

INSTITUT NATIONAL DES SCIENCES  
APPLIQUEES DE TOULOUSE

3<sup>ème</sup> Année FISA

---

**MISE A NIVEAU SIGNAL**

**ENONCE DE TP - SERIE DE  
FOURIER ET CALCUL DE TAUX DE  
DISTORSION HARMONIQUE**

Alexandre Boyer  
alexandre.boyer@insa-toulouse.fr  
[www.alexandre-boyer.fr](http://www.alexandre-boyer.fr)

## I. Objectifs du TP

- ✓ Vérifier la maîtrise des concepts et fonction de base de Matlab, pour le traitement de signal
- ✓ Développer et valider un script Matlab permettant la décomposition en série de Fourier d'une séquence échantillonnée d'un signal périodique
- ✓ Utiliser ce script pour calculer la distorsion harmonique d'un signal

## II. A rendre

A l'issue du TP, vous devrez :

- ✓ rendre un compte rendu répondant aux questions (de manière concise) et présentant les résultats des calculs de validation
- ✓ fournir les scripts (fichiers .m) bâtis en TP

### Consignes :

Ces fichiers **devront être envoyés par mail (alexandre.boyer@insa-toulouse.fr)** (un compte rendu par binôme).

Les scripts devront avoir un nom se conformant à la norme suivante : **noms\_binomes\_Gpe\_X\_TPY\_Z.m**. Les scripts devront présenter un entête contenant les commentaires suivants (lignes commençant par le symbole %) :

- Noms des étudiants
- 3<sup>e</sup> année FISA
- Date
- Description succincte du script

## III. Enoncé du TP

### 1. Partie 1 – Script de décomposition en série de Fourier

Dans un premier temps, il vous est demandé de :

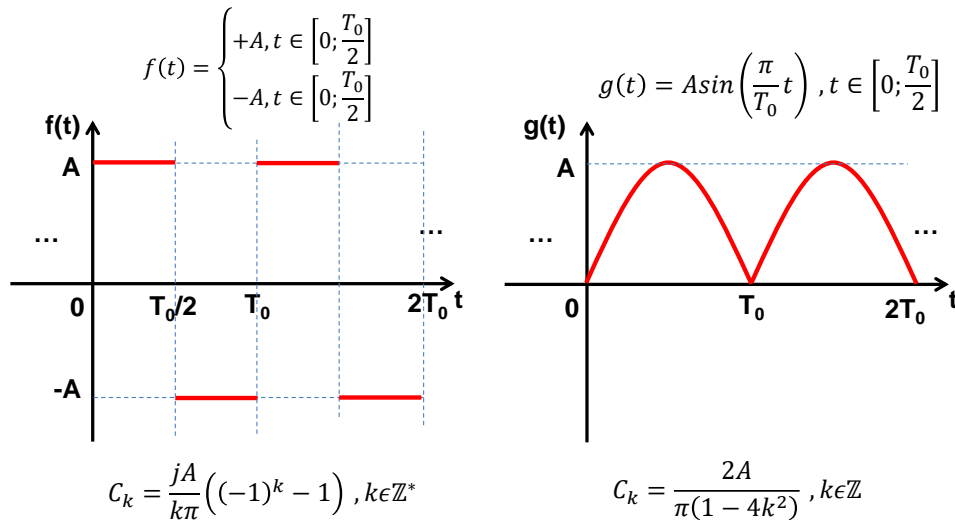
- Compléter un script Matlab/Octave calculant le développement en série de Fourier d'un signal échantillonné périodique quelconque (sous la forme complexe), jusqu'à l'harmonique de rang N, N étant fixé par l'utilisateur. Ce script sera appelé **noms\_binomes\_Gpe\_X\_TP\_1.m**. Ce script devra être déposé sur Moodle avec le compte rendu en ligne.
- Valider ce script en comparant les valeurs calculées et théoriques d'un signal de test.

Le script Matlab est donné par le fichier **Script\_TP1\_1.m**. Créez votre Workspace, copiez-y le script et renommez-le. Celui-ci est un script générique, permettant le calcul des N premières harmoniques du spectre d'une fonction mathématique périodique, définie sur une période. Il est dit générique dans le sens où l'utilisateur définit :

- La fonction
- La période
- La période d'échantillonnage de la fonction
- Le rang maximal N des harmoniques calculées (du rang 0 au rang N)

Remarque : le développement en série de Fourier suppose le calcul de l'intégrale d'un signal à temps continu. Or, ce calcul est impossible avec Matlab ou Octave qui ne peuvent traiter que des signaux discrets, d'où le besoin de travailler avec une version. Seule une approximation numérique des coefficients de Fourier peut être calculée. Dans la pratique, le calcul réalisé par ce script revient à un transformée de Fourier discrète. Néanmoins, cet aspect n'étant pas traité dans ce cours, nous l'ignorons durant le TP.

Pour valider le script, les fonctions périodiques considérées, notées  $f(t)$  et  $g(t)$  sont décrites ci-dessous. Les expressions théoriques des coefficients de Fourier (notés  $C_k$ ) sont fournies. On fixe les variables amplitude du signal  $A = 1$ , et période fondamentale  $T_0 = 4$  s. La fonction est échantillonnée avec un pas d'échantillonnage constant  $dt = 0.04$  s.



1. Le script Script\_TP1\_1.m. est incomplet. Complétez la première partie définissant les valeurs des variables de paramétrage du calcul.
2. Quelle est la valeur maximale de la variable Nmax, définissant le rang maximal du coefficient de Fourier pouvant être calculé par ce script sur la version échantillonnée de  $f(t)$  ?
3. Complétez la boucle de calcul des coefficients de Fourier.
4. Validez le script en l'appliquant sur la version échantillonnée de  $f(t)$ . On comparera le résultat de calcul avec les valeurs théoriques des coefficients de Fourier. On se limitera aux harmoniques de rang 0 à 5.
  - a. Déterminez les expressions littérales des coefficients de Fourier sous leur forme complexe de ces deux fonctions, en fonction du rang et des paramètres de ces fonctions. Vous pourrez reprendre le calcul réalisé en TD.
  - b. A partir des expressions théoriques des coefficients de Fourier, calculez les valeurs de ces coefficients jusqu'à l'ordre  $N = 5$ . Remplissez la deuxième colonne du tableau ci-dessous.

Rang	Calcul théorique	Script Matlab
-5		
-4		
-3		
-2		
-1		

0		
1		
2		
3		
4		
5		

c. A l'aide de votre script, calculez les coefficients de Fourier. Remplissez la troisième colonne du tableau précédent. Comparez les valeurs obtenues avec celles calculées dans la question précédente. Conclure.

5. Que se passe-t-il si les calculs des coefficients est réalisé au-delà de la limite Nmax calculée dans la question 2 ? Faites le test en fixant N = 200. Expliquez le résultat observé.

## 2. Partie 2 – Calcul de la distorsion harmonique produite par un éclairage à LED

Dans un second temps, on souhaite réutiliser le script précédent pour une application concrète : le calcul de la distorsion harmonique du courant consommé par un éclairage à LED. Alimenté sur le réseau secteur (220 V 50 Hz), en raison de la présence de composants non-linéaires (alimentation à découpage, pont redresseur, ...), le profil temporel du courant consommé par l'éclairage à LED est loin d'être sinusoïdal. Cette injection d'harmoniques nuit à la qualité du réseau secteur (déformation des formes d'onde, échauffement, vibrations).

Nous proposons de mesurer la distorsion harmonique sous la forme de deux indicateurs :

- Le **taux de distorsion harmonique DH<sub>k</sub>**. Il est défini, en %, pour chaque harmonique de rang k par la relation suivante, où I<sub>k</sub> est la valeur efficace de l'harmonique de rang k du courant et le dénominateur représente la valeur efficace totale du courant (la composante continue est supposée nulle). N représente le rang mesuré le plus élevé.

$$DH_k(\%) = 100 \frac{I_k}{\sqrt{\sum_{k=1}^N I_k^2}}$$

- Le **taux de distorsion harmonique totale (THD)**. Il s'agit du rapport entre la valeur efficace de l'ensemble des harmoniques de rang > 1 et la valeur efficace totale du courant.

$$THD(\%) = 100 \frac{\sqrt{\sum_{k=2}^N I_k^2}}{\sqrt{\sum_{k=1}^N I_k^2}}$$

Rappel : valeur efficace d'un signal

La valeur efficace d'un signal périodique s(t) de période T<sub>0</sub> est donnée par :

$$S_{eff} = \sqrt{\frac{1}{T_0} \int_0^{T_0} s(t)^2 dt}$$

En pratique, il représente l'amplitude d'un signal continu de même puissance moyenne que s(t). En d'autres termes, la valeur efficace est égale à la racine carrée de la puissance moyenne du signal périodique. Le lien avec l'amplitude des harmoniques peut être retrouvé à l'aide du théorème de Parseval.

Pour que l'éclairage à LED soit commercialisé, celui-ci doit présenter un **THD inférieur à 10 %**. On dispose de deux versions du produit :

- Une version originale, sans filtre
- Une seconde version, dite corrigée, incluant un filtre permettant d'éliminer les harmoniques de rang supérieur à 1.

On souhaite donc évaluer la distorsion harmonique de notre éclairage à LED. Pour cela, on souhaite afficher le taux de distorsion harmonique en fonction du rang ainsi que le THD pour une mesure de courant, fourni sous la forme d'un enregistrement du courant pendant une période avec un pas d'échantillonnage constant. L'import des données de mesure se fera à l'aide de la commande **audioread**, dont la syntaxe est la suivante :

[signal,Fe] = audioread(nom\_fichier) , où signal est le vecteur contenant les points associés au signal, Fe la fréquence d'échantillonnage du signal et nom\_fichier le chemin d'accès et le nom du fichier .wav. Exemple : si le fichier signal\_inconnu.wav est situé dans le répertoire U:\TP\_Signal, alors la commande Matlab a tapée est :

```
[signal,Fe] = audioread("U:\TP_Signal\ signal_inconnu.wav");
```

On dispose de deux enregistrements de courant :

- La mesure du courant de l'éclairage à LED en version originale (sans filtre harmonique) : I\_LED\_sans\_filtre.wav
- La mesure du courant de l'éclairage à LED dite corrigée (avec filtre harmonique) : I\_LED\_avec\_filtre.wav

Reprenez le script développé dans la partie précédente. Renommez-le selon la convention donnée dans les consignes (ce script sera appelé **noms\_binones\_Gpe\_TP\_2.m**). Modifiez-le pour :

- Tracer le profil temporel du courant mesuré
- Afficher la répartition de toutes les harmoniques contenues dans le signal
- Afficher la répartition de la distorsion harmonique en fonction rang
- Afficher la valeur du THD

Ce script devra être déposé sur Moodle avec le compte rendu en ligne. Appliquez votre script sur les deux enregistrements et répondez aux questions suivantes :

- Dans les versions sans et avec filtre, à partir de quels rangs le taux de distorsion harmonique devient inférieur à 5 % ?
- Sur quelle(s) harmonique(s) le filtre a-t-il eu un effet ?
- Le filtre permet-il de respecter la contrainte sur le THD ?

## IV. Guide de prise en main de MATLAB et OCTAVE

Matlab est un logiciel commercial de calcul matriciel développé par la société MathWorks. Il consiste essentiellement en un interpréteur de commandes, écrites dans un langage de programmation propre à Matlab. Les commandes Matlab sont saisies et interprétées ligne à ligne dans une fenêtre appelée console. Elles peuvent aussi être regroupées dans un fichier appelé script, avec une extension .m. Ce script pourra être appelé à tout moment.

GNU Octave est un logiciel libre de calcul numérique, très comparable à Matlab. Les instructions sont quasiment toutes compatibles. Le principe de fonction est et l'interface sont très proches. Des scripts peuvent aussi être écrits et portent la même extension .m. Ainsi, un script écrit sous Octave est compatible avec Matlab et réciproquement (hormis dans le cas de fonctions très spécifiques de Matlab).

Dans ce TP, nous travaillerons principalement avec Matlab. Il est accessible dans l'ensemble des salles de TP de l'INSA. Cependant, puisque sa licence est payante, vous ne pourrez pas l'utiliser sur vos machines personnelles. Chez vous, il est donc conseillé d'utiliser Octave, qui est téléchargeable à l'adresse suivante [www.gnu.org/software/octave/](http://www.gnu.org/software/octave/).

Ce document se concentre donc principalement sur la présentation de Matlab, puisque les principes et le langage de programmation est similaire sous Octave. Seule l'interface change dont nous présenterons donc les éléments principaux. La description de Matlab restera succincte et ne se focalisera que sur les éléments les plus importants pour les TP de filtrage et signal.

### 3. Lancement et présentation des interfaces

#### a. Matlab

A partir de l'environnement Windows, cliquez sur l'icône **Matlab R2020** pour lancer le logiciel. Il peut aussi se lancer depuis le menu Démarrer de Windows. L'interface présentée à la Fig. 1 s'ouvre, montrant ses principales parties. Vous pouvez taper des commandes dans la **Console**, à partir du curseur indiquant la ligne de commande en cours. Pour exécuter une commande que vous venez de taper, appuyer sur la touche **Entrée**. Le résultat s'affiche sur la console.

Un ensemble de commandes peuvent être regroupées dans un même fichier, appelé **Script** (fichier .m). L'exécution de ce script entrainera l'exécution successive des commandes du script.

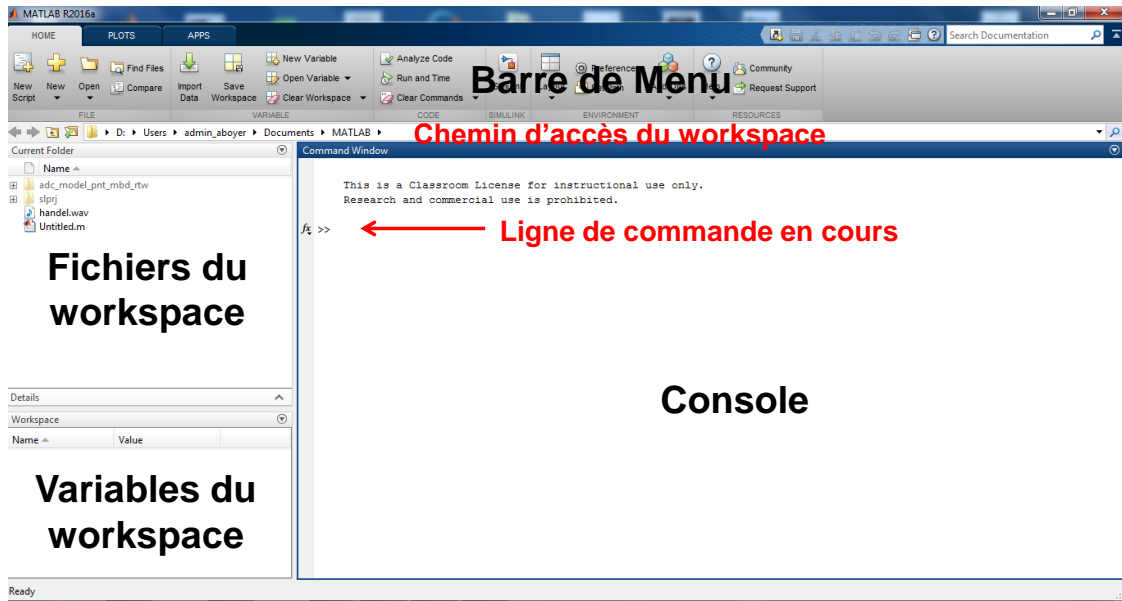


Fig. 1 - Interface de Matlab à l'ouverture du logiciel

**b. Octave**

A partir de l'environnement Windows, cliquez sur l'icône **GNU Octave (GUI)** pour lancer le logiciel. Il peut aussi se lancer depuis le menu Démarrer de Windows. L'interface présentée à la Fig. 2 s'ouvre, montrant ses principales parties. Vous pouvez taper des commandes dans la **Console**, à partir du curseur indiquant la ligne de commande en cours. Pour exécuter une commande que vous venez de taper, appuyer sur la touche **Entrée**. Le résultat s'affiche sur la console.

Un ensemble de commandes peuvent être regroupées dans un même fichier, appelé **Script** (fichier .m). Celui-ci est éditable à partir de l'onglet **Editeur**. L'exécution de ce script entrainera l'exécution successive des commandes du script.

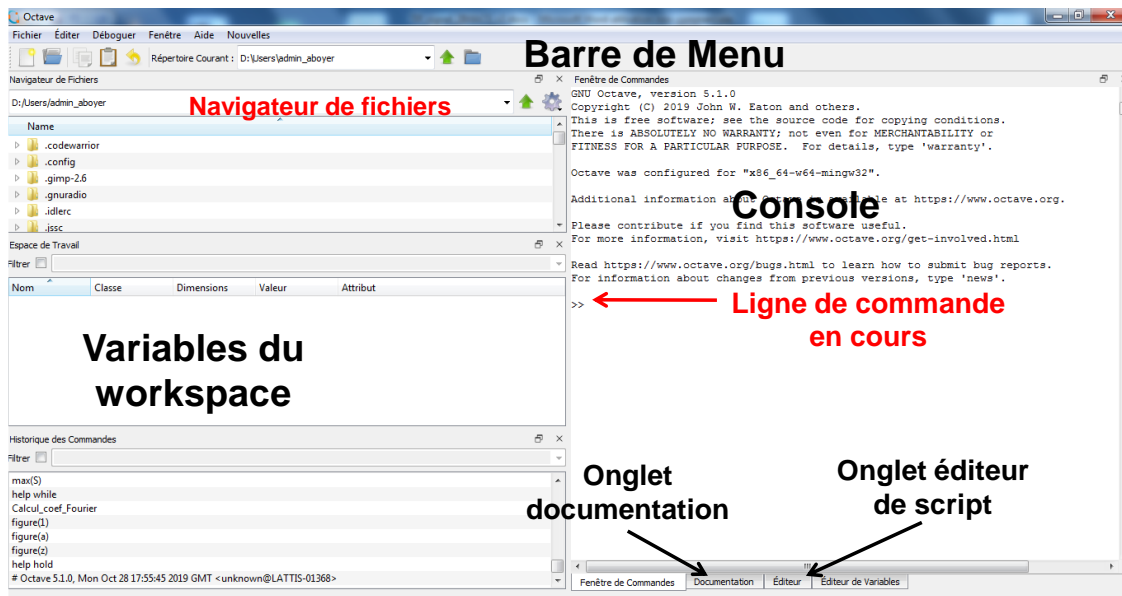



Fig. 2 - Interface d'Octave à l'ouverture du logiciel

## 4. Workspace

Une particularité fondamentale de Matlab et d'Octave est l'existence d'un espace de travail ou **Workspace**. Il correspond au répertoire dans lequel l'ensemble des variables et scripts seront sauvegardés et accessibles à tout moment. Matlab et Octave s'ouvrent avec un workspace par défaut. Vous pouvez le modifier à tout moment en sélectionnant le chemin d'accès de votre propre workspace à partir de la ligne située au-dessus de la console :

- sous Matlab à l'aide de l'icône 
- sous Octave à l'aide du navigateur de fichier

L'ensemble des fichiers situés dans le workspace, ainsi que les variables créées lors de l'exécution des commandes dans la partie gauche de l'écran (Fig. 1 et Fig. 2).

## 5. Principes et instructions de base de Matlab / Octave

Dans cette partie, seules les concepts et les instructions les plus basiques, ainsi que celles indispensables pour le TP, sont présentées. Pour plus d'informations sur les instructions, n'hésitez pas à consulter l'aide en ligne des logiciels :

- accessible sur Matlab à l'aide de l'icône  ou du raccourci clavier F1
- accessible sur Octave dans le **menu Aide** ou dans l'**onglet Documentation**

L'aide sur une commande peut aussi être obtenue en tapant dans la console la commande 'help nom\_de\_l\_instruction'.

### a. Les variables et les matrices (tableaux)

Matlab ou Octave manipulent différents types des données : nombres entiers, nombres réels, nombres complexes, caractères, booléens. En pratique, toutes les grandeurs numériques sont définies comme des **Double**, c'est-à-dire des nombres flottants codés sur 64 bits. Il est important de noter que Matlab ou Octave supportent nativement les **nombres complexes**.

Les variables sont toutes des tableaux définis au moment de leur affectation. Contrairement à de nombreux autres langages de programmation, il n'y a donc pas besoin de les déclarer. Une variable est repérée par un nom quelconque (évités les symboles d'opérations élémentaires). Attention **Matlab et Octave distinguent les minuscules des majuscules !**

Un nombre, qu'il soit entier, réel ou complexe est un tableau de taille 1x1. On parle alors de variable **scalaire**. Un **vecteur ligne** est un tableau de taille 1x N et un **vecteur colonne** est un tableau de taille N x 1, avec N un entier positif quelconque. Une **matrice** est un tableau de taille M x N (deux dimensions, M x N x P (trois dimensions), voire à plus de trois dimensions, où M, N, P sont des entiers positifs quelconques

Il faut retenir que Matlab et Octave ont été conçu pour **faciliter les opérations matricielles**.

### b. Affectation des variables

L'affectation d'une variable scalaire se fait par le symbole '='. L'affectation des variables d'un vecteur ou d'une matrice se fait de la manière suivante :  $A = [1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9]$ . Dans cet exemple, on a construit une matrice de 3 lignes et 3 colonnes (3x3) égale à :

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

Les éléments séparés par un espace (ou une virgule) sont sur la même ligne. Le séparateur ';' indique un changement de ligne. Pour effectuer cette opération, dans la fenêtre de commande (console),



tapez la commande 'A = [1 2 3;4 5 6; 7 8 9]' puis appuyez sur la touche entrée. Le résultat s'affiche sous la commande que l'on vient de taper. On pourra remarquer que la variable A est apparue dans la partie Workspace de l'écran, indiquant le type (Double) et les dimensions de la variable (3x3). Elle est donc disponible à tout moment en tapant A, jusqu'à ce qu'on arrête Matlab ou Octave et qu'on efface la variable via la commande '**clear A**', ou toutes les variables via l'instruction '**clear all**'.

Puisqu'une variable est un tableau, l'accès à une variable s'effectue en indiquant le numéro de la ligne et de la colonne, par exemple A(1,3). Avec l'exemple de la matrice 3x3 ci-dessus, l'instruction précédente renvoie la valeur 3, qui correspond à l'élément rangé à l'intersection de la ligne 1 et de la colonne 3 de la matrice A.

Si on accède en tapant directement le nom de la matrice, on obtient l'ensemble de la matrice. Si elle ne contient qu'un seul élément, il n'est pas utile d'indiquer l'indice. Par exemple, si b = 1, alors b(1,1) renverra la même valeur.

Il est possible d'accéder ou d'affecter plusieurs éléments d'un tableau, rangés dans une même ligne ou une même colonne. Pour cela il suffit d'indiquer la plage de sélection sur les lignes ou colonnes considérées à travers le symbole ':'. Ainsi, l'instruction 'm:n', avec n > m permet de sélectionner toutes les valeurs présentes entre les lignes ou colonnes d'indice m à n. Par exemple, avec l'exemple précédent, la commande A(1:2,1) renvoie les éléments des lignes 1 à 2 de la première colonne de la matrice A, c'est-à-dire [1;4]. La commande A(1:2,1:2) renvoie la matrice 2x2 égale à :

$$\begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix}$$

Matlab et Octave proposent plusieurs commandes permettant de générer plusieurs matrices simples. Celles-ci sont résumées dans le tableau ci-dessous.

Commande	Description
zeros(m,n), m et n deux entiers non nuls quelconques	Crée une matrice de dimensions m x n remplie de 0
ones(m,n), m et n deux entiers non nuls quelconques	Crée une matrice de dimensions m x n remplie de 1
eye(n), n un entier non nul quelconque	Crée une matrice identité de dimensions n x n
linspace(min,max,n) ou min:n:max	Crée un vecteur ligne contenant n valeurs également espacées entre min et max
logspace(min,max,n)	Crée un vecteur ligne contenant n valeurs espacées de manière logarithmique entre 10 <sup>min</sup> et 10 <sup>max</sup>

Tableau 1 - commandes permettant de construire quelques matrices simples

**c. Les opérations arithmétiques élémentaires**

Les opérations élémentaires sur les nombres entiers, réels ou complexes sont :

- addition : '+'
- soustraction : '-'
- multiplication : '\*'
- division : '/'
- puissance : '^'

Ces opérations s'appliquent aussi aux nombres complexes. Deux variables existent par défaut pour le nombre complexe  $\sqrt{-1} : i$  et  $j$ , et vous permettent de créer des nombres complexes. Si jamais vous créez une variable appelée i ou j, vous écraserez la valeur précédente et serez obligés de définir une nouvelle variable pour  $\sqrt{-1}$ .

L'accès à la partie réelle d'une variable complexe x se fait par l'instruction **real(x)**, tandis que l'accès à sa partie imaginaire par l'instruction **imag(x)**. Le module peut se déterminer par l'instruction

**abs(x)**, utilisée aussi pour la valeur absolue. Le conjugué est obtenu à l'aide de l'instruction `conj(x)` ou bien à l'aide du **symbole ' .**

Exemple : soit  $x = 3+4*i$

- `real(x)` renvoie 3
- `imag(x)` renvoie 4
- `abs(x)` renvoie 5
- `conj(x)` ou `x'` renvoie 3-4i

Ces opérations s'appliquent aussi sur des matrices, à condition de respecter les contraintes sur les tailles des matrices associées à ces opérations. Par exemple, l'addition de deux matrices suppose qu'elles aient rigoureusement la même taille. La multiplication d'une matrice de dimensions  $M \times N$  n'est possible qu'avec une matrice de dimension  $N \times P$  et donnera comme résultat une matrice de dimension  $M \times P$ .

Opérations matricielles terme à terme :

Lorsque le symbole '.' est placée devant un symbole de multiplication ou de division, on effectue alors une opération non pas au niveau matriciel, mais terme à terme. Par exemple, soit  $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$  et

$B = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$ . La commande `C=A*B` renvoie  $C = \begin{bmatrix} 4 & 7 \\ 8 & 15 \end{bmatrix}$ , correspondant au produit des deux matrices.

La commande `C=A.*B` renvoie  $C = \begin{bmatrix} 0 & 2 \\ 6 & 12 \end{bmatrix}$ , correspondant à la multiplication terme à terme des éléments de la matrice.

**d. Les fonctions mathématiques les plus courantes**

Le tableau ci-dessous résume la plupart des fonctions mathématiques les plus courantes.

Commande	Description
<code>sqrt(x)</code>	Renvoie la racine carrée de la variable x
<code>cos(x)</code> , <code>sin(x)</code> , <code>tan(x)</code>	Renvoie le cosinus, le sinus, la tangente de la variable x (exprimée en radians)
<code>acos(x)</code> , <code>asin(x)</code> , <code>atan(x)</code>	Renvoie le résultat, exprimé en radians, des fonctions arccos, arcsin et arctan appliquées sur la variable x.
<code>log(x)</code>	Logarithme népérien
<code>log10(x)</code>	Logarithme en base 10
<code>exp(x)</code>	Fonction exponentielle
<code>abs(x)</code>	Renvoie le module de la variable x. Si x est un nombre réel, il renvoie aussi la valeur absolue
<code>min(A)</code> , <code>max(A)</code>	Renvoie les valeurs min et max contenues dans un vecteur A

Tableau 2 - Fonctions mathématiques les plus courantes

**e. Instructions de boucles et de tests**

Matlab et Octave proposent deux types d'instructions de boucle :

- la boucle `for` : format :  

```
for (k=1:n)
    instructions
    ...
end,
```
- la boucle `while` : format

```
while (condition)
    instructions
    ...
end,
```

Ils proposent une instruction de test du type if - then - else, dont le format est donné ci-dessous :

```
if (condition1)
    instructions
    ....
elseif (condition 2)
    instructions
    ....
else
    instructions
    ....
end,
```

Les conditions sont liées à des tests booléens sur la valeurs prises par une variables. Le tableau ci-dessous résume les principaux opérateurs de test :

==	Test d'égalité	~=	Test de différence
<, >	Inférieur, supérieur	&	Et logique
<=	Inférieur ou égal		Ou logique
>=	Supérieur ou égal	~	Négation logique

Tableau 3 - Principaux opérateurs de test

## 6. Scripts

Matlab et Octave peuvent exécuter un ensemble d'instructions contenues dans un fichier appelé script. Ces fichiers ont une extension .m. Toutes les commandes disponibles dans la console sont utilisables dans un script. Un fichier script est composé d'un ensemble d'instructions Matlab/Octave. Si le fichier a le nom prog.m, alors la commande prog va l'exécuter.

Matlab comme Octave propose un éditeur permettant d'écrire un script. Les variables d'un programme script sont globales, et son exécution va changer leurs valeurs dans l'espace de travail. Il est aussi possible d'exécuter un script depuis l'éditeur à l'aide :

- sur Matlab, l'icône  **Run (F5)**
- sur Octave, l'icône  ou le raccourci clavier **F5**.

Il est recommandé de terminer chaque ligne du script par un **point-virgule** ';'. Sinon, le résultat de l'exécution de la commande sera affiché sur la console. Dans le cas d'un script de plusieurs centaines de lignes, cela peut s'avérer problématique.

Un script peut présenter une structure algorithmique complexe. A l'instar de tout programme informatique, il est vivement conseillé d'ajouter suffisamment de commentaires pour que le script reste lisible et compréhensible par tout le monde. Une ligne de commentaire commence par le **symbole** '%'.

## 7. Affichage d'un texte à l'écran

Les textes à afficher ne sont rien d'autres que des chaînes de caractères. Celles-ci sont définies en les encadrant par des quotes : s = 'texte'. Une chaîne de caractère peut être affichée sur la console à l'aide de l'instruction: disp('texte').

Souvent, une chaîne de caractère à afficher peut contenir la valeur d'une variable. Cela peut être possible à l'aide de la commande sprintf qui permet d'intégrer à la chaîne de caractères des variables de

type donnée, qui seront convertis automatiquement en chaîne de caractère. Par exemple, la commande ci-dessous génère une chaîne de caractère sauvegardée dans la variable Message. Celle-ci contiendra la valeur de la variable Pmoy qui sera placée entre le symbole '=' et le mot 'Watts'.

```
Message = sprintf('Puissance moyenne du signal reconstitue = %f Watts.\n',Pmoy);
```

Le symbole '%f' permet de définir le type de Pmoy : ici un Double. Le symbole '\n' signifie retour chariot, c'est-à-dire que lorsque la chaîne de caractère sera affichée, le curseur reviendra à la ligne. Pour afficher cette chaîne de caractère, il suffira d'exécuter la commande disp(Message).

## 8. Affichage graphique

Matlab et Octave proposent de nombreuses instructions permettant de créer des graphiques en deux et trois dimensions. Nous ne présenterons ici que les instructions principales pour les graphiques à deux dimensions.

L'affichage d'un graphique se fait sur une fenêtre graphique offrant toutes les commandes interactives avec le graphique. Il est nécessaire de la créer avec l'instruction '**figure**'. Il est possible de lui spécifier un numéro : 'figure(1)'.

Un graphique à deux dimensions permet l'affichage de l'évolution des valeurs d'un vecteur y par rapport à celles d'un vecteur x. **Ces deux vecteurs doivent faire la même dimension !** Ce tracé est réalisé par l'instruction : '**plot(x,y)**'.

Si on a un deuxième graphique à tracer, par exemple l'évolution du vecteur z en fonction du vecteur t, si on tape directement la commande 'plot(t,z)', il écrasera le graphique précédent. Si on souhaite le conserver, il y a deux possibilités :

- on trace le nouveau graphique dans une nouvelle fenêtre graphique. On en crée une nouvelle avec la commande figure.
- on le trace sur le même graphique, en tapant au préalable la commande '**hold on**'. Toute nouvelle instruction plot tracera une courbe sur le même graphique. L'action de l'instruction 'hold on' est effectuée à l'aide de l'instruction 'hold off'.
- on aurait pu aussi taper la commande plot(x,y,t,z) pour tracer directement les deux courbes sur le même graphique.

A cette fenêtre graphique, de nombreuses autres instructions sont disponibles pour modifier l'aspect du graphique, configurer les propriétés de la fenêtre, des axes, des courbes ... Le tableau ci-dessous liste les commandes principales.

plot(x,y)	Tracé du graphique du vecteur y en fonction du vecteur x sous la forme d'une courbe
stem(x,y)	Tracé du graphique du vecteur y en fonction du vecteur x sous la forme de points discrets
title('Texte')	Définit et affiche le titre du graphique
xlabel('Texte')	Définit et affiche le titre de l'axe des abscisses
ylabel('Texte')	Définit et affiche le titre de l'axe des ordonnées
grid	Affiche une grille sur le graphique
legend('txt1','txt2'...)	Ajout de la légende associée aux courbes affichées sur le graphique

Tableau 4 - Principales commandes de configuration d'un graphique

Pour modifier les propriétés des courbes, référez-vous à l'aide en ligne de Matlab ou d'Octave. Elles fournissent un grand nombre d'exemples qui vous aideront à mettre en forme vos graphiques.

## 9. Quelques conseils pour accélérer l'exécution de vos scripts

Matlab et Octave sont optimisés pour les calculs basés sur la manipulation de vecteurs et de matrices, mais beaucoup moins sur des calculs basés sur des boucles. Il est donc conseillé de bâtir des algorithmes qui exploitent au mieux cette particularité. On parle alors de vectorisation d'un algorithme. Pour illustrer ce principe, considérons l'exemple suivant. On souhaite générer un vecteur contenant 10 points d'un signal sinusoïdal de 10 Hz, avec la base de temps associée. Sous Matlab ou Octave, l'algorithme suivant permettant de générer ces vecteurs est :

```
for t=1:10
    temps(t) = t;    %base de temps en seconde
    signal(t) = 2*sin(2*pi*10*t);    %signal sinusoïdal
end,
```

Bien que cet algorithme fonctionne, son exécution est moins rapide que sa version vectorisée, présentée ci-dessous :

```
temps = 1:1:10;
signal = 2*sin(2*pi*10*temps);
```

La préallocation permet de rendre plus rapide l'exécution des boucles. C'est le cas où les résultats de calcul à l'intérieur d'une boucle doivent être stockés dans un vecteur ou une matrice. Dans l'exemple précédent, les vecteurs temps et signal peuvent être préalloués pour gagner du temps, comme le décrivent les lignes ci-dessous.

```
temps = zeros(1,10);
signal = zeros(1,10);
for t=1:10
    temps(t) = t;    %base de temps en seconde
    signal(t) = 2*sin(2*pi*10*t);    %signal sinusoïdal
end,
```

De plus, il est conseillé de limiter le plus possible le nombre d'instructions à exécuter à l'intérieur des boucles.