complexes. **i** est le nombre complexe (**j** peut aussi être utilisé).

```
>(1+2j)*(1+j)
ans = -1.0000 + 3.0000i
```

➡ *Attention : pour éviter toute ambiguïté, réservez la lettre "**i**" à la définition de la partie imaginaire d'un nombre complexe et bannissez son utilisation en tant que variable.*

Toutes les fonctions d'Octave savent traiter les nombres complexes ainsi :

```
>exp(i*pi/2)
ans = 6.1232e-17 + 1.0000e+00i
```

Il est possible de définir des variables en leur affectant une valeur.

Le nom d'une variable *débute* forcément *par une lettre*. Octave fait la distinction entre minuscules et majuscules. Il n'y a pas de spécification particulière suivant le type de variable (entier, réel, complexe, matrice,..) comme dans certains logiciels.

```
>a=(1+4i)/(1-4i)
a = -0.88235 + 0.47059i
```

- Calculez la partie réelle, la partie imaginaire, l'**angle**, le module de la variable "a" définie ci-dessus et son complexe conjugué (utilisez l'aide en ligne sur la fonction **real**, **angle** et **abs**).

➡ *Vous utiliserez pour ce faire l'aide en ligne (**help**) et regarderez la rubrique "see also" pour trouver les mots clés non indiqués.*

On peut avoir accès à la liste des variables déjà utilisées, en supprimer certaines de la mémoire :

Consultez l'aide en ligne de **who** et **clear**.

Si Octave sait calculer avec des scalaires, il a été conçu pour le calcul matriciel (après tout, un scalaire est une matrice 1*1).

II.1. Saisies de matrices (vecteurs & matrices)

Pour saisir une ligne, il suffit d'affecter à une variable une liste de valeurs entre crochets droits. On marque le passage à la ligne suivante par le "**;**". Le changement de colonne est effectué par un espace ou "**,**".

```
>a=[1 2 ; 3 4]
a =
1 2
3 4
```

Les éléments d'une matrice peuvent être des expressions à évaluer, une matrice,... Il est inutile d'en préciser les dimensions.

Exercice I-1

Soit la série de nombres suivants :

2 ; 3 ; $7+5j/2$; $\exp(-i\pi/2)$; $\sin^4(\pi/3)$; $\exp(0,34)^2$

- Créez un vecteur ligne **L** contenant ces éléments.
- Créez un vecteur colonne **C** contenant ces éléments.
- Créez une matrice **M** de 3 lignes contenant ces éléments.
- Donner **M(1,2)**, **M(2,1)**, **L(4)** et **C(4)**.
- Utilisez **size** et **length** pour déterminer les dimensions de **M**, **L**, **C**.

Octave sait aussi générer des vecteurs avec un pas constant.

Regardez l'aide en ligne sur la fonction **linspace** et le raccourci "**:**"

Exercice I-2

- Définissez un vecteur ligne **A1** dont la 1ère valeur vaut 0,2 la dernière 0,85 (les données sont équidistantes de 0,1).
- Définissez un vecteur ligne **A2** dont la 1ère valeur vaut 2 et la dernière 12 (les données sont équidistantes de 1).
- Définissez un vecteur ligne **A3** contenant 7 valeurs dont la 1ère valeur est 0,2 et la dernière 0,8.

ou avec un pas logarithmique (**logspace**) :

- Définissez un vecteur ligne **A4** contenant 4 valeurs dont la 1ère est 10^{-1} et la dernière est 10^2.

II.2. Opérations scalaires sur les matrices

Par exemple multiplier un vecteur par un scalaire :

Les valeurs suivantes correspondent aux abscisses des points d'une droite :

```
-3 ; -1 ; 0.5 ; 1 ; 2.5 ; 3 ; 4 ; 8 ;10
```

la pente de cette droite **y(x)** est 2 et l'ordonnée à l'origine est 0,5.

Exercice I-3

- Saisissez ces données dans un vecteur **x** et calculez le vecteur **y** correspondant aux ordonnées.
- Affichez **y(3)**. La valeur vous semble-t-elle correcte ?

II.3. Sélection de partie de matrices (sous matrices) - transposition

Le logiciel offre de nombreux outils de manipulation des matrices les opérateurs de sélection "**(**" et "**)**", et l'opérateur **colon** ou "**:**" permettent de créer des sous sélections.. L'opérateur de transposition est l'apostrophe " ' ". Enfin "**[ ]**" signifie matrice vide.

Exercice I-4

Soit la matrice **M1** = [3 : 3 : 9 ; 0 : 2 : 5 ; 15 : -5 : 5].

- Définissez la sous matrice S1 contenant les deux dernières lignes de **M1**.
- Supprimez la 2ème colonne de **S1**.

Soient le vecteur ligne **L** = [9 : -1 : 7] et le vecteur colonne **C** = [1 : 3].

- Ajoutez la ligne **L** à la matrice **M1**.
- Ajoutez la colonne **C** à la matrice **M1**.

Octave offre quelques fonctions particulières, souvent pratiques pour la création de matrices : les fonctions **ones**, **zeros**, **eye**.

Exercice I-5

- Créez une matrice 3 lignes et 4 colonnes contenant des "0".
- Créez une matrice 4 lignes et 3 colonnes contenant des "5".
- Créez une matrice 3 lignes et 3 colonnes contenant des "4" sur la diagonale.
- Avec une seule instruction, définir la matrice **A** suivante :

```
-3    -3    -3     0     0
-3    -3    -3     0     0
-3    -3    -3     0     0
 1     0     1     1     1
 0     1     1     1     1
```

- Extrayez la diagonale de **A** et sa transposée à l'aide de la fonction **diag**.

II.4. Calcul matriciel

Octave peut, évidement, réaliser des sommes et produits matriciels ; les opérateurs sont les mêmes que pour les produits et sommes scalaires. De même il est possible d'utiliser les opérateurs de comparaison sur une matrice (égalité (**==**), **>**, **<**, **<=**, différent de (**~=**)...).

Les opérateurs logiques sous Octave sont :

```
et    &
ou    |
non   ~ (tilde)
```

Octave sait aussi réaliser des opérations *terme à terme*. Le symbole de l'opérateur est alors précédé du point "**.**".

Exercice I-6

- Calculez la somme des matrices **A** = [1 2 3 ; 4 5 6 ; 7 8 9] et **B** = [1 2 3 ; 5 -8 9 ; 5 2 0], la différence, le produit, la division de **A** par **B**, et de **B** par **A**.
- Créez une matrice **C** identique à **A**, hormis les éléments de **A** inférieurs ou égaux à quatre qui seront remplacés par zéro.
- Créez une matrice **D** identique à **B**, hormis les éléments de **B** égaux à cinq qui seront remplacés par zéro.

Il existe les opérateurs **prod** et **sum** qui réalisent le produit des éléments d'un vecteur, ou la somme des mêmes éléments.

Exercice I-7

- Calculez 10!
- Calculer la somme **S** de la série géométrique si : **S** = 1 + **s** +$\mathbf{s}^2$ +...+$\mathbf{s}^n$ avec **s** = 2/3. Comparer **S** avec la limite théorique 1/(1-**s**) en prenant **n** = 10 puis **n** = 100.

Il existe bien d'autres opérateurs permettant, par exemple, de déterminer la valeur **max**imale, **min**imale, ou de calculer la valeur moyenne des éléments d'un vecteur (consulter l'aide en ligne).

II - Entrées / Sorties et Graphiques

I. Objectifs

- Connaître les instructions d'entrée / sortie simplifiées.
- Savoir écrire un programme élémentaire avec Octave.
- Savoir tracer des graphiques.

II. Entrées/Sorties simplifiées

La commande **input** permet d'afficher un message d'invite à la saisie d'une variable.

Tapez dans le terminal :

- a=**input**('Entrez a : ')
- Entrez comme réponse 2, puis [1 2 ; 3 4]

Il est possible d'obtenir un affichage formaté avec la commande **sprintf**. La syntaxe de **sprintf** est la même qu'en langage C :

```
sprintf( 'chaine de formatage', liste de variables )
```

La chaîne de formatage peut contenir du texte ou des commandes de format pour les variables à afficher -comment les interpréter- ou des commandes pour l'affichage -passage à la ligne par exemple-.

Les variables sont affichées en tant qu'entiers avec **%d**, en tant que réels avec **%f**. Il est possible de spécifier la longueur et le nombre de décimales affichées avec le mode **%f**. Il suffit d'écrire ces informations entre **%** et **f**. Par exemple **%8.3f** affichera avec 8 caractères dont 3 décimales.

- Depuis le terminal, affichez le message : La valeur de pi est 3.142

Il est possible de supprimez de l'affichage le mot "**ans**" à l'aide de la commande **disp**. La syntaxe sera alors : **disp(sprintf ( ))**.

- Définissez une variable A comme étant un vecteur de 3 valeurs équiréparties allant de 0 à 2π, affichez A à l'aide d'un **sprintf**, les valeurs étant affichées sur 6 caractères et 3 décimales.

III. Les scripts

Ce sont des fichiers texte comportant une listes d'instructions. Ils doivent *OBLIGATOIREMENT* porter l'extension "*.m*", d'où le nom de «M-File».

Pour éviter que le résultat partiel du calcul ne s'affiche à l'écran, il suffit de mettre à la fin de l'instruction un "**;**". Ces "**;**" ne servent pas à séparer les instructions comme en langage C, mais seulement à supprimer l'affichage.

Exercice II-1

- Écrivez un script définissant **y** = 1,5*sin(**x** + π/2) en ayant échantillonné **x** sur 50 intervalles entre 0 et 2π, d'abord avec l'opérateur colon puis avec l'opérateur **linspace**.
- Lancez le script, puis affichez les dimensions des vecteurs **x** et **y** dans la fenêtre de commande. Les valeurs des variables **x** et **y** ne doivent pas apparaître à l'écran.
- Dans ce même script, calculez le pas **h** *sur les abscisses* de deux manières différentes et affichez-le d'abord sans format, puis formaté (3 décimales).

- Modifiez votre script pour qu'il saisisse à l'aide de la commande **input** le nombre de points. Recalculez **x** et **y** et le pas **h** en fonction de cette donnée.

IV. Tracé de graphiques

Octave offre de nombreux outils pour tracer des courbes en 2D. Regardez l'aide en ligne sur **plot**.

La syntaxe est la suivante : **plot (x, y, format)**

format est une *chaîne de caractères* qui précise le mode de tracé. On peut préciser la couleur du tracé avec les codes de couleur :

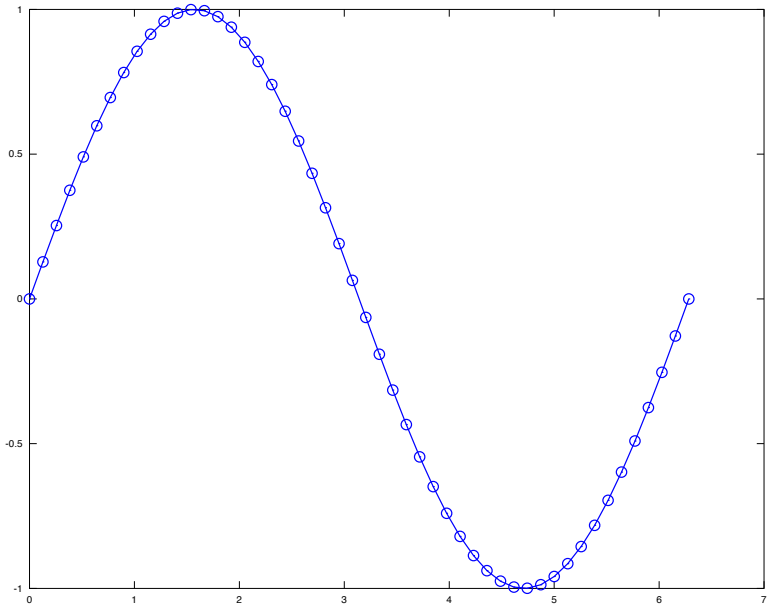
blue, red, green, magenta, cyan, white, black

et le type de tracé : en ligne avec un tiret ou bien en mode point :

symbole	+	o	*	.	x	s	d	^	v	>	<
marker	+	o	*	.	✖	■	♦	▲	▼	▶	◀

Par exemple la courbe suivante est obtenue avec la commande :

plot (x, y, 'bo')



Octave utilise Gnuplot pour les tracés. Il faut donc parfois demander au grapheur de retracer les courbes pour mise à jour (commande **replot**) mais c'est de plus en plus rare. On peut fermer le flux de données pour le tracé à l'aide de la commande **close**. On peut effacer un tracé avec **clf** ou **clg**, et avoir plusieurs fenêtres avec **figure**. **clearplot** (obsolète) et **clg** sont identiques et effacent le graphe. Par contre **clf** (clear figure) efface la fenêtre. Ceci sera mis à profit avec la fonction **subplot**, qui permet de tracer plusieurs graphes dans une même fenêtre.

Les axes auront un intitulé avec **xlabel** et **ylabel** et le tracé un titre avec **title**. On peut afficher une grille avec **grid**, ajuster les axes avec **axis**. Il existe des fonctions de tracés spécialisés (**bar**, **loglog**, **semilogx**, **semilogy**, **polar**, **stairs**). Il est possible d'annuler une commande **subplot** avec la commande *subplot(1,1,1*).

La commande **axis** sert à fixer les bornes du tracé. Elle accepte un seul paramètre qui est un vecteur dans lequel on précise les valeurs minimales et maximales de chaque axe. Pour un graphe en 3D la commande **axis** aura comme paramètre un vecteur du type

```
[ xmin xmax ymin ymax zmin zmax ]
```

La commande **axis** peut aussi être utilisée de façon non fonctionnelle en précisant le mot clé **tight** pour ajuster automatiquement les axes aux valeurs minimales et maximales des courbes tracées. Reportez-vous à l'aide en ligne sur la commande **axis**.

La commande **legend** sert à afficher une légende. Ses paramètres sont des chaines de caractères (une chaine par courbe tracée).

Exercice II-2

- Créez un nouveau script.
- Dans ce script, tracez la courbe définie par **y** = 1,5*sin(**x** + π/2) en ayant échantillonné la fonction sur 50 *intervalles* entre 0 et 2π.
- Imposez les valeurs minimale et maximale de l'axe des abscisses à 0 et 10 respectivement.
- Imposez les valeurs minimale et maximale de l'axe des ordonnées à -2 et 1 respectivement.
- Indiquez le nom des axes, donnez une légende et un titre.
- Dans le terminal, à l'aide de la commande **axis**, fixez les dimensions des axes à leur valeur de optimale.
- Superposez à la courbe tracée précédemment celle définie par **y** = exp(-**x**/5) en ayant échantillonné sur 100 *intervalles* entre 0 et 10. Utilisez pour ce faire la commande **hold**.

Exercice II-3

- Créez un nouveau script.
- Tracez les 2 courbes suivantes sur un même graphe, l'une en symboles, l'autre en traits pleins et de couleurs différentes :

  y = 10 sin (**x** + π/3)

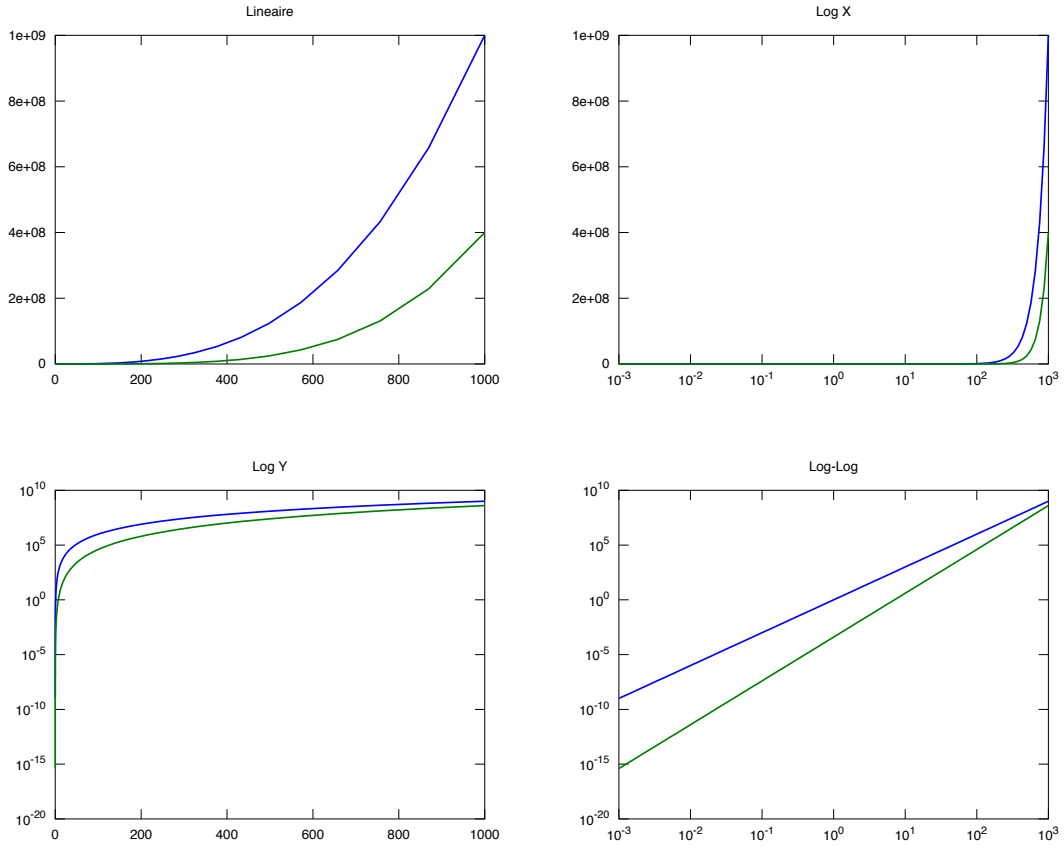
  z = 20 sin (**x** + π) + **y**

  le tout échantillonné sur 50 *intervalles* entre 0 et 2π.

➡ *On n'utilisera pas ici la commande **hold**.*

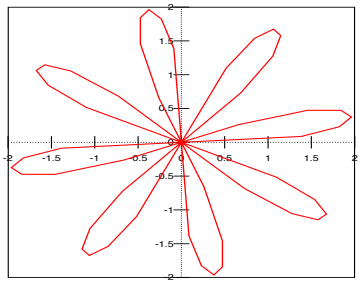
Exercice II-4

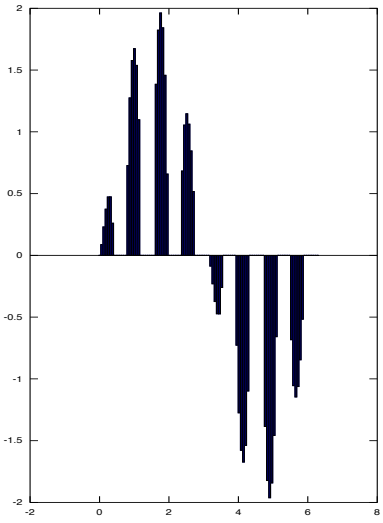
- Créez un nouveau script.
- Tracez les courbes définies par **y** = x^4 / 2500 et **y** = x^3 avec **x** compris entre 10^{-3} et 10^3 échantillonné logarithmiquement sur 100 points. Commencez avec un tracé en échelle linéaire puis en **semilogx** / **semilogy** et finalement en **loglog**.
- Utilisez la commande **subplot** pour avoir les 4 courbes dans la même fenêtre graphique (définir une fenêtre pouvant contenir 4 courbes (2 lignes et 2 colonnes)).



Exercice II-5

- Créez un nouveau script.
- Créez une fenêtre graphique pouvant contenir deux graphes.
- Dans le premier graphe, tracez en coordonnées polaires $y^2 = 2*\sin(8x)$ sur 100 intervalles avec x allant de 0 à 2π.
- Dans le second graphe, tracez en mode graphique à barre $z = y \sin(x)$ avec $y^2 = 2 \sin(8x)$ sur 100 intervalles entre 0 et 2π.





III - Programmer avec Matlab/Octave : le langage

I. Objectifs

- Savoir exploiter des données contenues dans un fichier.
- Connaître les instructions de base de la programmation Matlab/Octave.

II. Les Instructions Octave de Contrôle de Flux :

Octave comporte 5 opérateurs de contrôle de flux, Matlab 4. Contrairement à ce qui est indiqué dans l'aide en ligne, on utilisera pas les marqueurs de fin spécialisés. Ainsi l'instruction **if** sera close par un **end** et non un **endif**. Matlab ne connaissant pas les marqueurs spécialisés vous diminueriez la portabilité de votre script.

II.1. **if** ...**else** ... **elseif** ... **end** -Test, aiguillage binaire-

Il s'agit du test binaire qui offre deux aiguillages possibles pour le flux du programme. Il est possible avec **elseif** d'augmenter le nombre d'aiguillages.

Toutefois on préférera dans le cas où les possibilités sont nombreuses utiliser **switch**. La syntaxe est :

```
if ( condition )
       instructions…
else
       instructions…
end
```

II.2. **switch** / **case** -Aiguillage multiple-

Il s'agit de l'aiguillage multiple. Il est possible de construire autant de cas que voulu. Ne jamais omettre le cas "autrement". La syntaxe est :

```
switch ( expression )
       case { liste de valeurs }, instructions…
       case { liste de valeurs }, instructions…
       otherwise, instructions…
end
```

Il n'y a *pas besoin* de l'instruction *break* comme en langage C. De même, les parenthèses sont obligatoires. Enfin, Il faut qu'il y ait au moins un **case**. De plus, il est fortement recommandé de mettre un **otherwise**.

II.3. **while** (Tant que ...) - Boucle -

C'est la boucle conditionnelle de base.

➡ *Attention ! Les conditions logiques sont parfois complexes.*

➡ *Attention, il faut modifier la condition dans la boucle sous peine de n'en jamais sortir.*

La syntaxe est :

```
while ( condition )
       instructions…
end
```

II.1. **do** ... **until** (répéter jusqu'à)

C'est l'autre boucle conditionnelle. Cette boucle, contrairement à la boucle **while**, est toujours parcourue une fois. Le test est effectué *a posteriori* ce qui est parfois utile.

Bien entendu, comme pour la boucle **while**, il faut modifier la condition dans la boucle sous peine de n'en jamais sortir. La syntaxe est :

```
do
        instructions…
until ( condition )
```

➡ *Attention cette instruction n'existe que pour Octave, Matlab ne la connait pas. Ne pas l'utiliser si la compatibilité Matlab est indispensable.*

II.1. **for** -Boucle avec compteur-

C'est la boucle de base du calcul. L'index n'est pas forcément un entier. Le pas est par défaut égal à un. Si l'on veut une boucle décroissante, le pas doit être spécifié négatif.

➡ *Attention : Il ne faut pas modifier l'index dans la boucle, c'est Octave qui s'en charge !*

La syntaxe est :

```
for index = debut : pas : fin
        instructions…
end
```

III. Lecture de fichiers

III.1. Présentation

De plus en plus souvent, les données sont contenues dans des fichiers et on utilise un logiciel de traitement de données. Octave peut parfaitement remplir ce rôle.

La fonction **load** permet de lire un fichier et de stocker son contenu dans une matrice. Sa syntaxe est :

```
load -options nom_de_fichier
```

ce qui crée une variable portant le nom du fichier. Les options sont nombreuses, nous n'utiliserons que l'option **-ascii** signifiant que les données sont au format texte sous forme de colonnes séparées par une tabulation et sans en tête. Reportez-vous à l'aide pour plus de détails.

- Depuis le terminal chargez le contenu du fichier 'sonde.txt' dans une variable appelée **sonde**. Affichez les dimensions de **sonde**.
- Les abscisses de la courbe sont dans la première colonne, les ordonnées dans la deuxième. Tracez la courbe en mode point.

Il est possible de charger le contenu du fichier dans une variable en utilisant la syntaxe fonctionnelle :

```
M = load( '-ascii' , 'nom_du_fichier' )
```

III.2. Application

Créez un script qui :

Exercice III-1

- Demande le nom du fichier à lire (commande **input**).
- Lise les données du fichier dont le nom vient d'être saisi.

- Détermine le nombre de colonnes du fichier.
- Puis, en considérant que la première colonne contient les abscisses de toutes les courbes, trace les courbes dans une seule figure. Pour ce faire, tester l'une après l'autre, les instructions **while**, **do until** et **for**.
- Utilisez ce script avec les fichiers "sonde.txt" et "sond_der.txt".
- Complétez le script pour qu'il demande à l'utilisateur quel type de boucle utiliser (on impose l'utilisation d'un **switch**). Les choix autorisés sont : *'for'*, *'while'* ou *'do'*.
- Complétez le script pour que si l'utilisateur entre une réponse non autorisée, la question du type de boucle à utiliser lui soit posée à nouveau.
- Complétez le script de telle sorte que l'utilisateur n'ait pas à entrer l'extension du fichier.

IV. Les Fonctions

Ce sont des fichiers comportant des listes d'instructions, voire des fonctions. Nous allons écrire plusieurs fonctions. Voici la syntaxe d'une fonction Octave :

```
function [variables résultats] = <Nom>(variables d'entrée)
```

Il peut y avoir plusieurs fonctions dans un fichier **M-file**, toutefois seule la première fonction est accessible depuis la fenêtre de commande. Les autres fonctions du fichier M-File ne sont accessibles qu'aux autres fonctions du fichier.

➡ *Notez la présence des ";" dans l'écriture de la fonction comme dans l'écriture des scripts. Ces ";" servent à éviter que des résultats partiels de calcul ne s'affichent à l'écran.*

- Écrivez une fonction qui calcule le cube d'un nombre.

```
function z = cube(x)
z=x*x*x;
```

- Créez un nouveau fichier Octave et sauvegardez-le sous le nom "cube.m".

Attention :

➡ *Le nom du fichier doit être le même que celui de la fonction.*

➡ *L'extension du fichier doit obligatoirement être ".m"*

- A l'aide de la fonction cube calculez le cube de 3. Puis de la variable **A** qui prendra les valeurs : A = 3 ; A = 3 + i ; A = [1 2 ; 3 4].
- Affichez dans le terminal la valeur de la variable **z** et de la variable **x**.

V. Diagramme de Bode

Matlab/Octave ne prend sa vraie dimension qu'en l'utilisant pour ce qu'il est : un logiciel de calcul matriciel. Comme exemple d'application vous allez écrire une fonction qui trace le diagramme de Bode d'un système dont on connaît la fonction de transfert.

Le diagramme de Bode représente l'évolution fréquentielle de la réponse d'un système : son Gain et sa Phase. On place sur l'axe horizontal les pulsations (en pratique souvent les fréquences) sur une échelle logarithmique, et en ordonnée le gain en déciBels et la phase en degrés ou radians.

V.1. Tracé du diagramme de Bode (ordre 2)

Exercice III-2

- Écrivez un script appelé bode.m qui trace le diagramme de Bode entre 10^{-3} et 10^{3} Hz sur 120 points d'un système ayant pour fonction de transfert :

$$H(p) = \frac{1}{p^2 + 2p + 1}$$

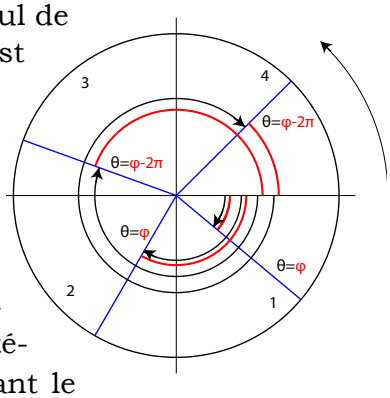
- en utilisant la commande **polyval** calculez le gain naturel du filtre.
- Dans une figure pouvant comporter deux graphes, tracez, en fonction de la pulsation, dans le premier le gain en décibels et dans l'autre graphe l'évolution de la phase en degrés.
- Recommencez les points précédents avec la fonction :

$$H(p) = \frac{1}{p^3 + 3p^2 + 3p + 1}$$

Que constatez-vous pour le diagramme des phases ? Comment corriger ce problème ? (Quel est le domaine d'arrivée de la fonction **angle** ?)

V.2. Correction de la phase

Les problèmes apparaissant lors du tracé sont dus au mode de calcul de l'argument correspondant à un nombre complexe. Cet argument est défini à 2π près et le domaine de définition va de - π à π, ainsi que le montre la figure.



La correspondance entre l'angle de rotation de phase dans le diagramme de Bode et l'angle retourné par la fonction angle d'Octave est donnée jusqu'à un circuit d'ordre 4. Sur cette figure est représenté en noir l'angle de rotation de phase et en rouge vers l'extérieur l'angle retourné par la fonction angle (sans flèche). En suivant le même raisonnement on peut exprimer cette correspondance pour un système d'ordre quelconque.

Exercice III-3

- Ecrivez une fonction cadre qui retournera le numéro du cadran c dans lequel se trouve un nombre complexe z. L'interface sera :

  function c = cadre(z)

  La numérotation des cadrans sera celle de la figure ci-dessus.
- Modifiez votre script bode de façon à ce que la phase soit corrigée dans une boucle **for**. Dans cette boucle, vous retrancherez systématiquement 2π à la phase de tout complexe **H(k)** se trouvant dans les cadrans 3 et 4.

V.3. Généralisation

Exercice III-4

- Tracez le diagramme de Bode du système ayant la fonction de transfert suivante :

$$H(p) = \frac{1}{p^5 + 3,2361p^4 + 5,2361p^3 + 5,2361p^2 + 3,2361p + 1}$$

Avec un ordre supérieur à 4 la phase de H fait plus d'un tour. A haute fréquence la phase de **H(k)** se trouve à nouveau en cadran 1, la correction n'est donc plus opérationnelle.

Il faut, donc, corriger systématiquement la phase dès que le gain complexe passe du cadran 2 au cadran 3.

La mise en place, non d'une détection de cadran, mais *d'un changement de cadran* (par exemple : **H(k)** en cadran 3, et **H(k-1)** en cadran 2) permet de corriger le problème.

Pour que la correction de phase s'applique à tous les points :

Exercice III-5

- Définissez une variable **cor** que vous initialiserez (à quelle valeur ?).
- Ajoutez **cor** à la phase calculée pour chaque **H(k)** : la variable **cor** sera décrémentée de 2π lors des détections de passage du cadran 2 au cadran 3.
- Modifiez le script précédent pour qu'il puisse tracer le diagramme de Bode d'un système quelconque F(p)/H(p) en corrigeant les phases.
- Tracez H(p) (celui de l'exercice III-4)
- Tracez les diagrammes de Bode des fonctions de transfert ayant comme numérateur les fonctions suivantes et le dénominateur H(p) de l'exercice III-4 :
  - Dérivé de Butterworth : $1 + 2{,}6131p + 3{,}4142p^2 + 2{,}6131p^3 + p^4$
  - Dérivé de Bessel : $105 + 105p + 45p^2 + 10p^3 + p^4$
  - Dérivé de Tchebyshev : $1 + 2{,}4447p + 3{,}1705p^2 + 2{,}1771p^3 + 1{,}2069p^4$

Exercice III-6

- Transformez le script bode.m en fonction ayant les paramètres d'entrée suivants :

Fréquence minimale, Fréquence maximale, nombre de points par décade, les coefficients du polynôme au numérateur, les coefficients polynôme au dénominateur.
Cette fonction ne retournera rien. Son interface sera donc :

```
function bode(Fmin, Fmax, Ndec, N, D)
```

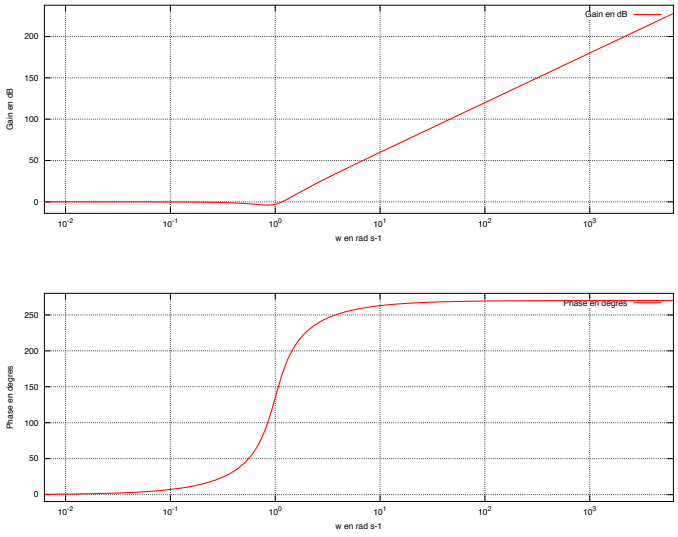
Par exemple pour générer le diagramme de Bode de $H_3(p)$, vous aurez à appeler la fonction comme suit : Bode(1e-3, 1e3, 20, 1 , [1 3 3 1]).

$$H_3(p) = \frac{1}{p^3 + 3p^2 + 3p + 1}$$

Tracez le diagramme de Bode du filtre ayant la fonction de transfert :

$$H(p) = \frac{p^5 + 3,2361p^4 + 5,2361p^3 + 5,2361p^2 + 3,2361p + 1}{p^2 + 2p + 1}$$

Voici ce que vous devriez obtenir, si vous avez pensé à tout...



IV - Transformée de Fourier

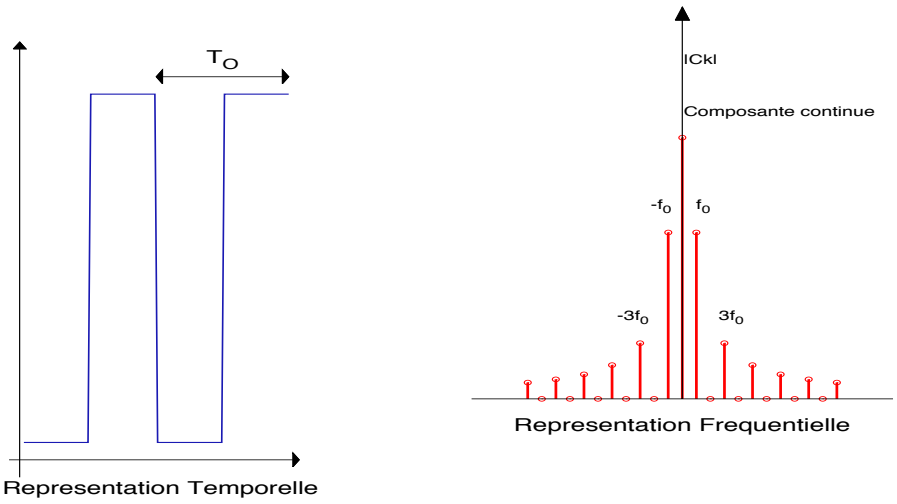
I. Rappels sur les séries de Fourier et la Transformée de Fourier Discrète (TFD)

I.1. Développement en série de Fourier d'un signal analogique périodique.

Tout signal analogique périodique peut s'écrire comme une somme de fréquences pures (exponentielles complexes ou couples de sinus/cosinus). Plus précisément, le développement en série de Fourier d'un signal **x(t)** période T_0 c'est-à-dire tel **x(t+T₀) = x(t)** s'écrit :

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{2j\pi k f_0 t} \quad \text{avec} \quad c_k = \frac{1}{T_0}\int_{-T_0/2}^{T_0/2} x(t) e^{-2j\pi k f_0 t} dt$$

où **j** est le nombre complexe, $f_0 = 1/T_0$ est la fréquence fondamentale du signal, kf_0 sont les fréquences de ses harmoniques et **t** le temps. Les coefficients c_k sont des intégrales d'une fonction de la variable complexe, ils sont donc complexes. On ne représente généralement que leur *module* en fonction de la fréquence que l'on appelle *spectre* du signal.



Notons qu'avec l'écriture choisie (forme complexe) il apparait des fréquences positives et négatives. En effet :

$$cos(2\pi f_0 t) = \frac{e^{2j\pi f_0 t} + e^{-2j\pi f_0 t}}{2}$$

Par conséquent un signal co-sinusoïdal de fréquence f_0 sera représenté par deux fréquences f_0 et $-f_0$ dans le domaine fréquentiel, chacune avec une amplitude 1/2.

➡ *Le spectre comporte donc des fréquences positives et négatives. Les fréquences négatives n'ont pas de signification physique, elles découlent du calcul. Ces fréquences ne peuvent être ignorées, elles portent autant d'énergie que les fréquences positives.*

I.2. Convention d'écriture :

On note en minuscule les signaux temporels et en majuscule les signaux fréquentiels (transformés). Les signaux continus sont notés comme étant des fonctions du temps **x(t)** ou de la fréquence **X(f)**. Les signaux échantillonnés sont notés en indiquant leur numéro d'échantillon entre crochets droits : **x[k]** pour un signal temporel, **X[k]** pour un signal transformé.

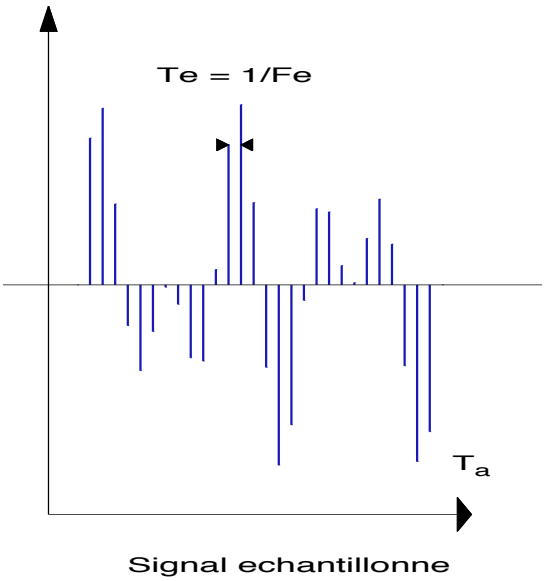
I.3. Représentation graphique d'un spectre.

On représente le spectre d'un signal échantillonné sous la forme d'un bargraphe (**bar** ou **stem**) du module de X[f].

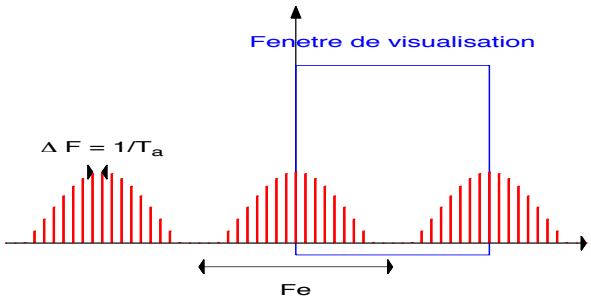
➡ *Physiquement un signal temporel est toujours borné en fréquence : il admet une fréquence maximale.*

Si l'on note T_a la durée de l'acquisition d'un signal, F_e la fréquence de répétition des acquisitions aussi appelée "fréquence d'échantillonnage", alors le spectre d'un signal échantillonné a une forme ressemblant à celle ci-dessous.

➡ *Notez que le spectre d'un signal échantillonné est périodique dans le domaine des fréquences.*



Signal echantillonne



Spectre d'un signal echantillonne

Le nombre de points **N** (*N = Fe*Ta*) est le même dans le domaine temporel et dans le domaine fréquentiel : **X[f]** a autant de points que **x[t]**. Les points de **X[f]** sont compris entre $-F_e/2$ et $F_e/2$. Toutefois, ces points sont numérotés à partir de *1* en Matlab, par conséquent si on trace le spectre par ordre de points croissants on observe les raies dans la "fenêtre de visualisation" indiquée dans le schéma ci-dessus. C'est à dire les fréquences positives de la période du spectre centré en autours de la fréquence 0 et les fréquences négatives de la période du spectre centrée autours de F_e.

La périodicité du spectre provient de l'utilisation d'un peigne de Dirac pour échantillonner mathématiquement le signal continu :

$$x[k] = \sum_{k=0}^{n-1} x(t)\delta(t - k * T_e)$$

Par conséquent **X[k]** est une somme de fonctions retardées de $k/T_e = k*F_e$. (Reportez-vous au cours de mathématiques)

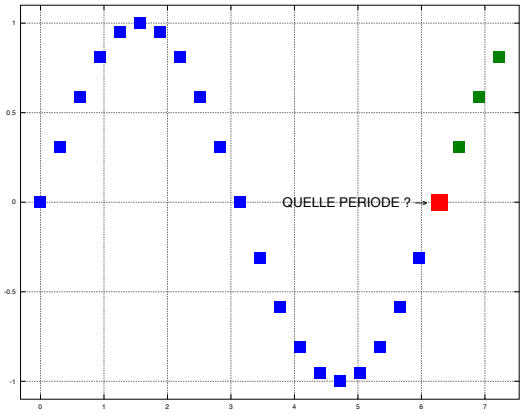
I.4. Notion de Période pour un signal échantillonné.

Un signal **x(t)** est de période **T** si l'on a **x(t+T)** = **x(t)** et ce, quel que soit **t**.

On dit qu'un signal échantillonné **x[k]** est de période **N** si l'on a **x[k+N]** = **x[k]** quel que soit **k**. Par conséquent, une période comporte **N** points. Si l'on prend **k** = 0, alors la première période comprend les **N** premiers points, soient les points **x[0]** à **x[N-1]** dans une numérotation débutant à zéro.

➡ *La période d'un signal échantillonné est forcément un nombre de points !*

Ceci est illustré par la figure ci-dessous.



Le point rouge est-il dans la période bleue ou la période verte ?

➡ *Lorsque l'on échantillonne une période d'un signal sinusoïdal, il ne faut donc pas inclure dans l'échantillon le point correspondant à l'angle 2π.*

II. Expressions de la Transformée Discrète

Les expressions suivantes permettent le calcul de la Transformée de Fourier Discrète. Dans celles-ci, **x[t]** est la représentation temporelle du signal, **X[f]** sa représentation fréquentielle. Dans les deux cas, le signal comporte **N** points **[0, N-1]**.

II.1. Transformée directe :

$$X[f] = \frac{1}{N}\sum_{t=0}^{N-1} x[t] * exp\left(-2 * i * \pi * \frac{f * t}{N}\right) \qquad f \in [0, N-1]$$

II.2. Transformée inverse :

$$x[t] = \sum_{f=0}^{N-1} X[f] * exp\left(+2 * i * \pi * \frac{t * f}{N}\right) \qquad t \in [0, N-1]$$

III. TP Fourier : signal sinusoïdal

III.1. Transformée de Fourier Directe

Exercice IV-1

- Écrivez une fonction qui calcule la TFD d'un signal quelconque. La fonction devra avoir l'interface suivante :

```
function X = fourierD( x )
```

La fonction utilisera, en première approche, des boucles **for**.

- Tracez le spectre de **S[f]**, transformée de Fourier du signal **s[t]** = sin(t) échantillonnée sur une période de 63 points.
- Observez l'amplitude des raies du spectre et expliquez la position des différentes raies.
- Rajoutez une composante continue à **s[t]** puis tracez le spectre de **S[f]**. Expliquez les différentes raies observées.
- Modifiez la fonction pour qu'elle utilise la fonction **sum** à la place de la boucle **for** interne.
- À quoi correspond la boucle interne de la Transformée de Fourier Discrète sur le plan mathématique ? Proposez en conséquence une autre écriture de cette boucle.

III.2. Transformée de Fourier Inverse

Exercice IV-2

- Modifiez la fonction fourierD pour qu'elle calcule également la transformée de Fourier inverse.Votre fonction modifiée aura un paramètre supplémentaire **a** indiquant le sens de transformation :

```
function Z = fourier( a , s )
% si a = -1, calcule la transformée directe
% si a =  1, calcule la transformée inverse
```

- Dans un script, calculez le signal **s** = sin(t). Calculez **S** la transformée de **s**.
- Calculez **x** la transformée inverse de **S** à l'aide de la fonction fourier. Tracez **x**.
- Comparez-le au signal initial.

➡ *Attention à la fonction **plot** lorsque la variable tracée est complexe...*

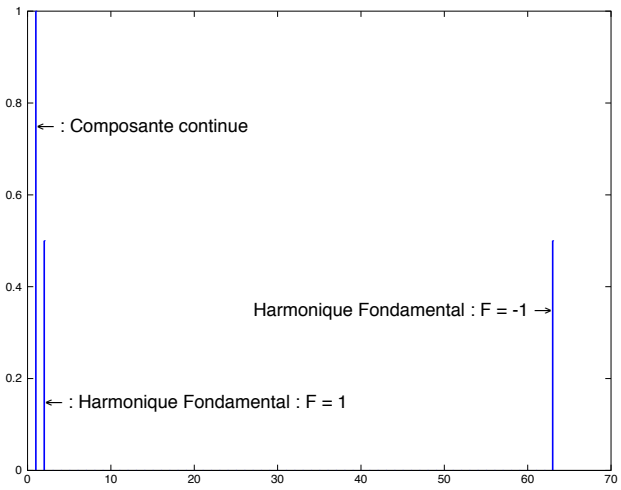
V - Etude de la Transformée de Fourier de signaux simples

I. Signaux "sinusoïdaux"

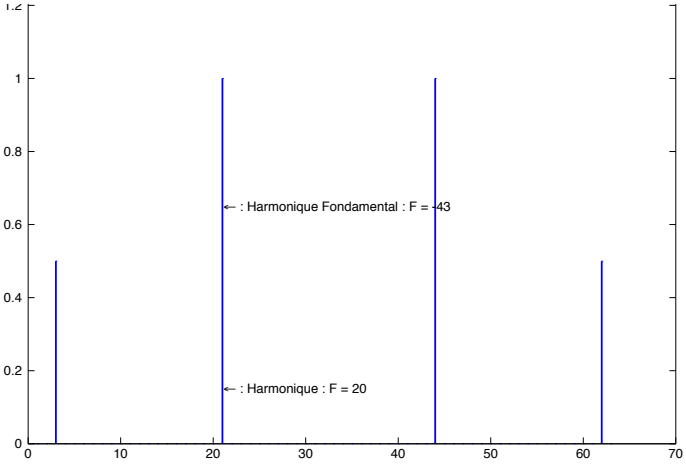
Exercice V-1

- Tracer le spectre **X[f]** d'une période de **x[t]** = sin(**teta**) échantillonnée sur 63 points à l'aide de la fonction fourier. Pour ce faire, vous créerez d'abord un vecteur **teta**.
- Ajoutez une composante continue à **x[t]** puis tracez le spectre de ce nouveau signal.

La première raie (Spectre ci-dessous) est la valeur moyenne du signal temporel. L'amplitude est correcte car cette raie est à la fois en fréquence négative et positive.



- Copiez les 32 premiers points de **X[f]** dans un vecteur **Y** puis calculez la transformée inverse de ce nouveau signal fréquentiel. Tracez ce signal temporel en fonction du numéro du point (indice). Que remarquez-vous au niveau des amplitudes ?
- Tracer le spectre de **x[t]** = sin(2***teta**). Que constatez-vous pour les raies du spectre **X** ?
- idem pour **x[t]** = sin(2***teta**) + sin (5***teta**) + sin(15***teta**). Même question que ci-dessus.
- idem pour : **x[t]** = sin(2***teta**) + sin(20***teta**) – sin(43***teta**) puis pour : **x[t]** = sin(2***teta**) + sin(20***teta**) + sin(43***teta**). Même question que ci-dessus. Donnez les fréquences des harmoniques du signal à partir du spectre, correspondent-elles au signal calculé ? Que pensez-vous des amplitudes des harmoniques ?



II. Signaux "carrés"

Exercice V-2

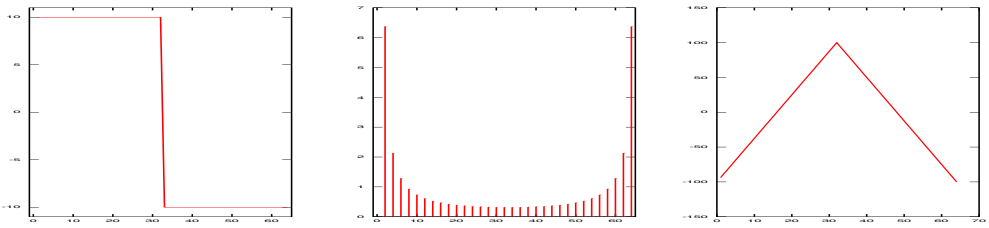
- Écrivez une fonction qui calculera les points d'une période de **N** points d'un signal carré de rapport cyclique sera égal à **tau**. **Vmin** est la valeur du signal à l'état bas, **Vmax** la valeur à l'état haut. L'interface de la fonction sera :

  function S = carre(Vmin, Vmax, N, tau)

- En utilisant la fonction carre générez, dans un script, un signal carré de rapport cyclique 0,5 et d'amplitude -10 / +10 échantillonné sur 64 points. Tracez ce signal et son spectre.
- Dans un script, calculez la Transformée de Fourier **S** d'un signal carré **s**, de rapport cyclique 0,5 et d'amplitude -10 /+10. Calculez, alors, le signal **S2** = S^2. Tracez le spectre de **S2**, puis calculez **s2** la Transformée de Fourier Inverse de **S2**. Tracez **s2**.

➡ *Le produit dans le domaine fréquentiel correspond à la convolution dans le domaine temporel.*

- Expliquez la forme du signal **s2**.



III. Utilisation de la Transformée de Fourier incluse dans Octave

I.La fonction fft de Matlab/Octave.

La Transformée de Fourier incluse dans Octave a pour syntaxe : **Z = fft(z)** où **Z** est un vecteur de nombres complexes et **z** le signal en entrée.

Exercice V-3

- Calculez une période d'un signal **s**, sinusoïdal, d'amplitude crête unité échantillonné sur 64 points par période. Observez l'amplitude des raies du spectre **S** de **s** et expliquez la position des raies. Que constatez-vous ?
- Ajoutez une composante continue au signal **s** et tracez le nouveau spectre. Que constatez-vous ?
- Divisez chaque harmonique de **S** par le nombre de points du signal temporel. Tracez le spectre de ce nouveau signal. Que constatez-vous ?

Cette opération consiste à normaliser le spectre.

➡ *À partir de maintenant toutes vos transformées seront normalisées.*

II. Conservation de l'énergie

La Transformée de Fourier conserve l'énergie. C'est ce qu'exprime l'identité de Parseval pour un signal échantillonné :

$$\frac{1}{N}\sum_{t=0}^{N-1} x\left[t\right]^2 = \sum_{f=0}^{N-1} \left|X\left[f\right]\right|^2 \qquad (I)$$

Il y a autant d'énergie dans la représentation fréquentielle d'un signal que dans sa représentation temporelle.

C'est ce que l'on peut vérifier aisément en choisissant la bonne expression de la transformée et un échantillon contenant un nombre entier de périodes d'un signal sinusoïdal : la somme des harmoniques positives et négatives pour une même fréquence vaut la valeur crête des signaux à cette même fréquence.

Toutefois, si le signal n'est pas sinusoïdal au départ, on ne peut pas faire cette comparaison simplement. On peut quand même calculer l'énergie totale des deux représentations du signal et les comparer.

Exercice V-4

- Échantillonnez un signal carré **s** de rapport cyclique 1/3, d'amplitude **smax=10** et **smin=-5** d'une longueur de 120 échantillons.
- Tracez le spectre **S** de **s** en utilisant la fonction **fft** d'Octave.
- Vérifiez l'expression (I).

III. Nombres de points transformés.

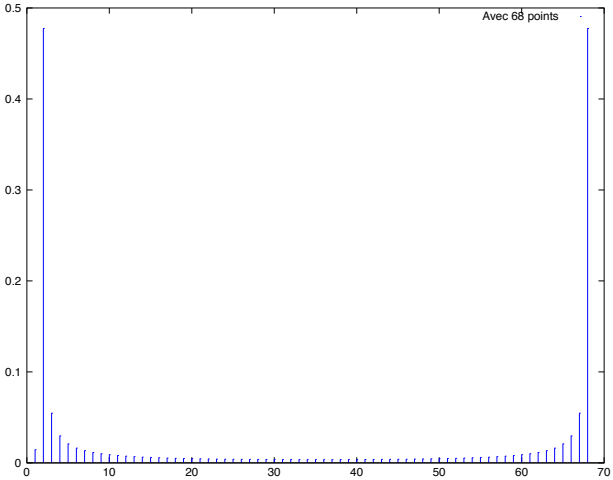
La syntaxe de la commande **fft** est multiple : on peut préciser le nombre de points à transformer (tapez help **fft**), l'appel de la commande devient : **Z** = **fft**(**z**, **n**) où **n** est le nombre de points pris dans le signal **z** pour calculer **Z** la Transformée de Fourier de **z**(1:**N**).

Exercice V-5

- Générez un vecteur **t** régulièrement espacé sur 70 points avec un pas **h**=2*π/63.
- Calculez un signal temporel **s[t]** = sin(**t**). Tracez **s**.
- Tracez le spectre de **S** la représentation fréquentielle de **s** en utilisant la fonction **fft**. Que constatez-vous ?

Pour parer ce problème, il faut ramener le nombre de points à N la période numérique de s. On se propose de la faire point à point en demandant confirmation à l'utilisateur.

- En utilisant la fonction **fft**, tronquez dans une boucle le signal **s** de **k** points avant le calcul de **S**. Tracez alors le spectre de **S** en indiquant en légende le nombre de points de **s** transformés. A chaque itération, il sera demandé à l'utilisateur s'il veut continuer ou non.
- A quelle valeur de **k** le spectre devient-il clair (sans raies parasites) ? Concluez.



Exercice V-6

Le but de cet exercice est de "voir" l'étendue spectrale d'un signal et l'importance des différents harmoniques.

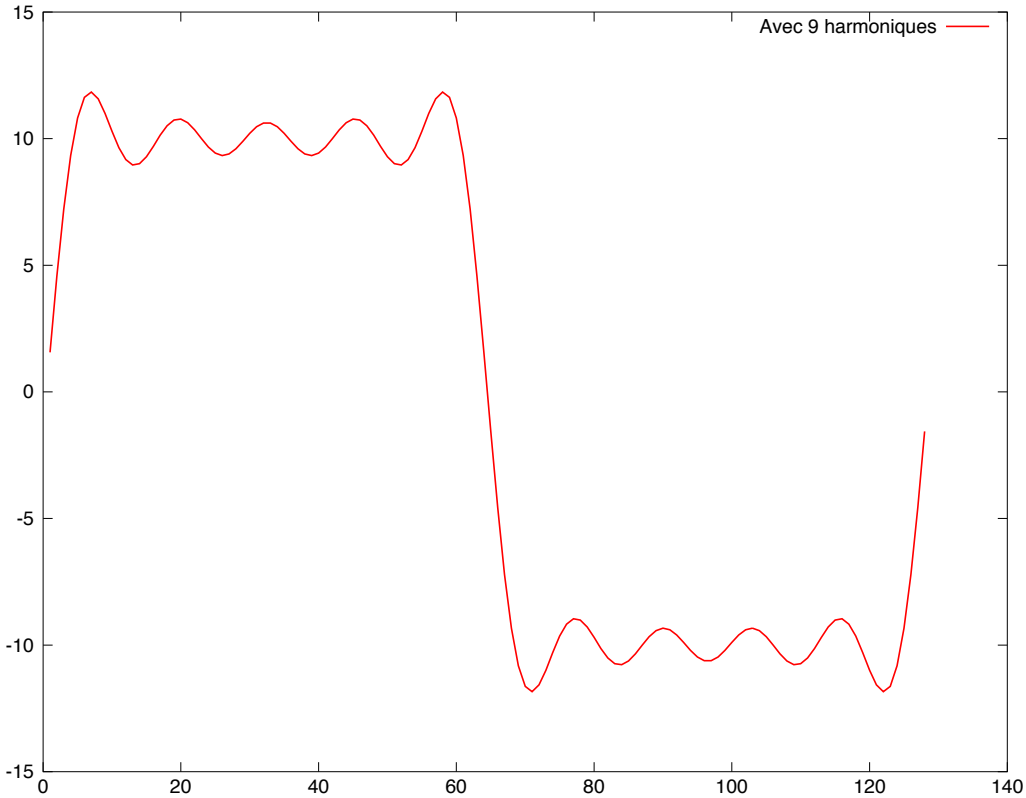
- En utilisant l'aide en ligne, donnez la syntaxe de la Transformée Inverse incluse dans Octave/Matlab.
- Dans un script, générez un signal carré **x** d'amplitude +/- 10 comportant 128 points avec un rapport cyclique de 0,5 en utilisant la fonction carre écrite précédemment.
- Tracez le spectre de **x**.
- Isolez dans un vecteur **R** l'harmonique fondamental de **X**. Calculez **r** la transformée inverse de **R**, tracez-le.
- Isolez dans un vecteur **R** les **k** premiers harmoniques de **X**. Calculez la transformée inverse de **R**. Tracez le signal temporel obtenu.

Indications :

➡ *on peut isoler* **k** *harmoniques à l'aide d'un masque ou d'une sélection ;*

➡ *n'oubliez pas de pensez aux fréquences négatives.*

- Observez la forme du signal temporel obtenu à partir de la transformée inverse d'un nombre croissant d'harmoniques, le script tracera avec **1** puis **2** harmoniques etc. Utilisez la commande **pause** entre les tracés.
- Que pensez-vous de l'étendue spectrale des signaux carrés ?
- Que constatez-vous pour l'amplitude maximale ?



VI - Résolution Fréquentielle du Spectre

I. Introduction

On calcule la transformée de Fourier des signaux échantillonnés à l'aide de l'expression :

$$X[f] = \frac{1}{N}\sum_{t=0}^{N-1} x[t] * exp\left(-2 * i * \pi * \frac{f * t}{N}\right) \qquad f \in [0, N-1]$$

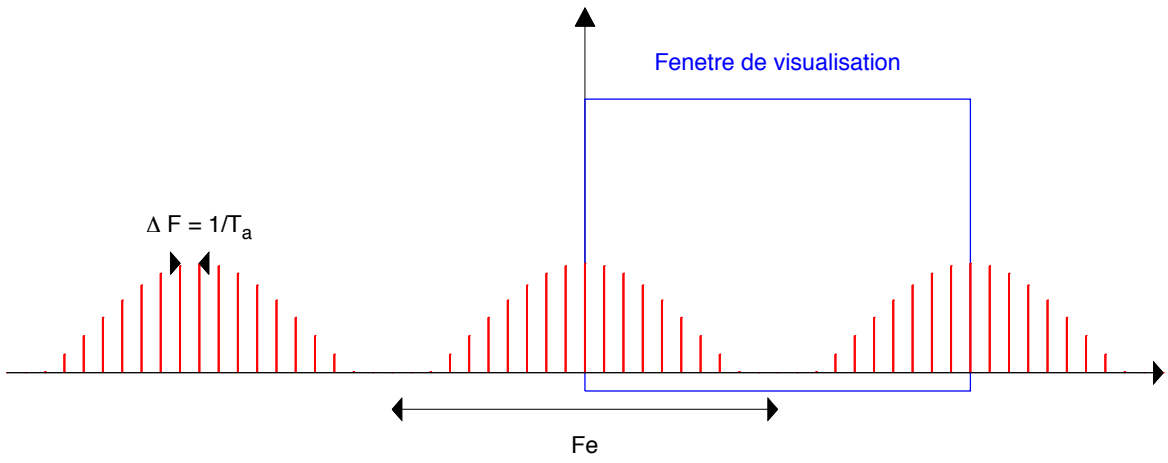
La transformée est échantillonnée et le spectre est période de période **F_e** où **F_e** est la fréquence d'échantillonnage, c'est à dire le nombre d'échantillons mesurés en une seconde.

On peut définir **T_a** la durée d'acquisition du signal. Il est clair que cette durée est limitée, et que par conséquent le nombre de points **N** l'est aussi.

On a **$N = F_e * T_a$**.

Il est alors possible de définir un "vecteur-temps" qui contient tous les temps auxquels on procède à l'échantillonnage (c'est à dire à sa mesure) du signal continu échantillonné : le premier temps est 0, le pas est de **$Te = 1/F_e$** et l'on mesure jusqu'à **T_a-T_e** (sinon on aurait **F_e*T_a+1** points à cause du temps t=0).

Dans le domaine fréquentiel on retrouve ces **N** points. Mais avoir échantillonné le signal temporel rend le spectre **F_e**-périodique. Par conséquent, l'écart entre deux raies successives du spectre est de **$F_e/N = 1/T_a$**. C'est ce que fait ressortir le schéma ci-dessous.



Spectre d'un signal echantillonne

Exercice VI-1

- Générez un "vecteur-temps" **t** d'une durée **Ta** d'1/10 de seconde, échantillonné à la fréquence **Fe** d'1 kHz.
- Calculez puis tracez à partir de **t** les valeurs que prend un signal sinusoïdal **s30** de fréquence f=30Hz aux instants échantillonnés.

➡ *On obtient ainsi le même signal qu'obtiendrait un Convertisseur Analogique/Numérique cadencé à 1kHz en 1/10 de seconde (le bruit en moins !).*

- Même question pour un signal **s40** de fréquence f=40Hz.
- Tracez les spectres de **S30** et de **S40** de ces deux signaux en fonction de la fréquence.
- En changeant le domaine de tracé à l'aide de la commande **axis**, comparez le rang des raies qui apparaissent dans les deux spectres.

- En utilisant les remarques faites ci-dessus, et la commande **axis**, tracez le spectre en fonction de la fréquence en Hz.

II. Signaux non périodiques : influence sur le spectre

II.1. Introduction

En réalité, on n'acquiert jamais des signaux ne contenant que des multiples entiers de la fréquence minimale. Pour vous convaincre que cela pose problème…

Exercice VI-2

- Reprenez le même vecteur-temps que précédemment (durée d'1/10s échantillonnée à 1kHz), et calculez à partir de ce vecteur-temps les valeurs que prend un signal sinusoïdal de fréquence 10Hz aux instants échantillonnés.
- Superposez à ce signal un signal de fréquence 5Hz.
- Calculez la transformée de Fourier de ce signal en utilisant la fonction **fft**.
- Tracez le spectre en fonction de la fréquence en Hz. Que constatez-vous ? Quelle solution proposez-vous ? Testez-là.

On pourrait penser que c'est parce que l'on a pris un signal dont la période "ne tient pas" dans la durée de l'échantillon, que pour les autres signaux, on n'aurait pas le même problème. Testez l'exemple suivant :

Exercice VI-3

- Dans les mêmes conditions que précédemment (durée d'1/10s échantillonnée à 1kHz) échantillonnez un signal **s** composé de la somme de deux sinusoïdes l'une de fréquence 10Hz l'autre de fréquence 25Hz.
- Tracez le spectre **S**. Que constatez-vous ?

Avec le "vecteur-temps" choisi, la résolution (= $1/T_a$) du spectre est de 10Hz. Il n'est pas possible de représenter un signal de fréquence 25Hz avec un telle résolution, puisque les harmoniques sont des multiples entiers de la résolution qui est aussi la fréquence la plus basse que l'on peut représenter.

Dans le cas où toutes les fréquences présentes dans le signal d'entrée sont des multiples entiers de la fréquence minimale, on parle de TFD Synchrone (Asynchrone sinon).

III. Méthodes utilisées pour corriger le problème de la Transformée Asynchrone

III.1. Introduction

Si le signal est asynchrone –c'est à dire qu'il ne contient pas uniquement des fréquences multiples de $1/T_a$, (T_a étant la durée d'acquisition)–, le spectre peut devenir complètement illisible. Le problème vient du fait que l'on ne "voit" le signal que pendant une courte "*fenêtre*" temporelle. En fait, c'est la troncature brutale du signal acquis qui pose problème, autrement dit on a affaire à un "effet de bord".

Il existe des techniques pour le spectre plus lisible. Elles sont basées sur la modification de la *fenêtre* de visualisation du signal.

III.2. Fenêtrage

L'idée est la suivante : puisque le fait d'avoir un signal asynchrone pose problème, il suffit de rendre ce signal synchrone (= avoir une période du signal pendant la durée de l'acquisition) pour lever la difficulté !

Ceci peut être réalisé en pondérant le signal par un ensemble de coefficients dont la valeur est comprise entre 0 et 1 et tels que leurs valeur soient nulles en 0 et en T_a.

Il y a autant de coefficients que de points dans le signal et cet ensemble de coefficient peut lui aussi être considéré comme un signal.

Ces signaux, de période T_a, sont appelés des *fenêtres*.

III.3. Transformée de Fourier à Temps Courts (STFT)

Une autre méthode consiste à diminuer le nombre de points de l'échantillon jusqu'à arriver sur un échantillon de longueur telle qu'il soit un multiple d'une des fréquences contenue dans le signal. Cette méthode est dite "Short-time Fourier Transform", vous l'avez déjà mise en pratique (exercice V-5).

Bien entendu, si la fréquence contenue dans le signal analysé n'est pas un sous multiple de la fréquence d'échantillonnage, cette approche est inopérante.

On combine en général les deux méthodes.

III.4. Exemples de Fenêtres

III.4.1. Fenêtre triangle (Bartlett)

Il s'agit d'un signal triangulaire de rapport cyclique 0,5 et d'amplitude unité entre 0 et 1.

III.4.2. Fenêtre de Welch

Il s'agit du signal temporel défini par : $1 - \{ [i - 0,5*(N-1)] / [0,5 * (N+1)] \}2$

où i est l'indice du point, N le numéro du dernier point dans une numérotation débutant à zéro. Il s'agit d'arcs de parabole.

III.4.3. Fenêtre de Hanning

Il s'agit du signal temporel défini par : $0,5 * (1 - \cos(2*\pi*t/T_a))$

III.4.4. Fenêtre de Hamming

Il s'agit du signal temporel défini par : $0,54 - 0,46 * \cos(2*\pi*t/T_a)$

III.4.5. Fenêtre de Blackman

Il s'agit du signal temporel défini par :

Blackman :

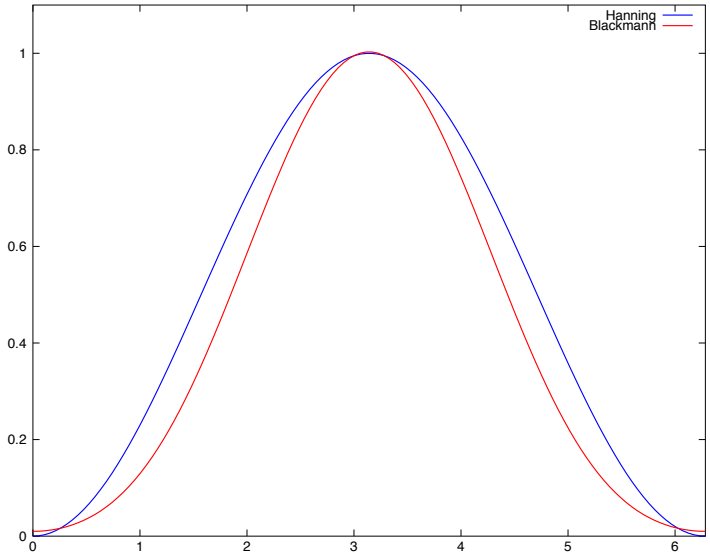
$0,42659071 - 0,49656062*\cos(2*\pi*t/T_a) + 0,07684867* \cos(4*\pi*t/T_a)$

Blackman-Harris :

$0,42323 - 0,49755*\cos(2*\pi*t/T_a) + 0,07922* \cos(4*\pi*t/T_a)$

➡ *Par rapport aux fenêtres précédentes on rend le signal plus "raide" sur les bords.*

➡ *Les coefficients ci-dessus résultent d'un processus d'optimisation et leurs valeurs évoluent suivant le critère retenu. On se contente en général de 2/3 décimales.*



Exercice VI-4

1. Échantillonnez un signal **s** composé de deux signaux sinusoïdaux de fréquences 50 et 625 Hz pendant 100ms à la fréquence de 5 kHz,
2. Tracez le signal **s**.
3. Tracez le spectre **S** en précisant en abscisse la fréquence des harmoniques.
4. Générez un signal **h** correspondant à une fenêtre de Hanning :
   0,5 * (1 - cos(2*π*t/Ta)),
5. Calculez le signal **sh** = **s***h. Tracez le spectre **SH** de ce nouveau signal. Comparez avec le spectre **S**, N'affichez que les fréquences entre 0 et 700 Hz.
6. Recommencez les points 4 et 5 avec la fenêtre de Blackman-Harris :
   0,42323 - 0,49755*cos(2*π*t/**Ta**) + 0,07922* cos(4*π*t/**Ta**).
7. Reprenez les points 1 à 5 avec les fréquences 50 et 627, puis 50 et 622 Hz, n'affichez que les fréquences entre 0 et 700Hz.
8. Reprenez les points 1 à 5 en utilisant une fenêtre triangulaire : amplitude entre 0 et 1, rapport cyclique 0,5.

III.2. Conséquences du fenêtrage

Il est bien sûr beaucoup plus compliqué dans ces conditions (TFD asynchrone + fenêtrage) de vérifier l'égalité de Parseval. Il faut donc regarder avec précaution les spectres donnés par des appareils automatiques (oscilloscopes, analyseurs...). Très souvent, ces appareils effectuent un fenêtrage de Hamming (ou de Hanning), et il n'est pas réaliste de parler de l'amplitude des raies sans avoir lu de façon approfondie la documentation de l'appareil à moins de ne se contenter que d'une mesure qualitative, de plus le spectre est souvent représenté en décibels.

III.3. Bibliographie

Vous trouverez d'autres fenêtres et des explications dans l'article de Frederic J. Harris "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform, Proceedings of the IEEE", Vol. 66, No. 1, January 1978. -allez voir sur wikipédia avec comme mot clé : window function.

De même il est possible d'installer la toolbox traitement du signal dont le contenu est décrit ici : http://octave.sourceforge.net/doc/signal.html

VII - Présentation succincte des filtres RII

I. Présentation

➡ *Ce TD est très long. Tout le TD ne sera pas traité en séance de TD. Préparez-le, et finissez-le chez vous.*

L'objectif de ce TD est une présentation des filtres numériques et une étude du comportement temporel des circuits du premier et du second ordre passe bas. Revoir le cours d'OEE est une bonne idée.

II. Présentation succincte des filtres numériques

II.1. Les relations de récurrence

Par filtre numérique, nous entendons système linéaire invariant. Ces systèmes comportent une entrée et une sortie. La relation liant la sortie y à l'entrée x constitue la fonction de transfert du filtre. Cette relation peut s'écrire sous la forme d'une équation aux différences dans le cas des signaux numériques :

$$y_k = \sum_{i=0}^{n} a_i x_{k-i} - \sum_{j=1}^{n} b_j y_{k-j} \qquad (I)$$

Cette équation est le résultat de la convolution des fonctions de transfert de l'entrée et de la sortie du filtre.

Si le système fait intervenir la sortie (=si les coefs **bj** ne sont pas tous nuls) on a affaire à un filtre récursif, dans le cas inverse, le filtre est dit non récursif. Il est possible d'échantillonner la réponse impulsionnelle H et d'utiliser la convolution numérique pour calculer la sortie :

$$y_k = \sum_{i=0}^{n} H_i x_{k-i} \qquad (II)$$

Si la réponse impulsionnelle est finie, il sera possible d'utiliser la relation (II) sinon on utilisera la relation (I). Dans le premier cas, on parlera de filtre à Réponse Impulsionnelle Finie (RIF), dans le second de filtre à Réponse Impulsionnelle Infinie (RII).

II.2. Détermination des relations de récurrence

Il existe plusieurs méthodes pour transposer dans le domaine numérique (discret) des filtres analogiques dont on connaît la fonction de transfert H(p). Une méthode dite d'Euler (ou d'équivalence de la dérivation) consiste à remplacer dans l'expression de H(p) la variable complexe p par :

$$p \rightarrow \frac{1 - z^{-1}}{T_e} \qquad (III)$$

La méthode homographique ou d'équivalence de l'intégration propose de remplacer **p** par :

$$p \rightarrow \frac{2\left(1 - z^{-1}\right)}{T_e\left(1 + z^{-1}\right)} \qquad (IV)$$

Notez :

➡ **T_e** *est la période d'échantillonnage,* z^{-1} *est l'opérateur retard unité (* z^{-1}**x[k]** → **x[k-1]** *).*

➡ *On écrit donc que le signal de sortie composé des échantillons* **y[k]** *vérifie la relation : Sortie/Entrée = H(p), soit si* **x[k]** *est un échantillon du signal d'entrée :* **y[k]** *= H(***z***).***x[k]***.*

III. Application au filtre passe-bas du premier ordre

Exercice VII-1

- Déterminez la relation de récurrence d'un filtre passe-bas du premier ordre en utilisant l'équivalence de la dérivation.

On rappelle que la fonction de transfert d'un premier ordre est la suivante :

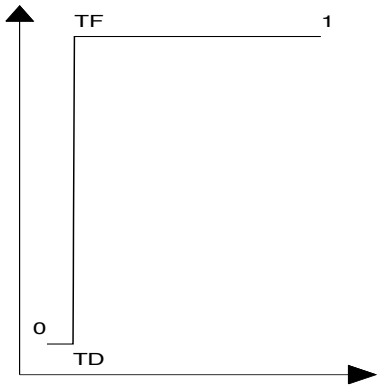
$$H(p) = \frac{1}{1+\tau p}$$

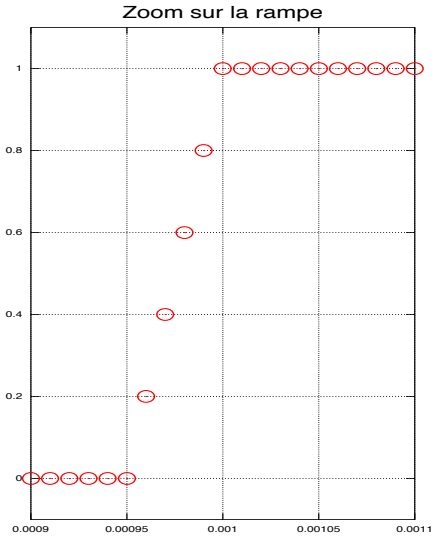
où **τ** = **RC**. Vous prendrez **R**=1kΩ et **C** = 1 µF.

III.1. Réponse impulsionnelle

Exercice VII-2

- Écrivez une fonction appelée **pulse** qui génère un signal impulsion tel que représenté ci dessous.





Vous prendrez **TD** = 9,5% et **TF** = 10% de la durée de l'échantillon. Tracez ce signal à titre de vérification. L'en-tête de la fonction sera :

```
function s = pulse( TD, TF,  Ta, Fe)
% TD et TF vaudront 9.5 et 10 respectivement.
```

- Générez, à l'aide de la fonction pulse, une impulsion d'une durée totale de 10 ms échantillonnée à une fréquence de 100kHz.
- Appliquez le filtre passe bas à l'impulsion et tracez le résultat. Avec **R** = 1kΩ et **C** = 1µF.
- Recommencez pour plusieurs valeurs de **C**.

III.2. Réponse harmonique

Exercice VII-3

- Échantillonnez un signal sinusoïdal de fréquence 100Hz pendant 20ms à 20kHz.
- Appliquez à ce signal le filtre RC avec **R** = 1kΩ et **C** = 1µF.
- Superposez le signal d'entrée et de sortie sur la même courbe. Conclusion ?
- Tracez les spectres en fréquence du signal d'entrée et de sortie. Comparez les amplitudes. Quelle est la fréquence de coupure du filtre RC ?

Exercice VII-4

- Échantillonnez un signal composé de deux signaux sinusoïdaux de fréquence respectives 100Hz et 3kHz pendant 25ms à 20kHz.
- Appliquez à ce signal le filtre RC avec **R** = 1kΩ et **C** = 1µF.
- Superposez le signal d'entrée et de sortie sur la même courbe. Conclusion ?
- Tracez les spectres en fréquence du signal d'entrée et de sortie. Comparez les amplitudes. Quelle est la fréquence de coupure du filtre RC ?

Exercice VII-5

- Échantillonnez un signal composé de deux signaux sinusoïdaux de fréquence respectives 100Hz et 3kHz pendant 25ms à 20kHz.
- Appliquez à ce signal le filtre RC avec **R** = 1kΩ et **C** = 1µF.
- Calculez l'affaiblissement en dB en comparant les amplitudes des dernières périodes des signaux d'entrée et de sortie.
- Tracez le diagramme de Bode de ce filtre dans la gamme de fréquence 10Hz- 10kHz

➡ *indication : attention à la durée du signal !*

- Comparez la courbe obtenue à la courbe théorique d'un premier ordre. Que constatez-vous aux hautes fréquences ?
- Recommencez avec **Fe** = 25kHz. Concluez.

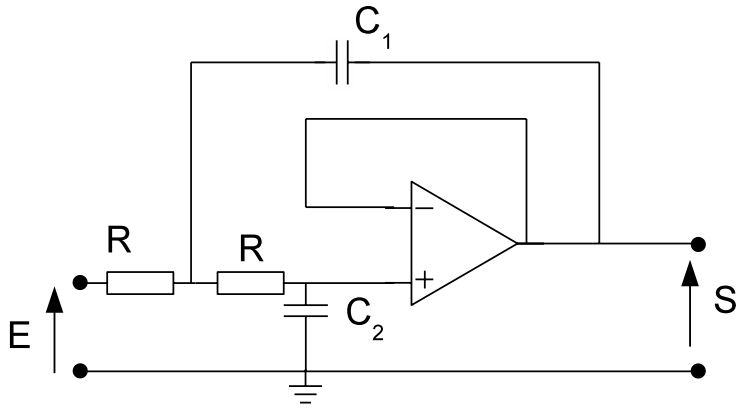
IV. Filtre passe-bas du second ordre

Exercice VII-6

- Déterminez la relation de récurrence, en utilisant l'équivalence de la dérivation, d'un filtre passe-bas du second ordre de la forme générale :

$$H(p) = \frac{1}{1 + ap + bp^2}$$

Le circuit que vous étudierez est le suivant :



Sa fonction de transfert est :

$$H(p) = \frac{1}{1 + 2RC_2p + R^2C_1C_2p^2}$$

IV.1. Réponse impulsionnelle

Exercice VII-7

- Tracez la réponse impulsionnelle de ce filtre. Vous prendrez **R** = 1kΩ, **C1** = 3,6µF et **C2** = 36nF. La durée de l'impulsion sera 10ms et le signal sera échantillonné à 50kHz.

IV.2. Réponse Harmonique

Exercice VII-8

- Appliquez à ce filtre des signaux sinusoïdaux de fréquences : 100Hz, 439Hz, 1kHz, 3kHz. Tracez sur un même graphe les signaux d'entrée et de sortie pour chaque fréquence. Vous choisirez une fréquence et une durée d'échantillonnage pour tracer au moins trois périodes de signal.
- Tracez dans deux fenêtres distinctes un signal composé de la somme de deux sinus de fréquences 100Hz et 3kHz et la réponse du circuit à ce signal.
- Calculez les valeurs de **C1** et **C2** pour que l'on ait **ωm** = 444Hz et **ξ** égal à 0,7071. En utilisant les valeurs calculées, tracez la réponse du nouveau filtre à l'impulsion et aux signaux sinusoïdaux des fréquences données au point 1 de cet exercice.

On rappelle que pour un système du second ordre on a :

$$H(p) = \frac{1}{1 + \frac{2\xi p}{\omega_m} + \frac{p^2}{\omega_m^2}}$$

- De la même manière que pour le premier ordre, calculez l'affaiblissement du filtre sur le signal d'entrée et comparez avec l'affaiblissement théorique pour un signal de fréquence 3kHz.

Exercice VII-9

- Tracez le diagramme de Bode de ce filtre dans la gamme de fréquence allant de 10Hz à 10kHz. La durée du signal sera adaptée pour que l'on ait pour chaque fréquence 3 périodes de signal.
- Même question pour le cas où **C1** = 3,6µF et **C2** = 36nF, on prendra en ce cas une durée de 10 périodes. La mesure s'effectuant toujours sur la dernière période.

Exercice VII-10

- Reprenez les mêmes études (premier et second ordre) avec l'équivalence de l'intégration (relation IV).

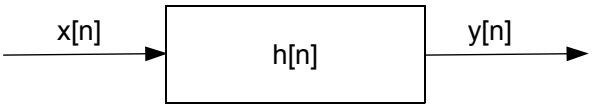
VIII - Débruitage

I. Objectifs

Dans cette séance, on cherche d'abord à estimer la réponse impulsionnelle d'un filtre en utilisant une méthode s'appuyant sur la corrélation des signaux. On utilise ensuite cette technique d'estimation pour débruiter des signaux.

II. Identification de la réponse impulsionnelle d'un filtre

On considère le système linéaire et invariant suivant :



L'objectif est d'estimer la réponse impulsionnelle de ce système, **h[n]**. Pour ce faire, on utilise une méthode qui consiste à injecter un signal aléatoire en entrée du système et à mesurer sa sortie **y[n]**. Comme on l'a vu en cours et TD de signaux aléatoires, on peut écrire :

$$R_{xy}[n] = R_x[n] * h[n]$$

Dans l'équation ci-dessus, $R_{xy}[n] = E\{ y[k+n]\, x[k] \}$ est l'inter-corrélation sortie-entrée et $R_x[n] = E\{ x[k+n]\, x[k] \}$ est l'auto-corrélation du signal d'entrée, **h[n]** est la réponse impulsionnelle du système, et le symbole * désigne l'opérateur de convolution discrète. En prenant la transformée de Fourier de l'équation ci-dessus, on obtient : $TF\{ R_{xy}[n] \} = TF\{ R_x[n] \}.TF\{ h[n] \}$

Ce calcul nous suggère la démarche suivante pour estimer la réponse impusionnelle d'un système linéaire invariant inconnu :

1. On injecte un signal **x[n]** en entrée du filtre et on mesure sa sortie **y[n]**.
2. A partir des signaux **x[n]** et **y[n]**, on estime l'auto-corrélation du signal d'entrée ($R_x[n]$) et l'inter-corrélation sortie-entrée ($R_{yx}[n]$).
3. On calcule la transformée de Fourier des corrélations ci-dessus.
4. On estime la transformée de Fourier de la réponse impulsionnelle du filtre (i.e. $H(\omega) = TF\{ h[n] \}$) en utilisant :
$$H(\omega) = \frac{TF\{ R_{xy}[n] \}}{TF\{ R_x[n] \}}$$
5. Question : quelle est la condition pour que l'estimation ci-dessus se passe sans problème ?
6. On estime la réponse impulsionnelle du filtre en calculant la transformée de Fourier inverse :
$$h[n] = TF^{-1}\{ H(\omega) \}$$

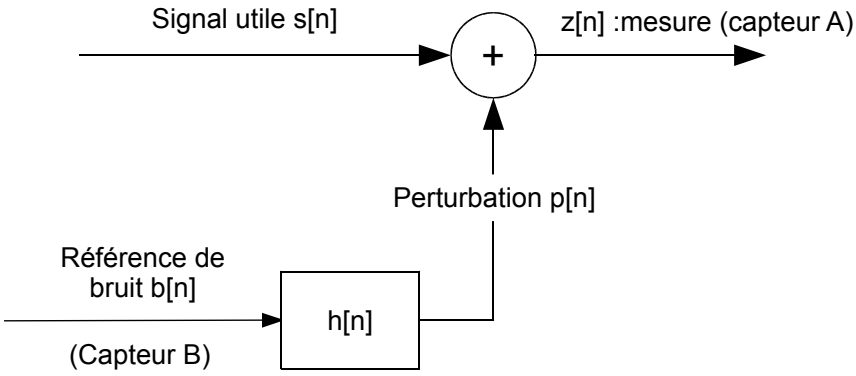
Exercice VIII-1

- Les fichiers "**x1.dat**" et "**y1.dat**" contiennent les échantillons des signaux **x[n]** et **y[n]**, l'entrée et la sortie d'un système linéaire et invariant.
- Mettez en œuvre la méthode exposée ci-dessus pour estimer la réponse impulsionnelle **h[n]** de ce système sur 50 échantillons.
  La fonction **xcorrel** vous permet de calculer l'inter-corrélation de deux signaux (voir l'annexe pour une description plus détaillée de cette fonction).
  Les fonctions **fft** et **ifft** permettent de calculer les transformées de Fourier directe et inverse d'un signal.

- Le fichier "**h1.dat**" contient la vraie réponse impulsionnelle du système ci-dessus. Comparez-la avec la réponse impulsionnelle estimée en traçant les deux courbes sur une même figure.

III. Débruitage d'un signal avec référence de bruit

On exploite, ici, la méthode précédente pour identifier la réponse impusionnelle d'un filtre dans une configuration de débruitage d'un signal avec référence de bruit telle que représentée ci-dessous.



On suppose disposer de deux capteurs : le premier mesure le signal bruité z[n] et le deuxième mesure la référence de bruit b[n].

Exercice VIII-2

Répondre aux questions suivantes, vues en TD.

- Quelle est l'expression de l'inter-corrélation $R_{zb}[n]$?
- Que devient cette équation si **b[n]** et **s[n]** sont non-corrélés et que **b[n]** est centré ?
- Comment s'écrit la relation ci-dessus dans le domaine de fréquence ?
- Comment peut-on estimer **h[n]** ?
- Comment déduire une estimation du signal débruité **s[n]** ?

La suite du TD consiste à appliquer la méthode décrite ci-dessus dans deux applications différentes.

IV. Application Audio

Le premier exemple est une application audio. On suppose que le signal utile **s[n]** est la voix d'une personne qu'on veut enregistrer avec un microphone «A» situé près de cette personne. En pratique, cet enregistrement est pollué par une perturbation, résultat de la propagation d'une source de bruit lointain et fort jusqu'au microphone «A». En modélisant la propagation de ce bruit par un effet de filtrage, nous pouvons utiliser la technique ci-dessus en plaçant un deuxième microphone «B» près de la source du bruit. On retrouve donc le modèle de la figure ci-dessus.

Exercice VIII-3

Le fichier "**z2.wav**" contient un signal sonore enregistré par le microphone «A». Le fichier "**b2.wav**" contient une mesure de ce bruit par le microphone «B».

- Ecoutez ces deux signaux.
- En utilisant la méthode expliquée ci-dessus, estimez la réponse impulsionnelle du filtre de propagation sur 300 échantillons et en déduire le signal utile. La fonction **filter** peut être utilisée pour effectuer la convolution (=filtrage RIF dans cet exemple).

➡ *Voir l'annexe pour une description plus détaillée de la fonction **filter**.*

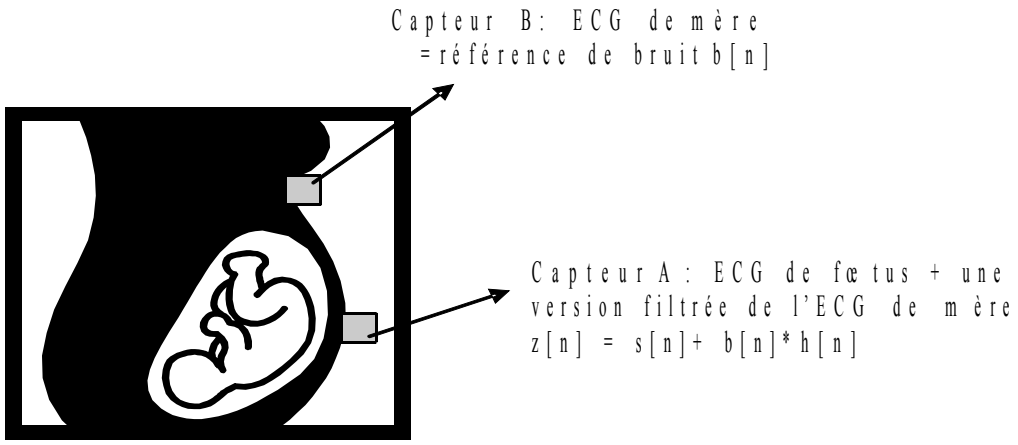
- Ecoutez le signal utile.

V. Application biomédicale

Le deuxième exemple est une application biomédicale qui concerne la mesure de l'électrocardiogramme d'un fœtus dans le ventre de sa mère.

Un capteur, situé sur le ventre de la mère donne le signal électrique du cœur du foetus (signal **s[n]**) additionné à un signal provenant du cœur de la mère mais qui a traversé les tissus abdominaux (perturbation **p[n]**). On modélise ce transfert abdominal par un système linéaire et invariant de réponse impulsionnelle **h[n]**.

Un autre capteur situé près du cœur de la mère donne le signal électrique du cœur de la mère (référence du bruit **b[n]**). Une fois estimée la réponse impulsionnelle **h[n]** il est aisé de débruiter le signal à partir de la référence de bruit.



Exercice VIII-4

Le fichier "**z3.dat**" contient le signal enregistré par le capteur «A» qui fournit l'ECG du fœtus noyé dans une version filtrée de l'ECG de sa mère.

Le fichier "**b3.dat**" contient un enregistrement de l'ECG de la mère (référence de bruit) par le capteur B.

- Tracez ces deux signaux.
- En utilisant la méthode expliquée précédemment, estimer la réponse impulsionnelle du filtre de propagation sur 50 échantillon et en déduire le signal utile.
- Comparer votre estimation avec le vrai ECG du fœtus, disponible dans le fichier "s3.dat" en traçant les deux signaux sur une même figure.

VI. Annexe: Fonctions Matlab xcorrel et filter

```
% function  R= xcorrel(y,x,p)
% Entrées : y et x sont deux vecteurs contenant les échantillons
% des signaux y[n] et x[n].
% Ces deux vecteurs doivent être de même longueur.
% p détermine le nombre de points pour lesquels l'inter-corrélation
% est calculée.
% Si P n'est pas précisé, par défaut p=(longueur de y)
% Sortie :   R est un vecteur contenant les 2*p-1
% (pour les indices -p+1 a p-1) premiers coefficients
% d'inter-corrélation.
```

```
% function  y=filter(b,a,x)
% Entrées :  b et a sont deux vecteurs contenant les coefficients
% d'un filtre RII, défini par l'équation ci-dessous:
% a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
%             - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
% Pour réaliser un filtre RIF, il suffit de prendre a(1)=1 et
% a(2)=...=a(na+1)=0.
% x est un vecteur contenant les échantillons du signal d'entrée x[n].
% Sortie :   y est un vecteur contenant les échantillons du signal
% de sortie y[n].
```