REPUBLIQUE **A**LGERIENNE **D**EMOCRATIQUE ET **P**OPULAIRE
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

Centre **U**niversitaire **N**our **B**achir **E**l-**B**ayadh
Institut des **S**ciences
Département de **T**echnologie

Polycopié de cours

UEM 2.1

INTITULE DU MODULE

Informatique 3

COURS / TP

Dr. TADJEDDINE Ali Abderrazak (MCB)
Dr. BENDELHOUM Mohammed Soufiane (MCA)
Dr. BENDJILLALI Ridha Ilyas (MCB)

Centre Universitaire Nour Bachir – El Bayadh

2022-2023

Avant-propos

Ce manuel de cours / TP, intitulé « Informatique 3 » est une matière méthodologique qui est étudiée par les étudiants en 2ème année licence, semestre 3, troc commun sciences et technologies de toutes les spécialités. Le programme de ce cours/TP est conçu pour apprendre aux étudiants une compréhension fondamentale sur la programmation et le calcule numérique par Matlab.

Bien que l'environnement Matlab signifie la contraction de MATrix LABoratory en Anglais i.e. laboratoire des matrices. MATLAB est un langage de programmation informatique puissant, dont vous pouvez l'utiliser comme une calculatrice, et c'est une bonne technique pour essayer des idées que vous pourriez utiliser dans votre programme. Toutefois, lorsque vous avez dépassé l'étape de l'expérimentation, vous comptez généralement sur MATLAB pour créer un programme qui vous aide à effectuer des tâches : Régulièrement, Facile et Rapide. Ces trois caractéristiques ne vous disent pas tout ce que MATLAB peut faire, mais elles vous fournissent des idées que vous pouvez poursuivre et utiliser à votre avantage.

Les disciplines des Sciences, de la Technologie, de l'Ingénierie et des Mathématiques (STIM) mettent actuellement l'accent sur les sujets de simulation des modèles mathématiques avent l'expérimentation. L'innovation de tout le tri requiert ces disciplines, comme le font de nombreux métiers pratiques. MATLAB dispose d'une riche et grande boîte à outils pour STIM qui comprend : Statistiques, Simulation, Traitement d'images, Traitement symbolique et l'Analyse numérique.

Méthode de rédaction

Les suggestions pour l'amélioration du cours seront très appréciées. Bien que toutes les précautions aient été prises pour éliminer les erreurs, il est très difficile de prétendre à la perfection. Je serai très reconnaissant aux enseignants et étudiants et aux membres d'utilisateurs de ce cours s'ils signalent toute erreur qui aurait pu s'y glisser.

Afin de simplifier l'apprentissage aux étudiants ; chaque fiche de cours/TP contient quatre partie :

1. **L'objectif de TP ;**
   Couvre le but de TP et les principales idées enseignée.
2. **Partie 01 : Partie théorique (l'interface de Matlab) ;**
   Les définitions des commandes et instructions utilisées lors de la séance.
3. **Partie 02 : Partie Simulation (MATLAB – SIMULINK) ;**
   Des exercices résolus incluant des figures d'apprentissage.
4. **Partie 03 : Partie expérimentale (MATLAB – SIMULINK).**

   Des exercices pour s'adapter avec les problèmes de programmation et l'application des notions visée sur les parties 1 et 2.

Organisation des fiches TP

Conformément au canevas officiel, ce cours est divisé en huit fiches :

TP N°	L'intitulé	Durée
TP 1	**Présentation d'un environnement de programmation scientifique (MATLAB)**	**1 semaine**
TP 2	**Lecture, affichage et sauvegarde des données**	**2 semaines**
TP 3	**Fichiers script et Types de données et de variables**	**2 semaines**
TP 4	**Vecteurs et matrices**	**2 semaines**
TP 5	**Instructions de contrôle (Boucles for et While, Instructions if et switch)**	**2 semaines**
TP 6	**Fichiers de fonction**	**2 semaines**
TP 7	**Graphisme (Gestion des fenêtres graphiques, plot)**	**2 semaines**
TP 8	**Utilisation de toolbox**	**2 semaines**

Table des Matières

Fiche TP : 01
Présentation d'un environnement de programmation scientifique (MATLAB)

Centre universitaire Nour Bachir El-Bayadh

Institut des sciences

Département de technologie

Spécialité : HYD, GC, ETT, ELN et TLC

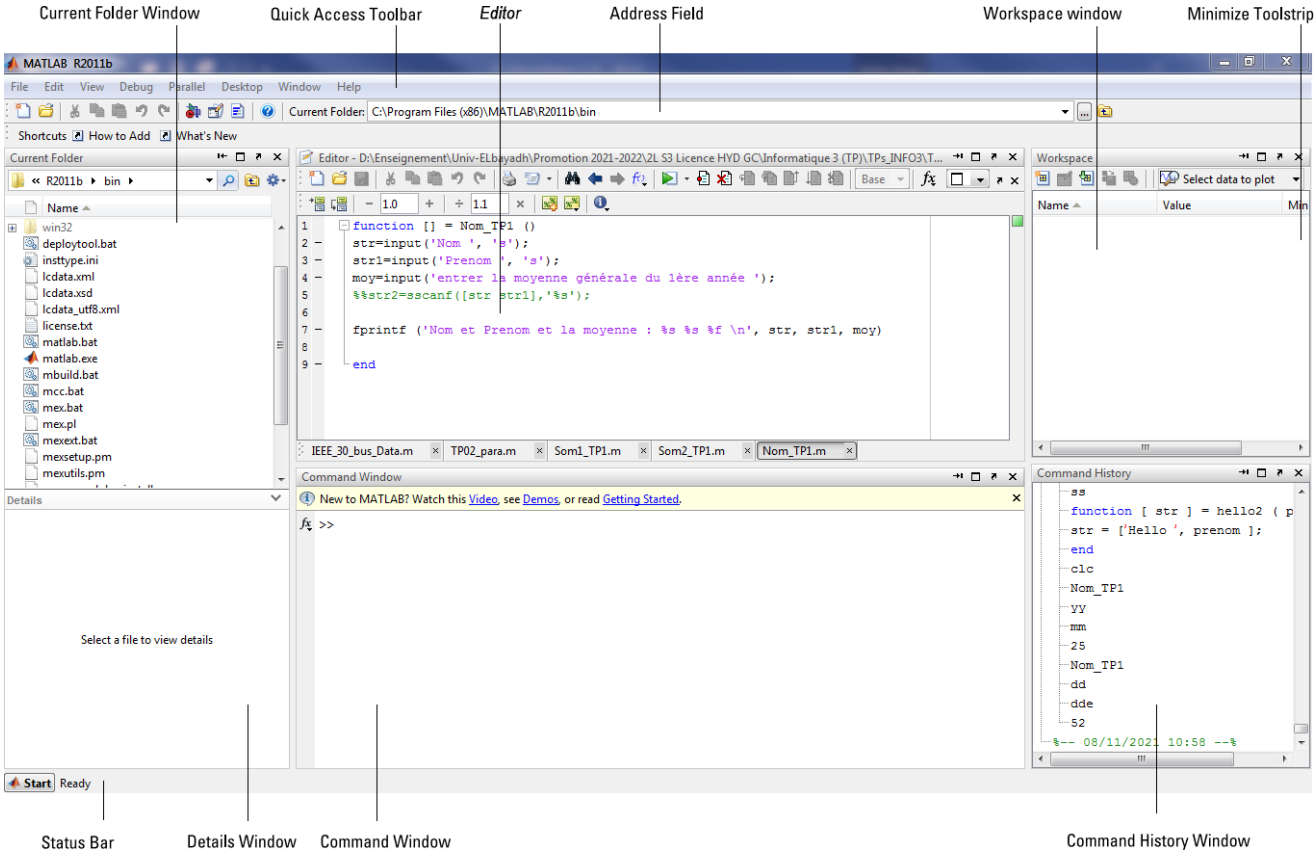
Niveau : Licence 02 (semestre 03)

Travaux Pratiques Informatique 3

TP 01 : Présentation d'un environnement de programmation scientifique (MATLAB)

L'objectif de TP :

Ce TP01 et TP02 a pour but de vous familiariser avec l'interface et les bases de l'environnement MATLAB afin d'utiliser quelques fonctions de base pour la lecture, l'affichage et le sauvegarde des données.

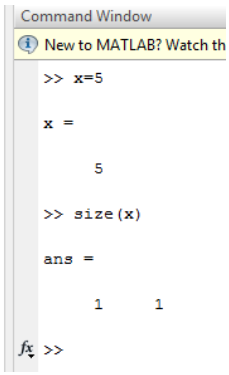
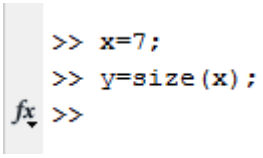


Partie 01 : Partie théorique (l'interface de Matlab)

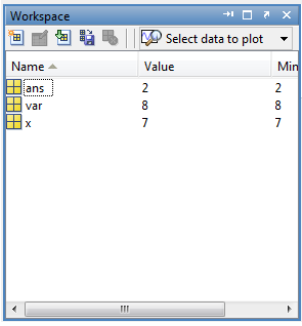
Nous allons nous familiariser avec l'interface de Matlab. Selon la version utilisée, l'interface peut changer légèrement mais les points centraux resteront identiques.

Command Windows

Une **variable** scalaire (**x**=5 : ***affectation de la valeur 5 à la variable x***) est vue par MATLAB comme une matrice de dimension 1x1 (ligne x colonne). Comme le montre l'exemple suivant dans lequel on affecte à la variable x la valeur 5 et on demande ensuite ses dimensions par la fonction ***size*** (***x***) dans l'espace de **Command Window:**

>> x=5 >> size(x)	Utilisation de point-virgule	*ans* (answer)
	Afin d'éviter l'affichage des résultats, il suffit de suivre la commande par un point-virgule (;). 	Matlab définit une variable **ans**, elle est une matrice de taille 1x1 (une ligne par une colonne).

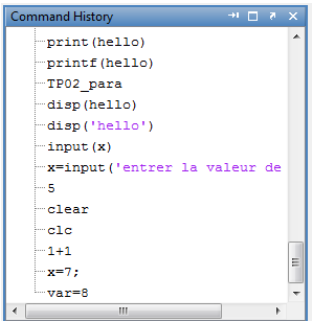
Une *commande* ***clc*** : Cette commande permet de vider l'écran de **Command Window** pour voir une fenêtre propre. Une *commande* ***clear*** : Cette commande permet **de supprimer toutes les variables** dans l'espace de **Workspace** pour repartir sur une fenêtre propre.



Workspace (go to Window -> Workspace)

Dans cette fenêtre, on obtient la liste des variables connues par Matlab. Il est possible de double-cliquer sur une variable pour l'afficher. Un clic-droit sur les variables donne de nombreuses options telles que : Copiez, Collez, Supprimez etc.

Command History (go to Window -> Command History)



L'espace Command History conserve une trace de toutes les opérations qui ont été réalisées sur l'espace Command Window. On peut également remonter dans la liste de commandes en se plaçant dans la **Command Window** et en pressant les flèches de direction ↑ ↓.

Current Folder (go to Window -> Current Folder)

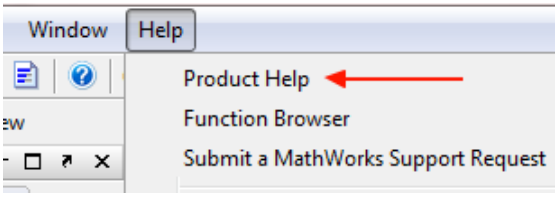
C'est le dossier qui contient le ficher de script, la programmation, le travail réaliser sur Matlab.

Help

L'espace de l'aide est essentielle lorsque l'on programme avec un langage de haut-niveau comme Matlab, où le nombre de fonctions est très important et la syntaxe est parfois complexe. Pour accéder à l'aide on peut au choix sélectionner une fonction et presser **F1** sur le clavier, ou bien taper dans l'espace **Command Window** : **help cos**, ***help input*** … etc. Il est essentiel que vous vous familiarisiez avec les outils de l'aide de Matlab pour réussir dans ce Cours/TP.

Command	***Description***
helpwin	ouvre une fenêtre contenant la liste des commandes Matlab ainsi que leurs documentations
help	donne la liste de toutes les commandes par thèmes
help nom	décrit la fonction nom.m
lookfor nom	Recherche une instruction à partir du mot clé nom

Use: help → Product Help to display the help Window



Script

Le script est le fichier avec l'extension '**.m**' qui contient le programme plus simplement. Il s'agit d'une liste de commandes et des **fonctions**.

Fonction

Une fonction va permettre de rentrer des arguments en entrée et d'obtenir différentes variables en sortie.

Editor

La plupart de votre travail sous Matlab va consister à créer ou modifier des fichiers avec l'extension « **.m** » qui définit les fichiers de Matlab. Lorsque l'on réalise une tache sous Matlab, il est très souvent possible de le faire en utilisant uniquement la **Command Window**. Cependant lorsque cette tache devient plus complexe (plusieurs dizaines de ligne de code) ou que l'on souhaite pouvoir la transmettre à quelqu'un d'autre simplement, on utilise la fenêtre **Editor**. On crée un fichier .m qui peut être au choix un **script** ou une fonction.

Partie 02 : Partie Simulation (MATLAB – SIMULINK)

Pour commencer on fixe le **USERPATH**, le dossier qui contient votre fichier (.m)

- Crée un dossier dans le Bureau avec le nom : **TP_INFO3**.
- Crée un autre dossier dans le dossier précédant avec le nom : **TP01_02_INFO3**.
- Dans l'espace **Current Folder**, choisir le dossier TP01_02_INFO3 pour le 1er TP et ainsi de suite pour les autres TPs.

Exercice1 : "Hello World"

Il s'agit d'un bref programme pour mettre en place les différents éléments nécessaires.

- Crée ensuite **new script** (.m) dans le dossier **TP01_INFO3**.
- Ecrire dans la 1ère ligne :

```
1. varibale= 'Hello world';
```

Run

- Sauvegarder le fichier avec le nom hello.m.

Run

Conclure !

Exercice2: " Somme de deux nombres "

On va essayer ici de réaliser un script pour en faire une **fonction** qui prend deux nombres en entrée et retourne la somme des deux.

- Cree un fichier **new script** (Ctrl +N).
- Réaliser un programme qui va calculer la somme de deux nombres.
- Sauvegarder avec le nom du fichier : **Som_xy_TP1.m (utiliser le tiret de 8 « _ » et non « - »)**

```
1 -     f=3;
2 -     d=4;
3 -     r=f+d;
```

Que remarquez-vous ?

Exercice3: " Somme de deux nombres quelconque "

On veut une **fonction** qui prend deux nombres aléatoire en entrée et renvoie par affichage la somme des deux.

- Cree un fichier **new script** (Ctrl +N). Nommer : **Som1_TP1.m**, Sauvegarder !!
- Ecrire la fonction :

```
1   function [z] = Som1_TP1 (x,y)
2 -   z=x+y
3 -   end
```

- Run
- Une erreur s'affiche :

```
Error using Som1_TP1 (line 2)
Not enough input arguments.
```

```
>> Som1_TP1 (4,5)

z =

     9


ans =

     9
```

On a défini une fonction **Som1_TP1**(x,y) avec deux arguments, mais on a appelé la fonction sans arguments ; alors pour corriger cette erreur il faut donner une valeur à x et y, on l'appelle à la fonction Som1_TP1 dans l'espace **Command Window** par : On remarque que Matlab a affiché le variable ***z*** puisque nous n'avons pas utiliser le point-virgule et la variable par défaut ***ans***.

Exercice4 : " Comment trouver l'information voulue "

Utiliser l'espace **Command Window**

- help log
- help mod

Exercice5: " Somme de deux nombres quelconque par choix"

On veut réaliser un programme qui prend trois chiffres par choix en entrée et renvoie un affichage de produit des premiers chiffres et la racine de produit des derniers nombres.

- Cree un fichier **new script** (Ctrl +N), nommer : **Som2_TP1.m**, Sauvegarder !!
- Ecrire la fonction :

```
function [] = Som2_TP1()
x=input('entrer le 1er nombre ')
y=input('entrer le 2e nombre ');
z=input('entrer le 3e nombre ');
a=x*y;
disp('^_^');
fprintf('le produit des 1er nombres est: %f \n', a)
b=sqrt(y*z);
fprintf ('la racine des deux dernier nombres est: %.2f \n', b);
disp('^_^');
end
```

On a utilisé 4 fonctions « input (), sqrt(), dips() et fprintf() »,

- Utiliser le « **help input () »** pour découvrir ces fonctions.

Quelle est le but de ces fonctions ?

Partie 03 : Partie expérimentale (MATLAB – SIMULINK)

Exercice5: " Introduire Nom et Prénom "

On va essayer ici de réaliser un script pour en faire une **fonction** qui prend 2 arguments votre nom et prénom en entrée et retourne votre nom et prénom ***sur la même ligne*** en sortie.

- Cree un fichier **new script** (Ctrl +N). Nommer : **Name_TP1.m**, Sauvegarder !!

Exercice6: " Introduire Nom et Prénom + la moyenne générale du 1ère année"

On va réaliser un script pour une **fonction** qui prend 3 arguments votre nom et prénom et la moyenne générale de la 1ère année comme des entrées et renvoie votre nom et prénom ***et la moyenne sur la même ligne*** en sortie.

- Cree un fichier **new script** (Ctrl +N). Nommer : **Name1_TP1.m**, Sauvegarder !!

Fiche TP : 02
Lecture, affichage et sauvegarde des données

Centre universitaire Nour Bachir El-Bayadh

Institut des sciences

Département de technologie

Spécialité : HYD, GC, ETT, ELN et TLC

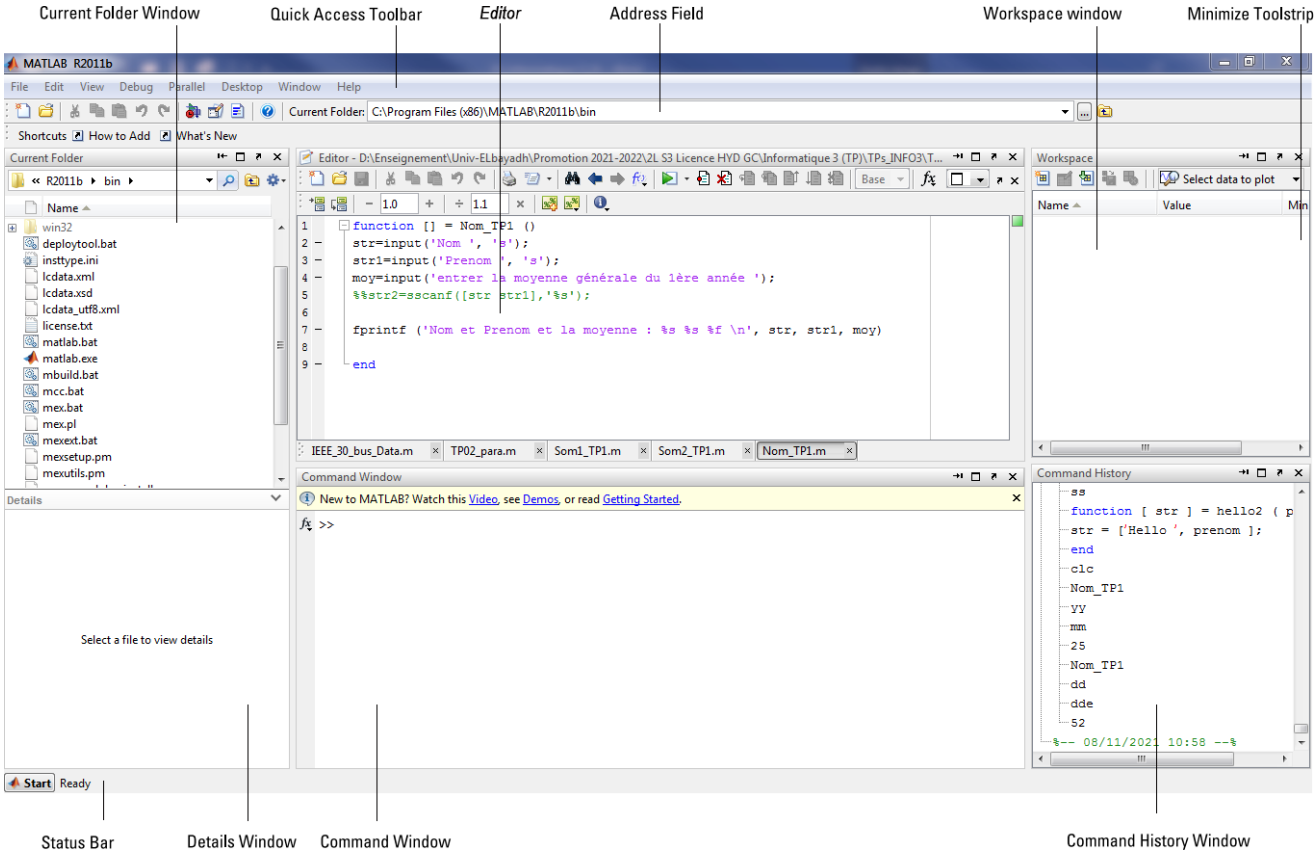
Niveau : Licence 02 (semestre 03)

Travaux Pratiques Informatique 3

TP 02 : Lecture, affichage et sauvegarde des données

L'objectif de TP :

Ce TP01 et TP02 a pour but de vous familiariser avec l'interface et les bases de l'environnement MATLAB afin d'utiliser quelques fonctions de base pour la lecture, l'affichage et le sauvegarde des données.

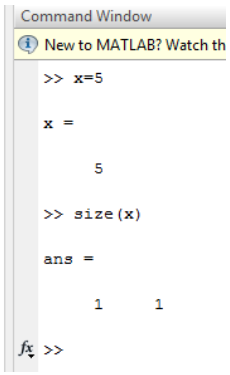
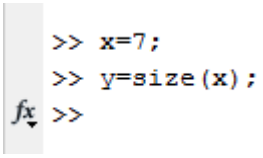


Partie 01 : Partie théorique (l'interface de Matlab)

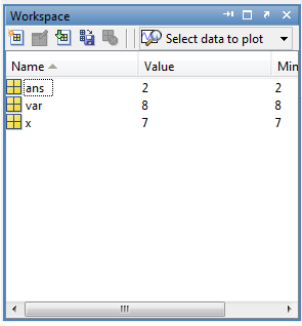
Nous allons nous familiariser avec l'interface de Matlab. Selon la version utilisée, l'interface peut changer légèrement mais les points centraux resteront identiques.

Command Windows

Une **variable** scalaire (**x**=5 : ***affectation de la valeur 5 à la variable x***) est vue par MATLAB comme une matrice de dimension 1x1 (ligne x colonne). Comme le montre l'exemple suivant dans lequel on affecte à la variable x la valeur 5 et on demande ensuite ses dimensions par la fonction ***size*** (***x***) dans l'espace de **Command Window:**

>> x=5 >> size(x)	Utilisation de point-virgule	*ans* (answer)
	Afin d'éviter l'affichage des résultats, il suffit de suivre la commande par un point-virgule (;). 	Matlab définit une variable **ans**, elle est une matrice de taille 1x1 (une ligne par une colonne).

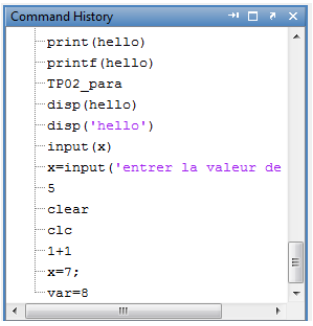
Une *commande* ***clc*** : Cette commande permet de vider l'écran de **Command Window** pour voir une fenêtre propre. Une *commande* ***clear*** : Cette commande permet **de supprimer toutes les variables** dans l'espace de **Workspace** pour repartir sur une fenêtre propre.



Workspace (go to Window -> Workspace)

Dans cette fenêtre, on obtient la liste des variables connues par Matlab. Il est possible de double-cliquer sur une variable pour l'afficher. Un clic-droit sur les variables donne de nombreuses options telles que : Copiez, Collez, Supprimez etc.

Command History (go to Window -> Command History)



L'espace Command History conserve une trace de toutes les opérations qui ont été réalisées sur l'espace Command Window. On peut également remonter dans la liste de commandes en se plaçant dans la **Command Window** et en pressant les flèches de direction ↑ ↓.

Current Folder (go to Window -> Current Folder)

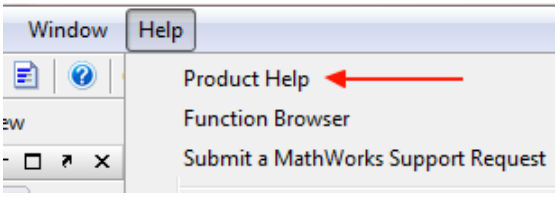
C'est le dossier qui contient le ficher de script, la programmation, le travail réaliser sur Matlab.

Help

L'espace de l'aide est essentielle lorsque l'on programme avec un langage de haut-niveau comme Matlab, où le nombre de fonctions est très important et la syntaxe est parfois complexe. Pour accéder à l'aide on peut au choix sélectionner une fonction et presser **F1** sur le clavier, ou bien taper dans l'espace **Command Window** : **help cos**, ***help input*** … etc. Il est essentiel que vous vous familiarisiez avec les outils de l'aide de Matlab pour réussir dans ce Cours/TP.

Command	***Description***
helpwin	ouvre une fenêtre contenant la liste des commandes Matlab ainsi que leurs documentations
help	donne la liste de toutes les commandes par thèmes
help nom	décrit la fonction nom.m
lookfor nom	Recherche une instruction à partir du mot clé nom

Use: help → Product Help to display the help Window



Script

Le script est le fichier avec l'extension '**.m**' qui contient le programme plus simplement. Il s'agit d'une liste de commandes et des **fonctions**.

Fonction

Une fonction va permettre de rentrer des arguments en entrée et d'obtenir différentes variables en sortie.

Editor

La plupart de votre travail sous Matlab va consister à créer ou modifier des fichiers avec l'extension « **.m** » qui définit les fichiers de Matlab. Lorsque l'on réalise une tache sous Matlab, il est très souvent possible de le faire en utilisant uniquement la **Command Window**. Cependant lorsque cette tache devient plus complexe (plusieurs dizaines de ligne de code) ou que l'on souhaite pouvoir la transmettre à quelqu'un d'autre simplement, on utilise la fenêtre **Editor**. On crée un fichier .m qui peut être au choix un **script** ou une fonction.

Partie 02 : Partie Simulation (MATLAB – SIMULINK)

Pour commencer on fixe le **USERPATH**, le dossier qui contient votre fichier (.m)

- Crée un dossier dans le Bureau avec le nom : **TP_INFO3**.
- Crée un autre dossier dans le dossier précédant avec le nom : **TP01_02_INFO3**.
- Dans l'espace **Current Folder**, choisir le dossier TP01_02_INFO3 pour le 1er TP et ainsi de suite pour les autres TPs.

Exercice1 : "Hello World"

Il s'agit d'un bref programme pour mettre en place les différents éléments nécessaires.

- Crée ensuite **new script** (.m) dans le dossier **TP01_INFO3**.
- Ecrire dans la 1ère ligne :

```
1. varibale= 'Hello world';
```

Run

- Sauvegarder le fichier avec le nom hello.m.

Run

Conclure !

Exercice2: " Somme de deux nombres "

On va essayer ici de réaliser un script pour en faire une **fonction** qui prend deux nombres en entrée et retourne la somme des deux.

- Cree un fichier **new script** (Ctrl +N).
- Réaliser un programme qui va calculer la somme de deux nombres.
- Sauvegarder avec le nom du fichier : **Som_xy_TP1.m (utiliser le tiret de 8 « _ » et non « - »)**

```
1 -     f=3;
2 -     d=4;
3 -     r=f+d;
```

Que remarquez-vous ?

Exercice3: " Somme de deux nombres quelconque "

On veut une **fonction** qui prend deux nombres aléatoire en entrée et renvoie par affichage la somme des deux.

- Cree un fichier **new script** (Ctrl +N). Nommer : **Som1_TP1.m**, Sauvegarder !!
- Ecrire la fonction :

```
function [z] = Som1_TP1 (x,y)
z=x+y
end
```

- Run
- Une erreur s'affiche :

```
Error using Som1_TP1 (line 2)
Not enough input arguments.
```

```
>> Som1_TP1 (4,5)

z =

     9


ans =

     9
```

On a défini une fonction **Som1_TP1**(x,y) avec deux arguments, mais on a appelé la fonction sans arguments ; alors pour corriger cette erreur il faut donner une valeur à x et y, on l'appelle à la fonction Som1_TP1 dans l'espace **Command Window** par : On remarque que Matlab a affiché le variable ***z*** puisque nous n'avons pas utiliser le point-virgule et la variable par défaut ***ans***.

Exercice4 : " Comment trouver l'information voulue "

Utiliser l'espace **Command Window**

- help log
- help mod

Exercice5: " Somme de deux nombres quelconque par choix"

On veut réaliser un programme qui prend trois chiffres par choix en entrée et renvoie un affichage de produit des premiers chiffres et la racine de produit des derniers nombres.

- Cree un fichier **new script** (Ctrl +N), nommer : **Som2_TP1.m**, Sauvegarder !!
- Ecrire la fonction :

```
function [] = Som2_TP1()
x=input('entrer le 1er nombre ')
y=input('entrer le 2e nombre ');
z=input('entrer le 3e nombre ');
a=x*y;
disp('^_^');
fprintf('le produit des 1er nombres est: %f \n', a)
b=sqrt(y*z);
fprintf ('la racine des deux dernier nombres est: %.2f \n', b);
disp('^_^');
end
```

On a utilisé 4 fonctions « input (), sqrt(), dips() et fprintf() »,

- Utiliser le « **help input () »** pour découvrir ces fonctions.

Quelle est le but de ces fonctions ?

Partie 03 : Partie expérimentale (MATLAB – SIMULINK)

Exercice5: " Introduire Nom et Prénom "

On va essayer ici de réaliser un script pour en faire une **fonction** qui prend 2 arguments votre nom et prénom en entrée et retourne votre nom et prénom ***sur la même ligne*** en sortie.

- Cree un fichier **new script** (Ctrl +N). Nommer : **Name_TP1.m**, Sauvegarder !!

Exercice6: " Introduire Nom et Prénom + la moyenne générale du 1ère année"

On va réaliser un script pour une **fonction** qui prend 3 arguments votre nom et prénom et la moyenne générale de la 1ère année comme des entrées et renvoie votre nom et prénom ***et la moyenne sur la même ligne*** en sortie.

- Cree un fichier **new script** (Ctrl +N). Nommer : **Name1_TP1.m**, Sauvegarder !!

Fiche TP : 03
Fichiers script et Types de données et de variables

Centre universitaire Nour Bachir El-Bayadh
Institut des sciences
Département de technologie
Spécialité : HYD, GC, ETT, ELN et TLC
Niveau : Licence 02 (semestre 03)
Travaux Pratiques Informatique 3

TP 03 : Fichiers script et Types de données et de variables

L'objectif de TP :

Ce TP03 a pour but de vous découvrir les opérations de base ainsi les types de variables ainsi leurs utilisations dans la résolution des exercices.

Partie 01 : Partie théorique (Types de données et de variables)

Outils de base

On l'a déjà dit, le principe de base de Matlab est de considérer la plupart des objets comme des matrices. Ainsi les opérations usuelles $+, -, *, /$ doivent se comprendre comme des opérations matricielles.

On consacrera la section suivante à ces opérations. Nous allons dans un premier temps regarder ce qu'ils se passent pour les matrices 1x1 (c'est à dire un seul élément).

Types de données de variables

Il existe *cinq grands* types de variables sous Matlab :

- **NOMBRES (les entiers, les réels, les complexes)**,
- **CARACTERES (les chaînes de caractères)**,
- **LOGIQUE** (1 ou 0).
  - Définissons une variable de chaque type :

```
a = 3.14; b = 1+i; s = 'TP-InFo3';
d1 = true(1==1); d2 = logical(1);
e = int8(2);
```

La variable 'a' représente un réel, b un complexe, c une chaine de caractères, d1 et d2 sont deux manières de définir une variable logique (VRAI dans le cas présent) et e est un entier codé sur 8 bits. On peut alors vérifier le type de ces différentes variables en utilisant **la fonction *whos*** :

```
>> whos
  Name        Size              Bytes  Class     Attributes

  a           1x1                   8  double
  b           1x1                  16  double    complex
  d1          1x1                   1  logical
  d2          1x1                   1  logical
  e           1x1                   1  int8
  s           1x8                  16  char
```

  - Utiliser les fonctions $\boldsymbol{ischar(variables)}, \boldsymbol{islogical}(\ldots), \boldsymbol{isreal}(\ldots)$.
  - **Conclure !!**

Variables MATLAB

Les noms des variables et de fonctions sont composés de lettres et chiffres. Matlab fait la distinction entre les majuscules et minuscules, i.e. INFO3, info3 et Info3 sont des variables différentes. Si la variable existe déjà, le contenu est écrasé par une nouvelle valeur affectée à cette variable.

Variable :

Dans Matlab il y a un seul type de données: le type de matrice (matrix).

Matrice	Vecteur	Scalaire
m x n, m, n >1,	1 x n, n >1,	1 x 1

Nombres ou chaînes de caractères :

Nom de variable	Nombres	'1, 2, 3, ...'
chaînes de caractères	1, 2, (1+i)….	**chaînes de caractères**

Classes and Format

Par la syntaxe introduite ci-dessus, MATLAB définit des variables qui appartiennent à la classe ***double array***, c'est à dire des tableaux de **réels** qui peuvent correspondre à des scalaires, des vecteurs ou des matrices. Mise à part cette classe fondamentale, il faut signaler qu'il existe d'autres classes MATLAB prédéfinies. La plus importante est certainement ***char array***, à laquelle appartiennent les chaînes de caractères, définies en utilisant **'…'**.

Format and Codage

Le tableau suivant répertorie les caractères et les sous-types de conversion disponibles sur MATLAB.

Value Type	Conversion	Details
Integer, signed	`%d` or `%i`	Base 10
Integer, unsigned	`%u`	Base 10
	`%o`	Base 8 (octal)
	`%x`	Base 16 (hexadecimal), lowercase letters `a–f`
	`%X`	Same as `%x`, uppercase letters `A–F`
Floating-point number	`%f`	Fixed-point notation
	`%e`	Exponential notation, such as `3.141593e+00`
	`%E`	Same as `%e`, but uppercase, such as `3.141593E+00`
	`%g`	The more compact of `%e` or `%f`, with no trailing zeros
	`%G`	The more compact of `%E` or `%f`, with no trailing zeros

	`%bx` or `%bX`
	`%bo`
	`%bu`
		Double-precision hexadecimal, octal, or decimal value
		Example: `%bx` prints `pi` as `400921fb54442d18`

	`%tx` or `%tX`
	`%to`
	`%tu`
		Single-precision hexadecimal, octal, or decimal value
		Example: `%tx` prints `pi` as `40490fdb`

Characters	`%c`	Single character
	`%s`	String of characters

On va utiliser le codage des formats ci-après pour nos TPs :

Numbers	Format Conversion	Declaration de variable	Character	Format Conversion	Declaration de variable
les entiers	%d	**int** variable	Single	%c	**char** variable
les réels	%f	**float** variable	String	%s	

Classes	
Double / single	char

Format d'affichage

Par défaut, Matlab affiche les résultats sous forme décimale. Ce format peut être changé à tout moment avec la fonction ***format***

Commande	Affichage	Exemple
format short	décimal à 5 chiffres	*31.416*
format long	décimal à 16 chiffres	*31.41592653535879*
format bank	virgule fixe à 2 décimales	*31.41*
format rat	fractionnaire	*3550/113*

Arithmétique et opérations sur les scalaires

On a déjà réalisé les opérations de base avec des variables et des fonctions sur le TP02, vous pouvez tester les exemples suivants:

```
>> x = 2; y = 1.5 ;
somme = x+y
difference = x-y
produit = x*y
division = x/y

somme = 3.5000
difference = 0.5000
produit =  3
division = 1.3333
```

On peut également travailler sur Matlab comme une calculatrice numérique utilisant les fonctions trigonométriques, puissance, logarithmiques etc.

Main mathematical function used in Matlab

`exp(x)`	:	exponentielle de x	`conj(z)`	:	conjugué de z
`log(x)`	:	logarithme néperien de x	`abs(z)`	:	module de z
`log10(x)`	:	logarithme en base 10 de x	`angle(z)`	:	argument de z
`x^n`	:	x à la puissance n	`real(z)`	:	partie réelle de z
`sqrt(x)`	:	racine carrée de x	`imag(z)`	:	partie imaginaire de z
`abs(x)`	:	valeur absolue de x	`rem(m,n)`	:	reste de la division entière de m par n
`sign(x)`	:	1 si $x > 0$ et 0 si $x \leq 0$	`lcm(m,n)`	:	plus petit commun multiple de m et n
`sin(x)`	:	sinus de x	`gcd(m,n)`	:	plus grand commun diviseur de m et n
`cos(x)`	:	cosinus de x	`factor(n)`	:	décomposition en facteurs premiers de n
`tan(x)`	:	tangente de x	`round(x)`	:	entier le plus proche de x
`asin(x)`	:	sinus inverse de x (arcsin de x)	`floor(x)`	:	arrondi par défaut de x
`sinh(x)`	:	sinus hyperbolique de x	`ceil(x)`	:	arrondi par excés de x
`asinh(x)`	:	sinus hyperbolique inverse de x			

Partie 02 : Partie Simulation (MATLAB – SIMULINK)

- Crée un nouveau dossier dans le dossier **TP_INFO3** avec le nom : **TP03_INFO3**.
- Change Current folder (path) to **TP03_INFO3** in Matlab.

Exercice 01

Créer un programme permet de réaliser la somme de 1 à 10. ($\boldsymbol{S = 1 + 2 + \cdots + 10}$)

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **Som_TP3.m**

Exercice 02

Chercher un programme permet de réaliser le produit de 3 à 7. ($P = 3 * 4 * \ldots * 7$)

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **Prod_TP3.m**

Exercice 03

Créer un programme permet de calculer la racine carré d'un nombre.

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **Rac_TP3.m**

Exercice 04

On cherche un programme permet de calculer le **déterminant** $\Delta = b^2 - 4ac$ d'un polynôme d'ordre 2.

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **Det_TP3.m**

Exercice 05

Réaliser une fonction permet de calculer les deux solutions d'une équation d'un polynôme du 2ème ordre à partir des 3 coefficients (a, b, c).

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **2ordr_TP3.m**

Soit le polynôme d'ordre 2 suivant : $P(x) = x^2 + 8x + 16$, les coefficients de polynôme sont : P=[1, 8, 16]

Dans le cas général un polynôme d'ordre 2 s'écrit : $P(x) = ax^2 + bx + c$ avec P=[a, b, c] et les racines sont : $P(x) = (x - r_1)(x - r_2)$ avec $r_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

On cherche à utiliser une fonction Matlab permet de résoudre le polynôme d'ordre 2 :

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **2ordr1_TP3.m**
- Utiliser les mêmes coefficients proposées dans la partie précédente,
- Utiliser la fonction ***roots***() pour résoudre l'équation du polynôme.

```
1 -    P1=[1, 8, 16];
2 -    r1=roots(P1);
3 -    r1
```

➔

```
r1 =

    -4
    -4
```

- Comparer les résultats obtenus avec vos résultats du programme, Conclure !
- Utiliser la fonction ***poly***() pour construire les coefficients d'un polynôme à partir de ces racines

```
5 -    P2=poly(r1);
6 -    P2
```

➔

```
P2 =

     1     8    16
```

Exercice 06

Soit le polynôme suivant :

$$P(x) = x^7 - 37x^6 + 555x^5 - 4295x^4 + 17924x^3 - 37668x^2 + 30240x$$

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **Pol_TP3.m**

En utilisant Matlab ;

- Résoudre $P(x) = 0$.
- Calculer le polynôme $P(x)$ pour $x = 2, x = 3$. (utiliser la fonction ***polyval*** $(\boldsymbol{p}, \boldsymbol{x})$)

Exercice 07

Soit les racines d'un polynôme :

$$R = [7, 8, 9{,}5, 0{,}2, 6]$$

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **RacPol_TP3.m**

En utilisant Matlab ;

- Chercher les coefficients du polynôme.
- Quelle est l'ordre du polynôme.
- Calcul la dérivée de la fonction polynomiale (utiliser la fonction ***polyder*** $(\boldsymbol{p})$)
- Calcul la primitive de la fonction polynomiale (utiliser la fonction ***polyint*** $(\boldsymbol{p})$)

Exercice 08

Soit les polynômes suivants :

$$P_1(x) = x^3 + 3x^2 - 24x - 80 \quad \text{et} \quad P_2(x) = x^2 - x - 20 \text{ et } P_3(x) = x + 4$$

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **divPol_TP3.m**
- Chercher les coefficients des polynômes.

En utilisant Matlab ;

- calculer les racines de $P_1(x)$ et $P_2(x)$.
- Calculer le produit de convolution $h(x) = P_2(x) * P_3(x)$.(utiliser la fonction ***conv***$(\boldsymbol{P2}, \boldsymbol{P3})$)

- Calculer le produit de déconvolution $h(x) = P_2(x) * P_3(x)$.(utiliser la fonction $\boldsymbol{deconv(P1, P3)}$)
- Conclure

Exercice 09

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **nbrcmplx_TP3.m**

Entrer les nombres complexes suivants :

$$z_1 = 1 + i\ , z_2 = z^2\ , z_3 = e^{i\pi/4}$$

Cependant, i et j sont des variables MATLAB réservées pour l'indice des nombres complexes.

- Calculer par Matlab les quantités suivantes :
  - La partie réelle des nombres complexes, affecter chaque partie à une variable.
  - La partie imaginaire des nombres complexes, affecter chaque partie à une variable.
  - Le conjugue de Z_1 *et* Z_3.
  - Le module de chaque nombre et affecter à une autre variable.
  - L'argument de chaque nombre complexe.

Utiliser les fonctions complexes dans le tableau **Main mathematical function used in Matlab.**

Partie 03 : Partie expérimentale (MATLAB – SIMULINK)

Exercice 10 de préparation (pour l'enseignement en ligne)

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **EX10_TP3.m**

1. **Evaluez les quantités suivantes dans Matlab avec 5 chiffres significatifs :**

   (a) tanh(e) (b) $\log_{10}(2)$ (c) $\frac{1}{1+\frac{1}{1+\frac{1}{2}}}$

   (d) PGCD(48972, 36533112) (e) $e^{10^{2.7}}$ (f) $\cos^{-1}(\frac{\pi}{4})$

   (g) PPCM (318,732) (h) $\sqrt{3\pi}$ (i) $\ln\frac{1}{\pi}$.

2. **Calculez $\frac{100}{8} - 0.5\left[\frac{4}{10}\right]^2$. Commentez le résultat.**
3. **Calculez e^{14} *et* 382801π jusqu'au 15ème chiffres significatifs. Quel est le plus grand ?**
4. **Comparez la division à droite '/' et la division à gauche '\'. Commentez.** (Utiliser ALTGR + 8 pour '\')
5. **Trouvez la partie réelle et imaginaire des nombres complexes suivants :**

   $$e^{i(3\pi+4)} \text{ et } \frac{1}{1+i} \text{ et } \ln(-1)$$

6. **Calculez le module et l'argument des nombres complexes suivants :**

   (a) $e^{i(3\pi+4)}$ (b) $\frac{1}{1+i}$ (c) $\ln(-1)$.

   (d) $3+7i$ (e) i^3+1 (f) $e^{i\frac{\pi}{5}}$.

Exercice 11

- Créer **new script** (Ctrl +N). Sauvegarder avec le nom du fichier : **EX11_TP3.m**

Soit le polynôme suivant :

$$P(x) = x^5 - 2x^4 - 8x^3 + 16x^2 + 16x - 32$$

- Evaluer les valeurs de $P(x)$ pour les points $x = 0, x = 1, x = 2$.
- Calculer la dérivée de $P(x)$
- Calculer la primitive de $P(x)$
- Calculer le polynôme d'ordre 2 $G(x)$ tel que $G(x) = \frac{P(x)}{x^3-6x^2+12x-8}$

Fiche TP : 04
Vecteurs et matrices (MATLAB)

Centre universitaire Nour Bachir El-Bayadh
Institut des sciences
Département de technologie
Spécialité : HYD, GC, ETT, ELN et TLC
Niveau : Licence 02 (semestre 03)
Travaux Pratiques Informatique 3

TP 04 : Vecteurs et matrices

L'objectif de TP :

Ce TP04 a pour but de vous découvrir les opérations sur les vecteurs et les matrices afin de les utiliser pour la résolution des exercices.

Partie 01 : Partie théorique (Vecteurs : Listes et tableaux)

Matlab utilise principalement les listes (vecteurs) ou tableaux (matrices) pour le calcul. Il est utile de savoir manipuler ces objets des maintenant.

Pour Matlab, une variable (a; x; . . .) ou une liste de nombres est un tableau particulier.

On peut retenir, pour simplifier, que dans Matlab tout est matrice (une matrice est un tableau). Si cela peut paraître bizarre au début, c'est ce qui permet à Matlab d'être aussi puissant et rapide en calcul.

Construction d'un Vecteurs (une liste)

On peut définir une liste de nombres en donnant à la suite ses éléments sépares par des espaces ou des virgules. La liste est délimitée par des crochets [].

La différence entre virgule, espace et point-virgule	

`>> P=[1, 8, 16]`
`P=[1 8 16]`
 	
`P =`
 	
`1 8 16`
	`>> P=[1; 8; 16;]`
	
	`P =`
	
	`1`
	`8`
	`16`

Vecteur à 3 éléments **horizontales**	Vecteur à 3 éléments **verticales**

La variable ***P*** est considérée comme un tableau comportant une ligne et trois colonnes :

1	8	16

Transposée d'un vecteur

La transposée pour passer d'une ligne à une colonne ou réciproquement :

```
>> vec1= [1 4 7]
>> vec1'
```

```
vec1 =

     1     4     7

>> vec1'

ans =

     1
     4
     7
```

Manipuler un vecteur

Accès aux éléments d'une liste

On peut extraire les éléments d'un vecteur. On accède l'élément **d'indice k** de la liste a avec $\boldsymbol{P(k)}$.

On cherche par exemple la 3ème valeur de vecteur **P** : $P(3)$

```
>> P(3)

ans =

    16
```

Exemple1

Soit : $\boldsymbol{P_2(x) = x^5 - 2x^4 - 8x^3 + 16x^2 + 16x - 32}$

Le vecteur des coefficients du polynôme $P_2(x)$ est définit par : P2=[1 -2 -8 16 -32]

```
>> P2=[1 -2 -8 16 -32]

P2 =

     1    -2    -8    16   -32
```

On cherche par exemple la 2ème valeur de vecteur de **P2** : $P2(2)$

```
>> P2(2)

ans =

    -2
```

On cherche les 3 premiers valeurs de vecteur de **P2** : $P2(1:3)$

```
>> P2(1:3)

ans =

     1    -2    -8
```

On cherche les valeurs des indices impaire de vecteur de **P2** : $P2(1:2:5)$

```
>> P2(1:2:5)

ans =

     1    -8   -32
```

L'expression $(\boldsymbol{a : p : b})$ crée une liste dont les éléments vont de $\boldsymbol{a}$ à $\boldsymbol{b}$ avec un pas de $\boldsymbol{p}$.

Lorsqu'on ne donne pas le pas, la valeur du pas est par défaut 1. L'accès à un élément d'indice négatif conduit à une erreur.

La taille d'un vecteur

La commande $\boldsymbol{length}()$ permet de retourner le nombre d'élément dans un vecteur.

```
>> length(P2)

ans =

     5
```

Concaténer deux vecteurs

P3= [P2 vec1]

Command linspace ()

linspace(***X1***, ***X2***, ***N***) generates: **N points between X1 and X2**, for N = 1, linspace returns X2.

$P4 = linspace(1,10,10)$

```
>> P4=linspace(1,10,10)

P4 =

     1     2     3     4     5     6     7     8     9    10
```

Construction d'une matrice (un tableau)

On peut créer des tableaux de nombres avec plusieurs lignes en donnant chaque ligne séparée par un point-virgule (;).

Soit : $P3 = [1\ -2\ -8\ 16\ -32\ ;\ 1\ 2\ 3\ 4\ 5]$

```
>> P3=[1 -2 -8 16 -32 ; 1 2 3 4 5]

P3 =

     1    -2    -8    16   -32
     1     2     3     4     5
```

```
>> P3=[1 -2 -8 16 -32 ; 1 2 3 4 5]

P3 =
          colonne  colonne              colonne
             1        2                    5
ligne 1      1       -2       -8    16    -32
ligne 2      1        2        3     4      5
```

Accès aux éléments d'une matrice (un tableau)

De même, on peut extraire des parties d'une matrice. Le premier nombre désigne la ligne et le deuxième nombre la colonne.

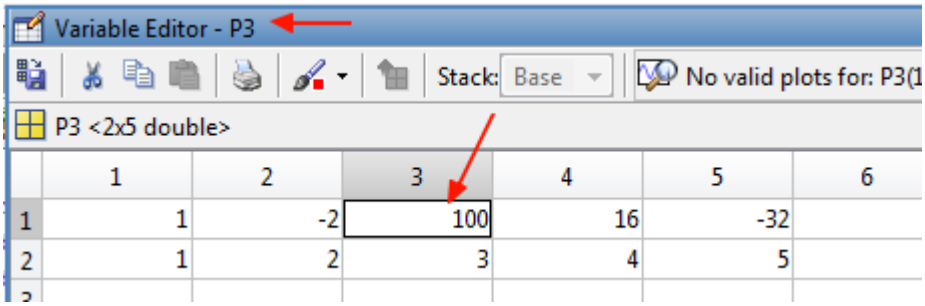
Exemple2

- Chercher la valeur d'élément (1,3) de la matrice P3 : ***P3*** (**1** , **3**)
- Chercher les valeurs de la 4ème colonne (**X**,4) de la matrice P3 : ***P3*** (: , **3**)
- Chercher les valeurs de la 2ème ligne (**2,** X) de la matrice P3 : ***P3*** (**2**, **X**)
- Chercher les valeurs de la 1ère ligne à partir du 3ème colonne de la matrice P3 : ***P3*** (**1**, **3**: **5**)

Changement d'un élément dans la matrice

- On veut modifier la valeur d'élément (1,3) par la valeur 100 : P3(1,3) =100.

- Autrement, dans l'espace **Workspace**, double-clic sur la matrice P3  (variable Editor), chercher l'élément (ligne 1, colonne 3), changer le par la valeur 100.



	1	2	3	4	5	6
1	1	-2	100	16	-32	
2	1	2	3	4	5	

Partie 02 : Partie Simulation (Opération sur les Matrices (tableaux))

- Crée un nouveau dossier dans le dossier **TP_INFO3** avec le nom : **TP04_INFO3**.
- Change Current folder (userpath) to **TP04_INFO3** in Matlab.

Toutes les opérations usuelles entre matrices et vecteurs sont disponibles (addition, multiplication, transposition etc.). Matlab permet aussi d'appliquer les principaux opérateurs éléments par éléments, il suffit en général de faire précéder l'opérateur voulu par un point " . "

Soit les deux matrices A et B suivantes :

$$A = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix},\ A1 = \begin{pmatrix} 4 & 1 & 2 \\ 6 & 1 & 6 \\ 3 & 3 & 1 \end{pmatrix},\ B = \begin{pmatrix} 8 & 1 \\ 5 & 7 \end{pmatrix},\ V_1 = \begin{pmatrix} 5 \\ 4 \\ 6 \end{pmatrix},\ V_2 = (2 \quad 1 \quad 6), V_3 = (4 \quad 9),$$

Exercice 1 (Concaténer, Comparer les matrices)

En utilisant Matlab (espace Command Window);

- La matrice C comprend la matrice A et A1 **horizontalement** : C=[A, A1]
- Utiliser la fonction $\boldsymbol{cat(dim, A, A1)}$ pour assembler les matrices A et A1 : C1=cat(2, A, A1)
- Comparer la matrice C et C1, utiliser la fonction : $\boldsymbol{isequal(C, C1)}$
- Comparer la matrice A et A1, use : **A==A1**
- Conclure !!

Exercice 2 (Ajouter – Supprimer (ligne/colonne) dans une matrice)

En utilisant Matlab (espace Command Window);

- Calculer la dimension de la matrice A et B.
- Calculer le nombre des éléments dans le vecteur V1 et V2.
- Ajouter le vecteur V1 dans la 3éme colonne de matrice B.

Méthode 1	Méthode 2

```
>> B(1,3)=V1(1);
>> B(2,3)=V1(2);
>> B(3,3)=V1(3);
>> B

B =

     8     1     5
     5     7     4
     0     0     6
```

```
>> B=[8 1; 5 7];
>> B(3,1)=0;
>> B=[B, V1]

B =

     8     1     5
     5     7     4
     0     0     6
```

- Ajouter le vecteur V2 dans la 3éme ligne de matrice B.

```
>> B(3,:)=V2

B =

     8     1     5
     5     7     4
     2     1     6
```

- Supprimer la deuxième ligne de matrice A et affecter la nouvelle matrice au C3.

Méthode 1	Méthode 2

`>> C1=[A(1,:)];`
`>> C2=[A(3,:)];`
`>> C3=[C1; C2]`
 	
`C3 =`
 	
`8 1 6`
`4 9 2`
	`>> A(2,:)=[]`
	
	`A =`
	
	`8 1 6`
	`4 9 2`
	
	`>> C3=A;`

Exercise 3 (Summation - product – transposition – square matrix)

- Create **a new script** (Ctrl +N). Save with the name : **EX3_TP4.m**
- Introduire les matrices dans le script.
- Vérifier les matrices par l'exécution de programme  .
- Calculer les dimensions de $A, A1, B, V1, V2\ et\ V3$.
- Calculer la matrice transposée de A et B : **transpose** (A), B'.
- Calculer la matrice carrée de B : **B^2**.
- A partir de la matrice A, extraire la sous matrice $A2 = (3{,}5\ ;\ 4{,}9)$
- Calculer la somme des matrices A et A1 : $Som = A + A1$;
- Calculer la somme des matrices B et A2 : $S = B + A2$;
- Calculer la somme des matrices A et C3 : $Som1 = A + C3$;

```
>> Som1=A+C3
Error using  +
Matrix dimensions must agree.
```

Pour calculer la somme des deux matrices, les dimensions des matrices doivent être les mêmes.

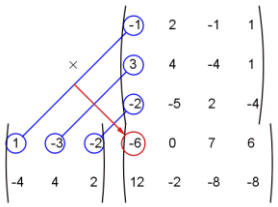
- Calculer le produit des matrices A et V1 : $prd = A * V1$;

- Vérifier manuellement le résultat.

```
>> prd=A*V1

prd =

    80
    77
    68
```



Pour le produit il faut que la dimension de la colonne de la première matrice égale à la dimension de la ligne de la deuxième matrice $(n, m) * (n', m') \gg m = n'$

Exercice 4 (matrices particulières)

En utilisant Matlab (espace Command Window);

- Matrice d'identité : utiliser la commande **eye(n,m) : 1) (n=3,m=1), 2) (n=3,m=2), 3) (n=3,m=3).**
- Matrice nulle : utiliser la commande **zeros(n,m) : 1) (n=3,m=1), 2) (n=3,m=2), 3) (n=3,m=3).**
- Matrice unitaire : utiliser la commande **ones(n,m) : 1) (n=3,m=1), 2) (n=3,m=2), 3) (n=3,m=3).**
- Matrice aléatoire : utiliser la commande **rand(n,m) : 1) (n=3,m=1), 2) (n=3,m=2), 3) (n=3,m=3).**
- Matrice Magique : utiliser la commande **magic(n).**

Exercice 5 (introduire une matrice (2,2))

- Create **a new script** (Ctrl +N). Save with the name : **EX5_TP4.m**
- Réaliser une fonction qui prend 4 nombres en entrées et renvoie en sortie une matrice $M(2,2)$.
  Utiliser la commande **zeros (n,m)** pour initier la matrice $M(2,2)$.

Exercice 6 (introduire une matrice (2,2) par l'utilisateur)

- Create **a new script** (Ctrl +N). Save with the name : **EX6_TP4.m**
- Réaliser une fonction qui prend 4 nombres en entrées par l'utilisateur et renvoie en sortie une matrice $M1(2,2)$. $M1 = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}$
  Utiliser la commande $m_{11} = \boldsymbol{input}$ ('$entre\ le\ 1er\ nombre$ ') ;
  Utiliser la commande $\boldsymbol{zeros}(\boldsymbol{n}, \boldsymbol{m})$ pour initier la matrice $M1(2,2)$.

Exercice 7 (introduire une matrice (3,3) par l'utilisateur)

- Create **a new script** (Ctrl +N). Save with the name : **EX7_TP4.m**
- Réaliser une fonction qui prend 9 nombres en entrées par l'utilisateur et renvoie en sortie une matrice $M2(3,3)$.
  Utiliser la commande vera11=**input** ('entre le 1er nombre ') ;
  Utiliser la commande $\boldsymbol{zeros}(\boldsymbol{n}, \boldsymbol{m})$ pour initier la matrice $M2(3,3)$.

Exercice 8 (Sommation Matrice(2,2) + Matrice(2,2) par choix)

- Create **a new script** (Ctrl +N). Save with the name : **EX8_TP4.m**
- On cherche à réaliser une fonction qui faire la sommation entre deux $matrice\ A\ +\ B$.

Exercice 9 (Multiplication Matrice(3,3) x Vecteur(3,1) par choix)

- Create **a new script** (Ctrl +N). Save with the name : **EX9_TP4.m**
- On cherche de réaliser une fonction qui faire le produit entre matrice(3,3) et un vecteur(3,1) entrée par l'utilisateur et affiche en sortie le résultat.
  Utiliser la commande b11=**input** ('entre le 1er nombre ') ;
  Utiliser la commande $\boldsymbol{zeros(n,m)}$ pour initier la matrice $M2(3,3)$.

Exercice 10 (résoudre d'un système linéaire matricielle AX=Y)

- Create **a new script** (Ctrl +N). Save with the name : **EX10_TP4.m**
- Réaliser une fonction qui résoudre l'équation matricielle $AX = Y$ et affiche en sortie le résultat du X, avec la matrice A(3,3), les vecteurs Y(3,1), X(3,1).
- Mesurer **le temps d'exécution** du programme en seconde ; use command **tic ; and toc** ;
  Utiliser la commande a11=**input** ('entre le 1er nombre ') ;
  Utiliser la commande $\boldsymbol{zeros(n,m)}$ pour initier la matrice $A(3,3)$.
  Utiliser la commande $\boldsymbol{zeros(n,m)}$ pour initier le vecteur $X(3,1)$.
  Utiliser la commande $\boldsymbol{zeros(n,m)}$ pour initier le vecteur $Y(3,1)$.

Partie 03 : Partie expérimentale (MATLAB – SIMULINK)

Exercice 11 de préparation (déterminant d'une matrice)

- Create **a new script** (Ctrl +N). Save with the name : **EX11_TP4.m**
- Réaliser une fonction qui renvoie le déterminant d'une matrice $\boldsymbol{N(2,2)}$ donnée par l'utilisateur.
  Utiliser la commande var11=**input** ('entre le 1er nombre ') ;
  Utiliser la commande $\boldsymbol{zeros(n,m)}$ pour initier la matrice $N(2,2)$.

Exercice 12 de préparation (inversement d'une matrice (2,2))

- Create **a new script** (Ctrl +N). Save with the name : **EX12_TP4.m**
- Réaliser une fonction qui inverse une matrice $\boldsymbol{L(2,2)}$ donnée par l'utilisateur.
  Utiliser la commande a11=**input** ('entre le 1er nombre ') ;
  Utiliser la commande $\boldsymbol{zeros(n,m)}$ pour initier la matrice $L(2,2)$.

Exercice 12 de préparation (inversement d'une matrice(3,3))

- Create **a new script** (Ctrl +N). Save with the name : **EX12_TP4.m**
- Réaliser une fonction qui inverse une matrice $\boldsymbol{G(3,3)}$ donnée par l'utilisateur.
  Utiliser la commande G11=**input** ('entre le 1er nombre ') ;
  Utiliser la commande $\boldsymbol{zeros(n,m)}$ pour initier la matrice $G(3,3)$.

Fiche TP : 05
Instructions de contrôle (Instructions: if & switch)

Centre universitaire Nour Bachir El-Bayadh
Institut des sciences
Département de technologie
Spécialité : HYD, GC, ETT, ELN et TLC
Niveau : Licence 02 (semestre 03)
Travaux Pratiques Informatique 3

TP 05: Instructions de contrôle (Instructions: if & switch)

L'objectif de TP :

Ce TP05 a pour but de vous découvrir les instructions et les conditions de contrôle. La structure conditionnelle permet d'exécuter une action si et seulement si un test préalable et renvoie la valeur **'VRAI'**.

Partie 01 : Partie théorique (les Instructions conditionnées: if & switch)

Il existe deux commandes possibles permettant de réaliser des tests de conditions sur les données.

L'instruction *if*() permet de tester la valeur d'une variable et d'effectuer différents traitements suivant les cas testés.

L'instruction *switch*() permet de choisir entre différents cas, et de faire correspondre un traitement adapté à chacun des cas reconnus.

Les opérateurs de comparaison et les opérateurs logiques sont utilisés essentiellement dans les instructions de contrôle :

Opérateurs de comparaison (relationnels)	**Opérateurs logiques**
➢ Strictement supérieur : (X > Y) ➢ Strictement inférieur : (X< Y) ➢ Comparer deux objets : (X= = Y) ➢ Supérieur ou égale : (X>= Y) ➢ Inférieur ou égale : (X<= Y) ➢ Différent de : (X~=Y)	➢ And : & : (X &Y) ➢ Or : | : (X | Y) ➢ Not : X (-X)

L'instruction *If* … *elseif* … *else* … *end*

Exécuter instruction ou un groupe des instructions si la condition est vraie.

elseif & ***else*** : sont facultatifs et n'exécutent les instructions que lorsque les expressions précédentes du bloc ***if*** sont fausses.

Un bloc ***if*** peut inclure plusieurs instructions ***elseif***.

Syntax

```
if expression
   statements
elseif expression
   statements
else
   statements
end
```

- *Expression logique* est une expression dont le résultat peut être vrai ou faux (donnée logique);
- *Séquence d'instructions (statements)* est le traitement à effectuer si *l'expression logique* est vraie.

Description

Les expressions peuvent inclure des opérateurs relationnels (tels que < ou ==) et des opérateurs logiques (tels que &&, || ou ~). MATLAB évalue les expressions composées de gauche à droite, en respectant les règles de priorité des opérateurs.

- Vous pouvez utiliser n'importe quel nombre **d'instructions if**, chaque instruction if nécessite **end.**
- Évitez d'ajouter un espace dans **elseif** (***else if***). L'espace crée une nouvelle instruction if imbriquée qui requiert son propre mot-clé de fin.

Example 1 (Instructions conditionnées if-else-end)

- Create a new folder with name **TP05_INFO3** in folder **TP_INFO3**.
- Change Current folder (userpath) to **TP05_INFO3** in Matlab.
- Create **a new script** (Ctrl +N). Save with the name : **Exemple1_TP5.m**
- Affecter les valeurs suivantes (5,1,10/2) aux variables (a,b,c) successives:

Description :	**Script :**
Les lignes (1,2,3) : Initiation • **Initiation des valeurs de variables.** **Les lignes (5 à 8) : teste** • **L'instruction if** : teste la variable a est différente de c (***valeur !!***) Si la valeur de $\boldsymbol{a} = 5$ est différente de $\boldsymbol{c} = \frac{10}{2} = 5$ **Alors** calculer : la nouvelle valeur de $\boldsymbol{b} = (\boldsymbol{ancien\ b} = \mathbf{1}) + \mathbf{1} = \mathbf{1} + \mathbf{1} = \mathbf{2}$ **disp('b=')** : afficher dans l'espace *CW* l'expression : **b=** **disp(b)** : afficher la valeur de b : 2 ; **Les lignes (9 à 14) : contre le teste** • **L'instruction else** : ''teste la variable a est égale à c'' Non la valeur de $\boldsymbol{a} = \mathbf{5}$ est égal de $\boldsymbol{c} = \frac{\mathbf{10}}{\mathbf{2}} = \mathbf{5}$ **Alors** : **disp(a), disp(c)** : afficher dans l'espace *CW* l'expression la valeur de la variable a et c. **b=b-1 : calcule de la nouvelle valeur de b et affecter à b** **disp('b=')** : afficher dans l'espace *CW* l'expression : **b=**	1 – a=5; 2 – b=1; 3 – c=10/2; 4 5 – if a~=c 6 – b=b+1; 7 – disp('b= '); 8 – disp(b); 9 – else 10 – disp(a) 11 – disp(c) 12 – b=b-1; 13 – disp('b= '); 14 – disp(b) 15 – end

disp(b) : afficher la valeur de b : 0 ; **La ligne (15) : fin** • **L'instruction end** : termine et ferme la condition if.	

Example 2 (Instructions conditionnées if-elseif-end)

- Create **a new script** (Ctrl +N). Save with the name : **Exemple2_TP5.m**
- On cherche une fonction qui affiche une matrice carrée de taille **n ; soit zeros(), ones() ou bien rand() selon les entrée 1 ou 2 ou 3.**
- Utiliser l'instruction ***if***.

```
1   function [M]= Exemple2_TP5 ()
2   n=input('entrer la taille de la matrice ');
3   var=input('entrer une valeur pour la matrice : 1=zeros, 2=ones, 3=rand ');
4
5   if var == 1
6       M = zeros(n);
7   elseif var == 2
8       M = ones(n);
9   elseif var == 3
10      M = rand(n);
11  else
12   error('numero d''exemple non prevu ...');
13  end
14
15  end
```

L'instruction *switch*() … *case* … *otherwise* … *end*

L'instruction **switch** exécute conditionnellement un ensemble d'instructions parmi plusieurs choix nombres, chaînes, objets. Chaque choix est un cas.

Syntax

```
switch switch_expression
   case case_expression
      statements
   case case_expression
      statements
    :
   otherwise
      statements
end
```

Une '**switch_expression**' tester un scalaire ou une chaîne de caractères. Le bloc de commutation teste chaque cas jusqu'à ce que l'un des cas soit vrai.

- **If** un cas est vrai, MATLAB exécute les instructions correspondantes, puis quitte le bloc de commutation.
- **Sinon** (otherwise) s'exécute uniquement lorsqu'aucune casse n'est vraie.

Exemple 3 (Instructions conditionnées, Nombres : *switch … case … end*)

- Create **a new script** (Ctrl +N). Save with the name : **Exemple3_TP5.m**
- On cherche une fonction qui affiche une matrice carrée de taille **n soit zeros(), ones() ou bien rand() selon les entrée 1 ou 2 ou 3.**
- Utiliser l'instruction ***switch***.

```
function [M]= Exemple3_TP5 ()
n=input('entrer la taille de la matrice ');
var=input('entrer une valeur pour la matrice : 1=zeros, 2=ones, 3=rand ');
switch var
    case var==1
        M = zeros(n);
    case var == 2
        M = ones(n);
    case var == 3
        M = rand(n);
    otherwise
        error('numero d''exemple non prevu ...');
end
end
```

Partie 02 : Partie Simulation (Conditions)

Exercice 1 (teste d'une variable réelle : if)

- Create **a new script** (Ctrl +N). Save with the name : **EX1_TP5.m**
- Réaliser une fonction qui teste une variable réelle a si positive, nulle ou négative et renois par affichage.
- Utiliser l'instruction **if**.

Exercice 2 (teste d'une variable réelle : switch)

- Create **a new script** (Ctrl +N). Save with the name : **EX2_TP5.m**
- Réaliser une fonction qui teste une variable réelle a si positive, nulle ou négative et renvois par affichage.
- Utiliser l'instruction **switch**.

Exercice 3 (Enregistrement d'un fichier)

- Create **a new script** (Ctrl +N). Save with the name : **EX3_TP5.m**
- Réaliser une fonction qui affiche « **Voulez-vous enregistrer ? écrire Y ou N :** » et elle renvois par affichage l'enregistrement ou l'annulation du fichier.
- Utiliser l'instruction **if**.
- Réutiliser l'instruction **switch**.

```
function []=EX3_TP5()
car = input('Voulez-vous enregistrer  ? écrire Y ou N : ', 'S');

switch car
  case {'Y', 'y'}
    disp('le fichier est enregistré ');
  case {'N','n'}
    disp('l''enregistrement du ficheier est annulé ...');
end
end
```

Partie 03 : Partie expérimentale (MATLAB – SIMULINK)

Exercice 4 (teste d'inversement d'une matrice)

- Create **a new script** (Ctrl +N). Save with the name : **EX4_TP5.m**
- Create a random matrix of size 3. $\boldsymbol{Am = rand(3)}$;
- Réaliser une fonction qui renvoi l'inversement d'une matrice si possible ou bien erreur d'inversement si le déterminant d'une matrice égale à 0.
- Use instruction: $\boldsymbol{if}$
- Use instruction: $\boldsymbol{inv(matrix)}$
- Use instruction: $\boldsymbol{error}$('erreur d'inversement')
- Pour le teste : Utiliser la matrice carrée suivante : $M = [2, 4, 6; 3, -8, -5; 5, -4, 1]$.

Exercice 5 de préparation (Ajouter le teste d'inversement d'une matrice à l'EX12_TP4)

- Create **a new script** (Ctrl +N). Save with the name : **EX5_TP5.m**
- Copier **le script de l'exercice EX12_TP4.m** dans le script d'EX5_TP5.
- Modifier le script en ajoutant la fonction qui renvoi le teste d'inversement d'une matrice si possible ou bien erreur d'inversement si le déterminant d'une matrice égale à 0.
- Use instruction: **switch**.
- Pour le teste : Utiliser la matrice carrée suivante : $\boldsymbol{M = [2, 4, 6; 3, -8, -5; 5, -4, 1]}$.

Exercice 6 de préparation (Ajouter un test pour polynôme 2ème dégrée)

- Create **a new script** (Ctrl +N). Save with the name : **EX6_TP5.m**
- Copier **le script de l'exercice 2ordr_TP3.m** dans le script d'EX6_TP5.
- Créons un programme qui trouve les racines d'une équation de second degré ax^2 $+ bx + c = 0$ avec les conditions du discriminant.

Fiche TP : 06
Instructions de contrôle (Boucle: for & while)

Centre universitaire Nour Bachir El-Bayadh
Institut des sciences
Département de technologie
Spécialité : HYD, GC, ETT, ELN et TLC
Niveau : Licence 02 (semestre 03)
Travaux Pratiques Informatique 3

TP 06: Instructions de contrôle (Boucle: for & while)

L'objectif de TP :

Ce TP06 a pour but de vous découvrir les instructions de contrôle ainsi les opérations sur les boucles, cette dernière, elle permet de répéter la même commande à un grand nombre de fois en faisant varier un paramètre.

Partie 01 : Partie théorique (Boucle: for & while)

L'instruction : $for \ldots end$

L'instruction ***for*** (pour) s'exécute une/plusieurs instructions MATLAB dans une boucle de répétition avec une/plusieurs condition. C'est plus pratique pour la définition des suites par récurrences comme nous le verrons.

Corps de la boucle *for*

Syntaxe

```
for indice = borne_inf : borne_sup
Séquence d'instructions
end
```

- **Indice :** est une variable appelée l'indice de la boucle;
- **Borne_inf et Borne_sup :** sont deux constantes réelles (appelées paramètres de la boucle);
- **Séquence d'instructions** : est le traitement à effectuer pour les valeurs d'indices variant entre borne_inf et borne_sup avec un incrément de 1.

- Pour forcer une sortie immédiate de la boucle, utilisez une instruction ***break*** *ou* ***return***.
- Pour parcourir les valeurs d'un seul vecteur colonne, transposez-le d'abord pour créer un vecteur ligne.

Example 1

- Create **a new script** (Ctrl +N). Save with the name : **Exemple1_TP6.m**
- Réaliser une fonction Matlab qui compte de 1 jusqu'à 10.
- Use instruction: **for**.

Example 2

- Create **a new script** (Ctrl +N). Save with the name : **Exemple2_TP6.m**
- Use instruction: **for**.

➢ Réaliser une fonction qui affiche les nombres d'itérations de 0 jusqu'à n.	➢ Modifier la fonction d'afficher de $n/2$ jusqu'à n.

```
function []= Exemple2_TP6()

n=input('entrer n : ');
for i=0:n
    disp(i)
end
```

```
function []= Exemple2_TP6()

n=input('entrer n : ');
for i=n/2:n
    disp(i)
end
```

Example 3

- Create **a new script** (Ctrl +N). Save with the name : **Exemple3_TP6.m**
- Use instruction: **for**.

➢ On cherche une fonction qui calcule la somme de 1 à n. $S = 1 + 2 + 3 + 4 + \cdots + n$	➢ Modifier la fonction, On cherche de calculer la somme de a à b. $S_{ab} = a + 1 + 2 + \cdots + b$

```
function [] = Exemple3_TP6 ()

n = input('n= ');
% Initiation de S avec valeur 0
S = 0 ;
for i=1:n
S = S+i;
end
disp(S);
```

```
function [] = Exemple3_TP6 ()

a = input('a= ');
b = input('b= ');

% Initiation de S avec valeur a
S = a ;
for i=a+1:b
    S = S+i;
end
disp(S);
```

L'instruction : *while* … *end*

L'instruction ***while*** (tant que) exécute une/plusieurs instructions **tant que** la condition est vraie.

Corps de la boucle *while*

- **Expression logique** : est une expression dont le résultat peut être vrai ou faux;
- **Séquence d'instructions :** est le traitement à effectuer tant qu'expression logique est vraie.

Syntaxe :

while expression logique
Séquence d'instructions
end

Remarque

Expression logique est en général le résultat d'un test (**par exemple i < Imax**) ou le résultat d'une fonction logique (par exemple **all(x)**). Il est impératif que le traitement de la ***séquence d'instructions*** agisse sur le résultat ***d'expression logique*** sans quoi on boucle indéfiniment.

Example 4

- Create **a new script** (Ctrl +N). Save with the name : **Exemple4_TP6.m**
- Use instruction: **while**.

➢ Réaliser une fonction Matlab qui compte de 1 jusqu'à 10. ➢ Use exemple **Exepmle1_TP6**	➢ Ecrire une fonction Matlab qui permet de calculer la somme de 1 jusqu'à n. ➢ Use exemple **Exepmle3_TP6**

```
function [] = Exemple4_TP6 ()
i=0;
while i<10
    fprintf('i=%d ', i);
    i=i+1;
end
```

```
function [] = Exemple4_TP6 ()

n=input('n= ');
i=0;
k=0;
while i<=n
    k=k+i;
    fprintf('i=%d ', i);
    i=i+1;
end
fprintf('\n la somme = %d \n', k);
end
```

Example 5

- Create **a new script** (Ctrl +N). Save with the name : **Exemple5_TP6.m**
- Use instruction: **while**.

➢ Réaliser une fonction Matlab qui calcule le produit de 1 jusqu'à n.

```
function [] = Exemple5_TP6 ()
disp('calcule de produit de 1 jusqu''a n');
n=input('n= ');
i=1;
k=1;
while i<=n
    k=k*i;
    fprintf('i=%d ', i);
    i=i+1;
end
fprintf('\n le produit de 1 à %d égal = %f \n', n, k);
end
```

➢ Réaliser une fonction Matlab qui permet de calculer la factorielle de n.

```
function [] = Exemple5_TP6 ()
disp('calcule de produit de 1 jusqu''a n');
n=input('n= ');
i=1;
k=1;
if n==0
    fprintf('\n la factorielle de %d égal = 1 \n', n);
else
while i<=n
    k=k*i;
    fprintf('i=%d ', i);
    i=i+1;
end
fprintf('\n la factorielle de %d égal = %f \n', n, k);
end
end
```

Partie 02 : Partie Simulation (Conditions)

Exercise 1 (display elements of any vector)

- Create **a new script** (Ctrl +N). Save with the name : **Exercice1_TP6.m**
- Réaliser une fonction qui affiche les éléments d'un vecteur un-par-un sur la même ligne.
- Créer un vecteur de 1 à n nombres (n<20).
- Use instruction: **for/while**.

```
function [] = Exercice1_TP6 ()
disp('affichage des éléments d''un vecteur de 1 à n');
n=input('n= ');
P=1:1:n;

i=1;
while i<=n
    fprintf('l''element %d est : P(%d)= %.f \n', i, i,P(i));
    i=i+1;
end

end
```

Exercise 2 (display elements of matrix)

- Create **a new script** (Ctrl +N). Save with the name : **Exercice2_TP6.m**
- Créer une matrice $M = [1\ 2\ 3;\ 3\ 4\ 5;\ 6\ 7\ 8]$;
- Réaliser une fonction qui affiche les éléments d'une matrice un-par-un.
- Use boucles: **for ou while** (il faut utiliser 2 boucles).

```
function [] = Exercice2_TP6 ()
clc;
disp('affichage des éléments un-par-un d''une matrice');
M=[2 3 4; 5 6 7; 8 9 10];
disp(M);
for i=1:3
    for j=1:3
    fprintf('M(%d,%d)= %.f \n',i,j,M(i,j));
    end
end
end
```

Exercise 3 (display elements of matrix)

- Create **a new script** (Ctrl +N). Save with the name : **Exercice3_TP6.m**
- Réaliser une fonction qui affiche les éléments d'une matrice aléatoire de dimension n un-par-un.

```
n=input('dimension de la matrice carrée égal');
M=rand(n);
```

Exercise 4 (display elements of matrix)

- Create **a new script** (Ctrl +N). Save with the name : **Exercice4_TP6.m**
- Réaliser une fonction qui affiche la 3ème ligne d'une matrice magique de dimension 5.

```
>> A=magic(5);
for j=1:5
fprintf('A(5,%d)=%.f, ', j,A(5,j));
end
A(5,1)=11, A(5,2)=18, A(5,3)=25, A(5,4)=2, A(5,5)=9,
```

Exercise 5 (introduce a matrix element by element)

- Create **a new script** (Ctrl +N). Save with the name : **Exercice5_TP6.m**
- Réaliser une fonction qui introduire élément-par-élément une matrice carrée de dimension n.
- Use instruction: **for/while**.

```
function [] = Exercice5_TP6 ()
clc;
disp('introduce a matrix element by element');
n=input('entrer nbr des lignes ');
m=input('entrer nbr des colonnes ');
M=zeros(n,m);

for i=1:n
    for j=1:m
        fprintf(' l''element Matrix(%d,%d)= ',i,j);
        M(i,j)=input('');
    end
end
fprintf('la matrice(%d,%d) introduire égal \n',i,j);
disp(M);
end
```

Partie 03 : Partie expérimentale (MATLAB – SIMULINK)

Exercice 6 de préparation (teste d'inversement d'une matrice)

- Create **a new script** (Ctrl +N). Save with the name : **Exercice6_TP6.m**
- Réaliser une fonction qui contient une matrice carrée M d'ordre 12 contenant les entiers de 1 à 144 rangés par ligne.
- Calculer le déterminant de M.
- Calculer l'inverse de la matrice M.
- **Extraire de cette matrice :**
  - Une sous-matrice **A** formée par les coefficients $\boldsymbol{A_{ij}}$ pour : $\mathbf{1} < (\boldsymbol{i},\boldsymbol{j}) < \mathbf{6}$;
  - Une sous-matrice **B** d'ordre 3 formée par les premiers coefficients pairs de $\boldsymbol{M_{ij}}$.
  - Modifier par la valeur de zéros toutes les valeurs du diagonal de la matrice M.

Fiche TP : 07
Graphisme (Gestion des fenêtres graphiques, plot(), fplot())

Centre universitaire Nour Bachir El-Bayadh
Institut des sciences
Département de technologie
Spécialité : HYD, GC, ETT, ELN et TLC
Niveau : Licence 02 (semestre 03)
Travaux Pratiques Informatique 3

TP 07 : Graphisme (Gestion des fenêtres graphiques, plot(), fplot())

L'objectif de TP :

Ce TP07 a pour but de vous découvrir la gestion des fenêtres graphiques et les outils pour réaliser des graphes scientifiques de qualité présentant clairement les données scientifiques.

Partie 01 : Partie théorique (Gestion des fenêtres graphiques 2D)

S'il est utile de pouvoir faire des calculs numériques, il est aussi utile d'avoir une représentation graphique des résultats.

On va commencer par tracer le graphe d'une fonction, la commande principale qui permet de tracer un graphe sous Matlab est l'instruction ***plot***() ou bien ***fplot***().

La commande ***plot***()

l'instruction ***plot***() permet de tracer graphiquement 2D un ensemble de points de coordonnées :

$$(x_i\ ,\ y_i),\ avec\ i = 1, \dots, N$$

Syntaxe

$plot\ (y)$

$plot\ (x, y)$

$plot\ (x, y, 'PropertyName', PropertyValue, \dots)$

Où

x : est le vecteur contenant les valeurs x_i en abscisse, et

y : est le vecteur contenant les valeurs y_i en ordonnée.

Bien entendu les vecteurs x et y doivent être de même dimension mais il peut s'agir de vecteurs lignes ou colonnes. Par défaut, les points (x_i, y_i) sont reliés entre eux par des segments de droites.

La première chose à faire est donc de regarder Syntaxe de la fonction (et de la comprendre) :

- L'instruction **Plot** prend un, deux ou trois arguments, ou des multiples de trois ;
- S'il n'y a qu'un seul argument, plot dessine la valeur du vecteur passe en argument, en fonction de son index ;
- S'il y a deux arguments X et Y, plot dessine Y en fonction de X ;
- Le troisième argument permet de passer des options :

(Changer la couleur, l'apparence du graphique etc.)

Exemple 1

- Create **a new script** (Ctrl +N). Save with the name : **Exemple1_TP7.m**
- Tracer le graphe de la fonction $f(x) = 5x + 2$ entre -5 et $+5$:

```
% Vecteur X
x=[-5:0.001:5];
% Vecteur f(x)
f = 5*x+2;
% graphe y=f(x)
plot (x, f);
```

- Comparer avec l'instruction $\boldsymbol{plot(f)}$; utiliser ***Hold on*** pour afficher les deux fonctions sur la même figure.

Exemple 2

- Create **a new script** (Ctrl +N). Save with the name : **Exemple2_TP7.m**
- Tracer le graphe de la fonction $g(x) = 5x + 2$ entre -5 et $+5$:

```
% Vecteur X
x=[-5:1:5];
% Vecteur g(x)
g = 5*x+2;
% graphe y=g(x)
hold on
plot (x, g);
```

Exemple 3

- Create **a new script** (Ctrl +N). Save with the name : **Exemple3_TP7.m**
- Tracer le graphe de la fonction $y(x) = x * \sin(x)$ entre $-\boldsymbol{2\pi}$ $\boldsymbol{et}$ $\boldsymbol{2\pi}$:

```
% Vecteur X
x=[-2*pi:0.01:2*pi];
% Vecteur y(x)
y = x.*sin(x);
% graphe y=y(x)
plot (x, y);
```

- Utiliser l'instruction ***grid*** pour un quadrillage dans la figure.

Exemple 4

- Create **a new script** (Ctrl +N). Save with the name : **Exemple4_TP7.m**
- Tracer le graphe de la fonction $y1(x) = x * \sin(x)$ entre $-\boldsymbol{2\pi}$ $\boldsymbol{et}$ $\boldsymbol{2\pi}$:

```
% Vecteur X
x=[-2*pi:1:2*pi];
% Vecteur y(x)
y1 = x.*sin(x);
% graphe y=y1(x)
hold on
plot (x, y1);
```

➢ Comparer les résultats des exemples 3 et 4, Conclue !

Dans les exemples précédents on a défini un vecteur x_i de valeurs équi-réparties entre -5 **et** 5 **ou bien** -2π **et** 2π (avec un pas de 0.001 et 0.01 dans le premier cas et de 1 dans le deuxième cas) et on a calculé l'image par la fonction $f, g, y\ ou\ y1$ de ces valeurs (vecteur des images y_i), et par l'instruction plot on a affiché les points de coordonnées $(x(i),\ y(i))$.

Pour une courbe d'une fonction quelconque, on peut spécifier à MATLAB quelle doit être **sa couleur**, quel doit être ***le style de trait*** et/ou quel doit être ***le symbole à chaque point*** $A(xi\ ,\ yi)$.

Pour cela on donne un troisième paramètre d'entrée à la commande plot qui est une chaîne de 3 caractères de la forme $'cst'$ avec ***c*** désignant la couleur du trait, ***s*** le symbole du point et ***t*** le type de trait. Les possibilités sont les suivantes:

Couleur de trait		**symbole du point**		**type de trait**	
y	*jaune*	*.*	*point*	*-*	*trait plein*
m	*magenta*	*o*	*cercle*	*:*	*pointillé court*
c	*cyan*	*x*	*marque x*	*-*	*pointillé long*
r	*rouge*	*+*	*plus*	*-.*	*pointillé mixte*
g	*vert*	***	*étoile*		
b	*bleu*	*s*	*carré*		
w	*blanc*	*d*	*losange*		
k	*noir*	*v*	*triangle (bas)*		
^	*triangle (haut)*				
<	*triangle (gauche)*				
>	*triangle (droit)*				
p	*pentagone*				
h	*hexagone*				

Les valeurs par défaut sont $'c\ =\ b'$, $'s\ =\ .'\ et\ 't\ =\ -'$ ce qui correspond à un trait plein bleu reliant les points entre eux.

Il n'est pas obligatoire de spécifier chacun des trois caractères. On peut se contenter d'en spécifier un ou deux. Les autres seront les valeurs par défaut. Il est possible de tracer plusieurs courbes sur la même figure en spécifiant plusieurs tableaux x1, y1, x2, y2, ..., comme paramètres de l'instruction ***plot***. Si l'on souhaite que les courbes aient une apparence différente, on utilisera des options de couleurs et/ou de styles de traits distincts après chaque couple de vecteurs x, y.

Sauvegarder une figure

La commande ***print*** permet de sauvegarder la figure d'une fenêtre graphique dans un fichier sous divers formats d'images. La syntaxe de la commande print est:

$$\boldsymbol{print\ -f < num >\ \ -d < format >< nomfic >}$$

D'où:

- $< \boldsymbol{num} >$: désigne le numéro de la fenêtre graphique (figure 1, 2, ...).
- $< \boldsymbol{format} >$: désigne le format de sauvegarde de la figure. Ces formats sont nombreux. On pourra obtenir la liste complète en tapant help plot. Par exemple : Format d'image **JPEG**.
- $< \boldsymbol{nomfic} >$: désigne le nom du fichier dans lequel est sauvegardée la figure.

Partie 02 : Partie Simulation (Conditions)

Exercice 1

- Create **a new script** (Ctrl +N). Save with the name : **Exercice1_TP7.m**
- On trace sur l'intervalle $x \in [-5, 5]$ la fonction $\boldsymbol{f(x) = x^2\ cos(x)}$ en trait plein bleu, et la fonction $\boldsymbol{g(x) = x\ cos(x)}$ en trait pointillé rouge.

```
% Vecteur X
x = [-5:0.01:5];
% Vecteur f(x) et g(x)
f = x.^2.*cos(x); g = x.*cos(x);
% les graphes y=f(x) et y1=g(x)
plot(x,f,'b-',x,g,'r:');
```

Exercice 2 (l'instruction $loglog()$)

- Create **a new script** (Ctrl +N). Save with the name : **Exercice2_TP7.m**
- Si x et y sont deux vecteurs de même dimension, l'instruction $\boldsymbol{loglog(x, y)}$ permet d'afficher le vecteur $\boldsymbol{log(x)}$ contre le vecteur $log(y)$. La commande $loglog()$ s'utilise de la même manière que la commande $plot()$.

```
% Vecteur x
x = [1:10:1000];
% Vecteur y=f2(x)
f2 = x.^3;
% les graphes y=f2(x) et log(y)=log(f2(x))
plot(x,f2)
loglog(x,f2);
```

- Quelle est la pente de la droite?

Améliorer la lisibilité d'une figure

Légender une figure

Pour mettre une légende à une figure :

- la commande ***xlabel*** permet d'ajouter un texte en légende sous l'axe des abscisses. La syntaxe est : $\boldsymbol{xlabel('légende')}$
- La commande ***ylabel*** fait de même pour l'axe des ordonnées : $\boldsymbol{ylabel('légende')}$
- La commande ***title*** permet de donner un titre à la figure. La syntaxe est : $\boldsymbol{title('le\ titre')}$
- La commande ***gtext*** permet quant à elle de placer le texte à une position choisie sur la figure à l'aide de la souris. La syntaxe est $\boldsymbol{gtext('un\ texte')}$.

Exercice 3 (Amélioration de la légende d'une figure)

- Create **a new script** (Ctrl +N). Save with the name : **Exercice3_TP7.m**
- Utiliser les différentes commandes précédentes pour légender une figure.

```
P = 5;
t = [0:.01:2];
c = 12*exp(-2*t) - 8*exp(-6*t);
plot(t,c); grid
xlabel('temps en minutes')
ylabel('concentation en gramme par litre')
title(['evolution de la concentration du produit '])
gtext('concentration maximale')
```

Exercice 4 (Tracer le graphe d'une fonction; la commande $\boldsymbol{fplot}$())

La commande ***fplot*** permet de tracer le graphe d'une fonction sur un intervalle donné.

Syntaxe :

$\boldsymbol{fplot('nomf', [xmin\ ,\ xmax])}$

D'où :

- ***nomf*** **:** soit le nom d'une fonction MATLAB incorporée (cos, sin ...), soit une expression définissant une fonction de ***la variable x***, soit le nom d'une fonction utilisateur.
- $[\boldsymbol{xmin}\ ,\ \boldsymbol{xmax}]$: est l'intervalle pour lequel est tracé le graphe de la fonction.

- Create **a new script** (Ctrl +N). Save with the name : **Exercice4_TP7.m**
- **1.** Illustrons les trois façons d'utiliser la commande ***fplot***. On cherche le graphe de la fonction *sinus* entre $\boldsymbol{-2\pi\ et\ 2\pi}$ par l'instruction: $\boldsymbol{fplot('sin', [-2 * pi\ 2 * pi])}$.
- **2.** Pour tracer le graphe de la fonction $\boldsymbol{h(x) = x\ sin(x)}$ entre $\boldsymbol{-2\pi\ et\ 2\pi}$, on peut définir la fonction utilisateur $\boldsymbol{h(x)}$ de la manière suivante (attention de bien lire x.*sin(x) et non pas x*sin(x)):

$$\begin{cases} \boldsymbol{function\ y = h(x)} \\ \boldsymbol{y = x.* sin(x);} \end{cases}$$

- Chercher alors le graphe de la fonction $\boldsymbol{h(x)}$ par l'instruction: $\boldsymbol{fplot('h', [-2 * pi\ 2 * pi])}$.

- **3.** L'autre façon de procéder est d'exécuter l'instruction :$\boldsymbol{fplot('x*sin(x)', [-2*pi\ 2*pi])}$.

Exercice 5 (Tracer le graphe d'une fonction; la commande $\boldsymbol{fplot}$())

- Create **a new script** (Ctrl +N). Save with the name : **Exercice5_TP7.m**
- Illustrons deux fonctions $f(x) = \frac{sin(x)}{x}$ sur $[-5;\ 5]$ et $g(x) = \frac{cos(x)}{x}$ sur [-1; 1] utilisons l'instruction $\boldsymbol{fplot('[f,g]', [-5, 5\ , -1, 1])}$.
- Légender la figure et utiliser des courbes avec des couleurs différents.

Partie 03 : Partie expérimentale (MATLAB – SIMULINK)

Exercice 6 de préparation ()

- Create **a new script** (Ctrl +N). Save with the name : **Exercice7_TP7.m**
- Tracez le graphe de la fonction $\boldsymbol{f(x) = -48e^{-18x}\sin(314x - 120) - 48x}$ sur [-10; 10] utilisant ;
- L'instruction $\boldsymbol{fplot}$;
- L'instruction $\boldsymbol{plot}$.
- Conclure !!

Exercice 7 de préparation (teste d'inversement d'une matrice)

- Create **a new script** (Ctrl +N). Save with the name : **Exercice7_TP7.m**
- Tracez le graphe de la fonction carré sur [-1; 1] utilisant l'instruction : $\boldsymbol{plot}$()
- Tracez le graphe de la fonction carré sur [-1; 1] utilisant l'instruction : $\boldsymbol{fplot}$()
- Comparer les deux résultats.

Exercice 8 de préparation (Superposition des courbes dans une même figure)

- Create **a new script** (Ctrl +N). Save with the name : **Exercice8_TP7.m**
- Tracez les fonctions $\boldsymbol{f(x) = exp(x)}$ sur [-1; 1] ; **g(x)=x** sur [-1; 1] ; **h(x)=log(x)** sur [1/e; e] sur la même figure.
- Utiliser les instructions : $\boldsymbol{fplot}$() $\boldsymbol{ou\ plot}$()
- Utiliser les deux instructions : $\boldsymbol{hold\ on\ et\ hold\ off}$

Fiche TP : 08
Utilisation de toolbox

Centre universitaire Nour Bachir El-Bayadh
Institut des sciences
Département de technologie
Spécialité : HYD, GC, ETT, ELN et TLC
Niveau : Licence 02 (semestre 03)
Travaux Pratiques Informatique 3

TP 08 : Utilisation de toolbox

L'objectif de TP :

Ce TP08 a pour but de vous découvrir les opérations sur les vecteurs et les matrices afin de les utiliser pour la résolution des exercices.

Partie 01 : Partie théorique (Toolbox)

La Toolbox de MATLAB est une collection de fonctions et d'outils spécialement conçus pour une application spécifique. Les utilisateurs peuvent utiliser ces fonctions pour effectuer des tâches spécifiques sans avoir à écrire leur propre code. Pour utiliser une Toolbox, il suffit de l'installer et de l'ajouter à votre environnement de travail de MATLAB. Vous pouvez alors accéder aux fonctions de la Toolbox à partir de la ligne de commande ou de l'interface utilisateur graphique. Certaines Toolboxes peuvent nécessiter une licence pour être utilisées, mais il existe également de nombreuses Toolboxes gratuites disponibles.

La "boîte à outils" dans MATLAB fait référence à un ensemble de fonctions MATLAB (également appelées "boîtes à outils") qui fournissent des fonctionnalités supplémentaires pour une application ou un domaine d'étude spécifique. Ces boîtes à outils sont distinctes du système MATLAB de base, mais peuvent être ajoutées à l'environnement MATLAB selon les besoins.

Par exemple, il existe des boîtes à outils disponibles pour les systèmes de contrôle, le traitement d'image, l'optimisation, le traitement du signal, les statistiques et de nombreux autres domaines. Ces boîtes à outils fournissent un ensemble complet de fonctions et d'outils pour résoudre les problèmes et effectuer des tâches dans le domaine respectif.

Une fois qu'une boîte à outils est installée, ses fonctions peuvent être appelées comme n'importe quelle autre fonction MATLAB, et ses outils d'interface utilisateur (tels que des boîtes de dialogue ou des interfaces utilisateur graphiques) peuvent être utilisés pour interagir avec la boîte à outils.

Control System Toolbox

Un package pour Matlab composé d'outils spécifiquement développés pour les applications de contrôle. Le progiciel propose des structures de données pour décrire des représentations de système courantes telles que des modèles d'espace d'état et des fonctions de transfert, ainsi que des outils d'analyse et de conception de systèmes de contrôle. Il existe également des outils de simulation de systèmes.

Dans cette fiche, vous apprendrez à connaître les commandes de base de Control System Toolbox. Lorsque vous aurez terminé cet exercice, vous devriez être en mesure de comprendre et d'utiliser Control Systems Toolbox pour créer et analyser des systèmes linéaires. Une utilisation intensive de la commande d'aide de Matlab est recommandée. Il est également recommandé de créer un fichier de script (par exemple EX01.m) dans lequel vous écrivez vos commandes.

En exécutant un fichier de script au lieu de taper les commandes directement à l'invite Matlab, il est plus facile de corriger les erreurs et votre travail sera également enregistré pour une utilisation ultérieure.

Exemple 1

Le système que vous utiliserez est :

$$\dot{x} = Ax + Bu = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = Cx = [1 \quad 0]$$

Création et conversion de systèmes

Control System Toolbox prend en charge plusieurs représentations système de systèmes linéaires invariants dans le temps. Dans cet exercice, nous utiliserons deux des représentations les plus courantes ; modèles d'espace d'états et fonctions de transfert.

Définissez les matrices du système A, B, C et D données ci-dessus. (Quelle est la valeur de D dans le modèle ?) Créez une description de l'espace d'état du système à l'aide de ss et nommez-la sys_ss. Découvrez comment utiliser ss en utilisant la fonction d'aide (***help ss***). A ce stade, vous devriez avoir obtenu une description de l'espace d'état du système.

Créons maintenant un modèle de fonction de transfert équivalent du système ci-dessus.

Cela pourrait, comme vous le savez, être fait en utilisant la formule :

$$G(s) = C(sI - A)^{-1} B + D$$

Cependant, Matlab peut également être utilisé pour la tâche. Utilisez la ***commande tf*** pour convertir le modèle d'espace d'états en une fonction de transfert et nommez-le ***sys_tf***. Notez que ***tf*** peut être utilisé pour la création de fonctions de transfert ainsi que pour la conversion.

Quelle est la syntaxe dans les deux cas respectivement ?

Exemple 2 (Analyse de stabilité)

La stabilité d'un système linéaire est déterminée par l'emplacement de ses pôles dans le plan complexe.

Quelle est la condition de stabilité ?

Utilisez ***les commandes ssdata et tfdata*** pour extraire les données nécessaires des modèles, et ***eig et roots*** pour déterminer la stabilité du système. Vérifiez que les racines du polynôme caractéristique de la fonction de transfert sont les mêmes que les valeurs propres de la matrice du système.

Quelles sont les valeurs propres / pôles ?

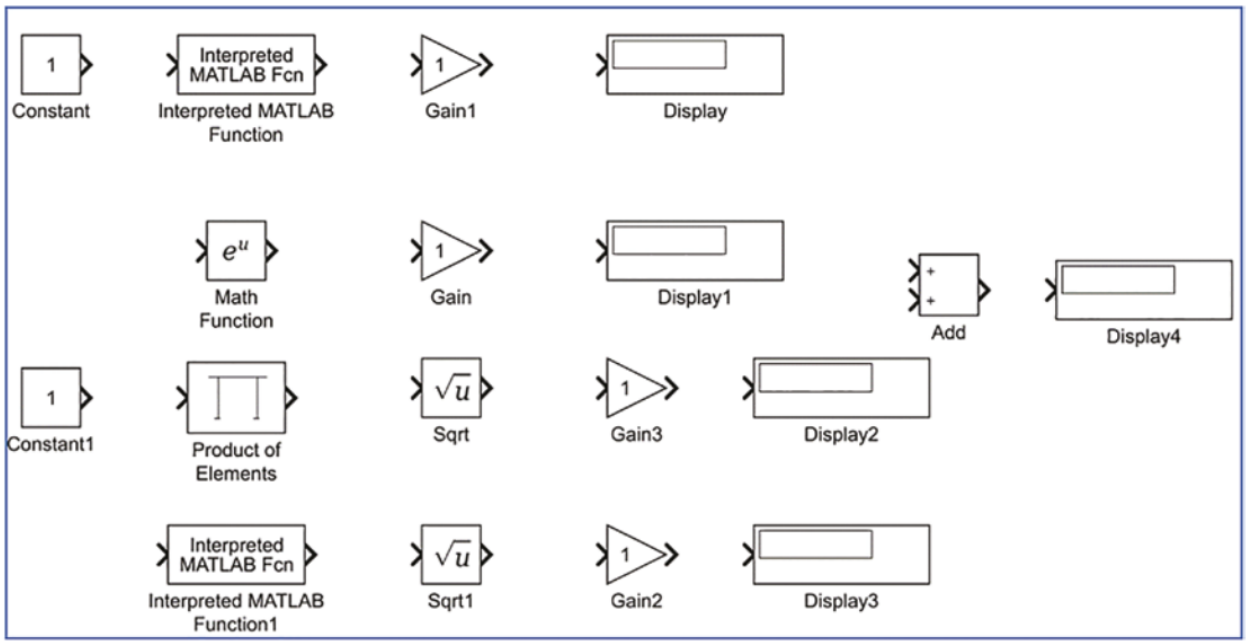
Le système est-il stable ?

Vous pouvez également utiliser le pôle de commande, ou pour une vue graphique, ***pzmap***.

Exemple 3 (Analyse du domaine temporel)

Utilisez la ***commande step*** pour tracer la réponse indicielle du système. Reliez les caractéristiques de la réponse indicielle à l'emplacement des pôles. Si vous avez le temps, utilisez initial et ***lsim*** pour étudier la réponse du système.

Tous les blocs nécessaires pour ce modèle

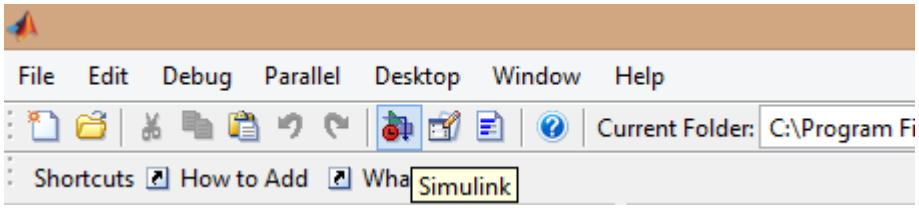


Quelques commandes Matlab utiles

plot	Linear plot.
subplot	Breaks the Figure window into small axes.
axis	Control axis and scaling appearance.
hold	Hold current graph.
grid	Grid lines.
title	Graph title.
xlabel, ylabel	Axes labels.
tf	Create a transfer function model.
tfdata	Extract numerator and denominator.
ss	Create a state-space model.
ssdata	Extract state-space matrices.
zpk	Create a zero/pole/gain/model.
step	Step response.
initial	Response of state-space model with given initial state.
pole	System poles.
zero	System zeros.
roots	Find polynomial roots.
pzmap	Pole-zero map.
eig	Compute eigenvalues and eigenvectors.
bode	Draw a bode frequency response.
lsim	Simulate time response of LTI models to arbitrary inputs.
simulink	Open the simulink browser.
sim	Use a Simulink model from a Matlab script
simset	Set options for the sim command
ictools	Interactive tools for control
pplane6	Phase plane analysis, download from our webside
linmod	Obtain the state-space linear model of the system of ordinary differential equations described in a simulink model, note that the model must contain a simulink block *out* from sinks and a simulink block *in* from sources

Présentation de Simulink

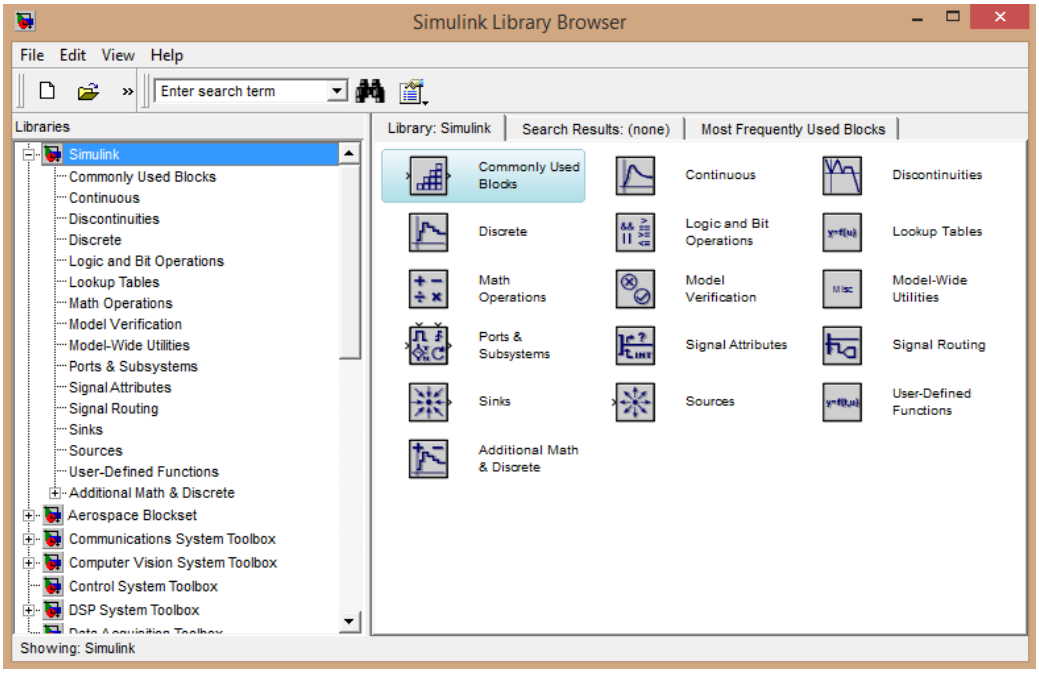
L'instruction Simulink donne un programme de simulation basé sur Matlab nommé Simulink. Il existe plusieurs façons de définir un modèle. On peut travailler graphiquement et connecter des blocs-diagrammes avec des blocs prédéfinis. Alternativement, on peut donner la description mathématique sous forme d'équations différentielles dans un fichier m (le format des programmes écrits dans le langage de programmation Matlab). Matlab/Simulink prend en charge ces deux représentations ainsi que les combinaisons. De plus, on peut utiliser des descriptions qui incluent une hiérarchie de sous-systèmes connectés.



Pour comprendre comment les modèles sont décrits et simulés à l'aide de schémas fonctionnels, il est préférable d'exécuter de petits exemples sur un ordinateur. Le reste de la section 2 montre quelques exemples. Si vous connaissez Simulink, vous pouvez passer directement à la section 3.

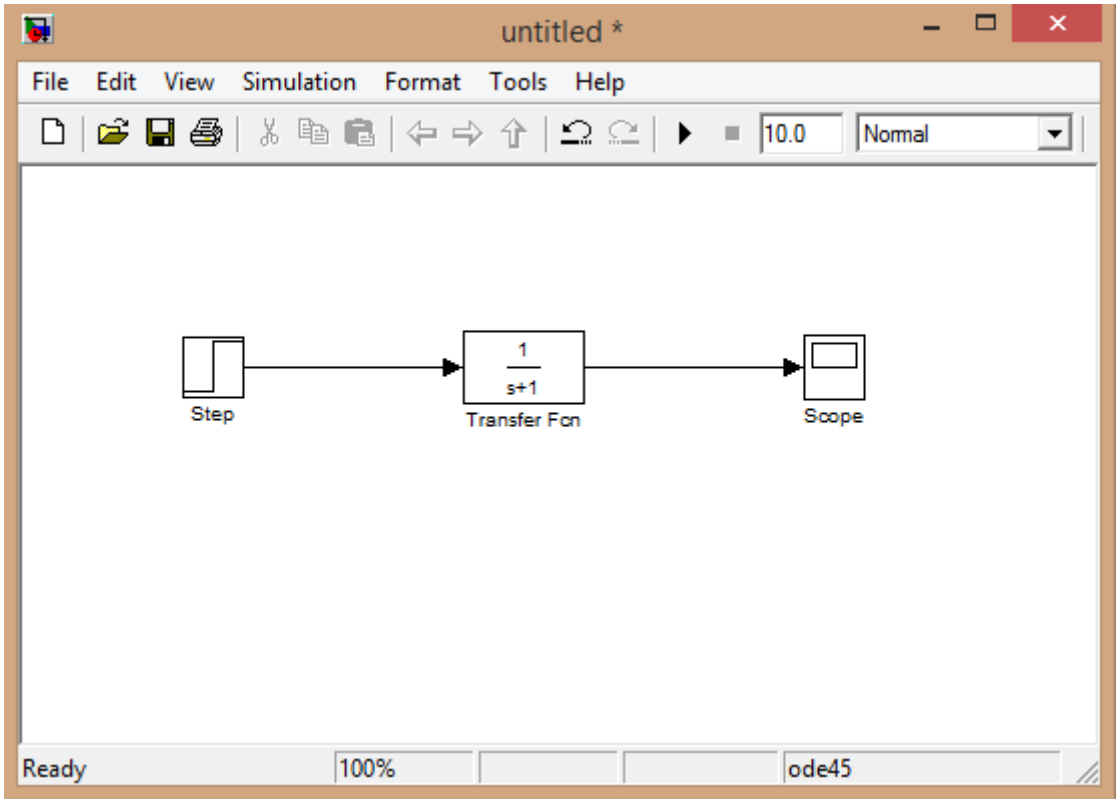
Comment démarrer Simulink

Donnez la commande simulink dans Matlab (command window). Cela donne une fenêtre avec des blocs comme dans la figure ci-après.



Un système simple

- Cliquez sur Fichier dans la fenêtre Simulink et choisissez Nouveau->Modèle.
- Cliquez sur le bloc Continu et déplacez un ***Transfert Fcn*** vers la nouvelle fenêtre appelée « untitled ».
- Faites de même avec Source->Step Fcn et Sinks->Scope.
- Dessinez des flèches (bouton gauche de la souris) et connectez les ports sur le bloc.
- Vous devriez maintenant avoir un schéma fonctionnel comme dans la figure ci-après.



- Choisissez Simulation->Paramètres dans la fenêtre intitulée " untitled ".
- Réglez l'heure d'arrêt sur 5. Ouvrez la fenêtre Portée en double-cliquant dessus.
- Mettez la plage horizontale à 6.
- Lancez une simulation par Simulation->Démarrer (ou en appuyant sur Ctrl-t dans la fenêtre intitulée "Sans titre").

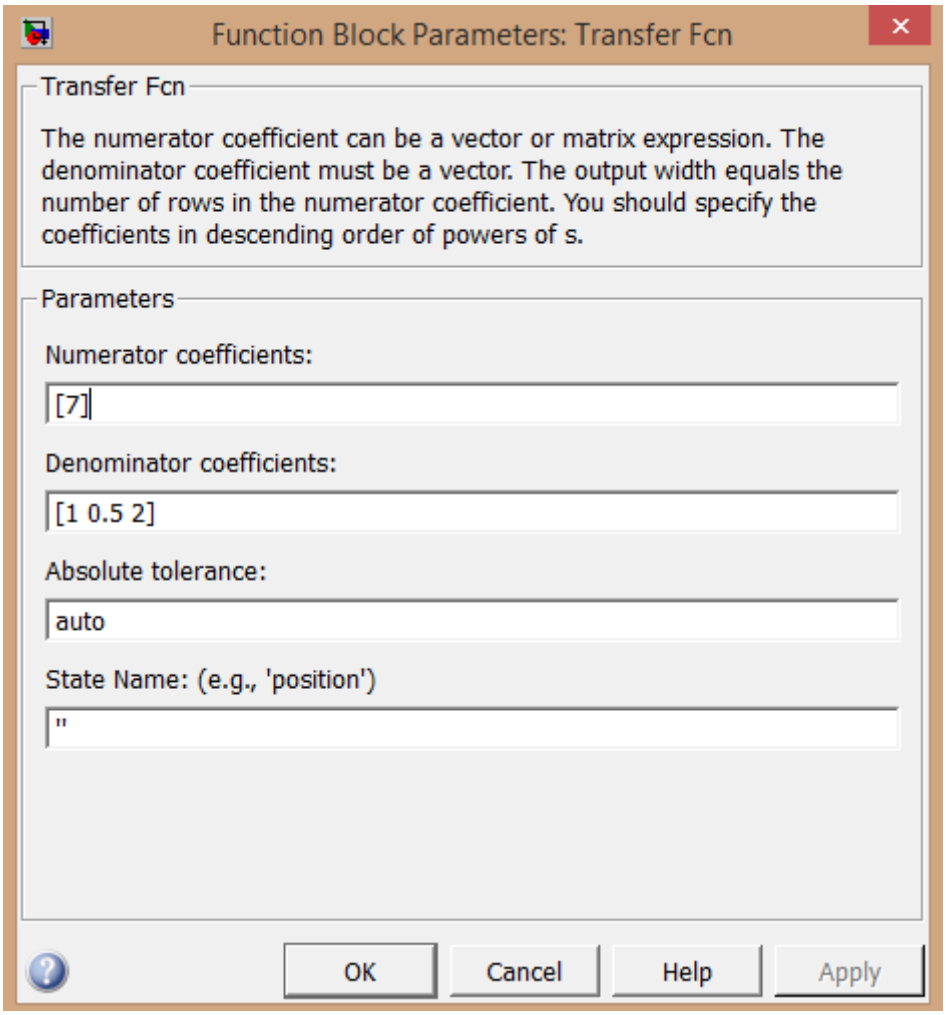
Exemple 4 (changer un système)

Pour changer le système en :

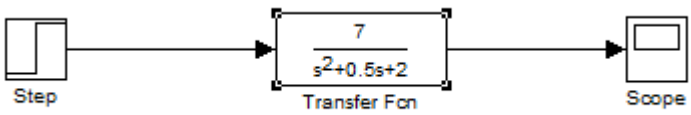
$$\frac{7}{s^2 + 0.5s + 2}$$

Double clic sur ***Transfert Fcn***

- Vous devriez maintenant avoir un schéma fonctionnel comme dans la figure ci-après.



Le système devient :



Comment changer un signal d'entrée Pour changer le signal d'entrée, commencez par supprimer le bloc Step Fcn en cliquant dessus et supprimez-le en utilisant Edit->Cut (ou en appuyant sur Ctrl-x). Remplacez-le par un bloc Sources->Signal Gen. Double-cliquez sur Signal Gen et sélectionnez le signal, l'amplitude et la fréquence. Modifiez également Simulation->Démarrer->Temps d'arrêt sur 99999 et appuyez sur Simulation->Démarrer. Cela donne une simulation "infinie" qui peut être arrêtée en appuyant sur Simulation->Stop (ou Ctrl-t).

L'amplitude du signal d'entrée peut-elle être modifiée pendant la simulation ? Essayez également de changer le bloc Transfert Fcn pendant la simulation.

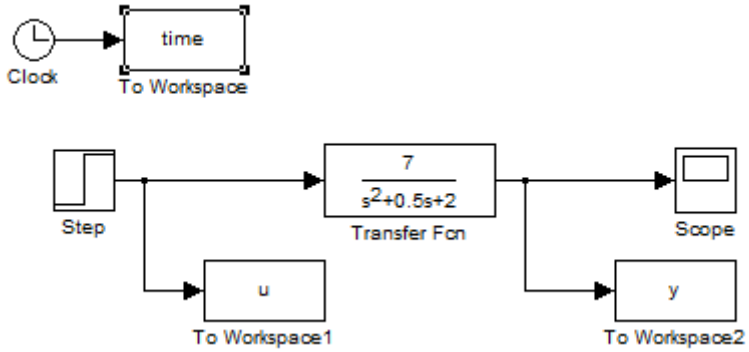
Comment utiliser les variables Matlab dans les blocs Notez que les variables définies dans l'environnement Matlab peuvent être utilisées dans Simulink. Définissez le numérateur et le dénominateur en écrivant ce qui suit dans la fenêtre Matlab.

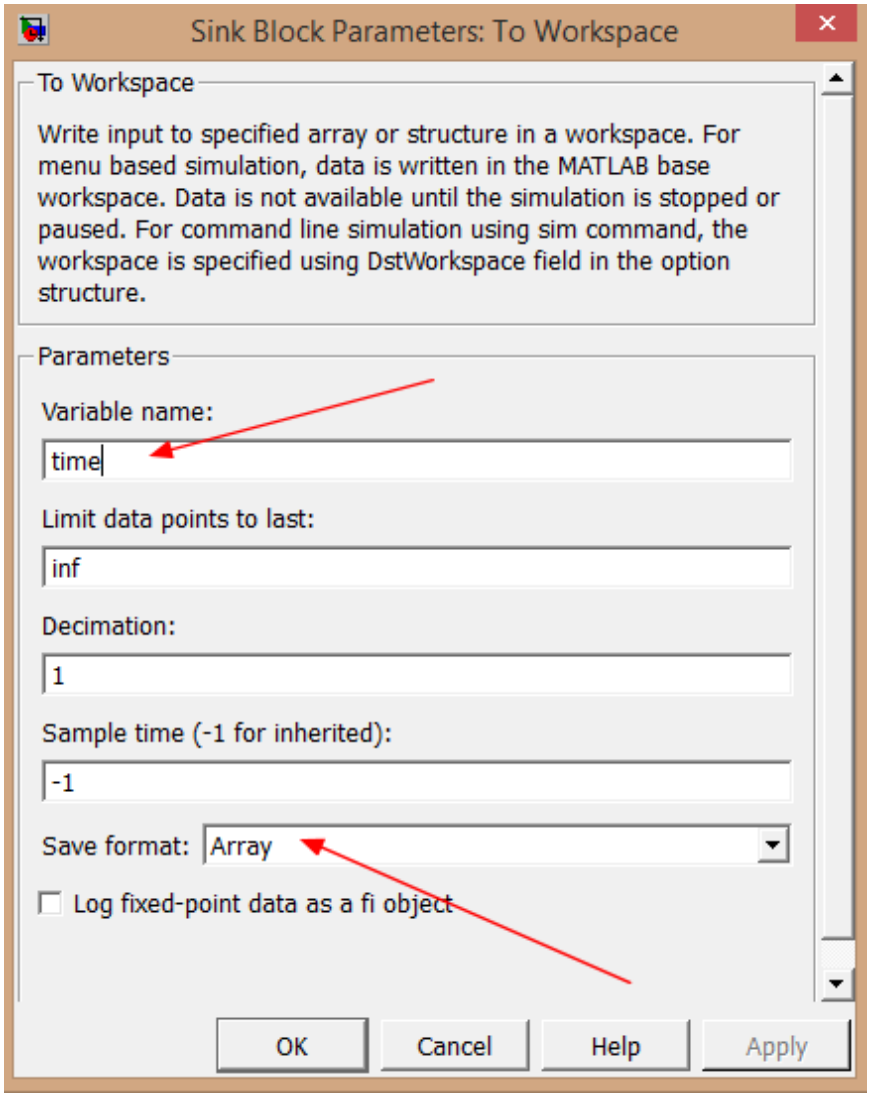
nombre=[7 6]

repaire=[1 2 3 4]

Remplacez Transfer Fcn->Numerator par num et Transfer Fcn->Denominator par den.

Comment enregistrer les résultats dans les variables Matlab Pour enregistrer l'entrée et la sortie, déplacez deux copies du bloc Sinks->To Workspace. Assurez-vous que l'option "enregistrer le format" est définie sur "Array", voir figure 2. Connectez-les avec l'entrée et la sortie au bloc Transfer Fcn. Obtenez également une Source-> Clock et connectez-la à un Sinks-> To Workspace. Remplacez les noms de variables par u, y, t respectivement. La fenêtre doit ressembler à la figure.





- Comment utiliser les résultats de la simulation dans les calculs Matlab Supposons que le signal d'entrée soit sinusoïdal avec une fréquence de 0,1 rad/s et une amplitude de
- Effectuez une simulation suffisamment longue pour que la sortie devienne stationnaire.
- Calculez la valeur maximale de y lorsque le système s'est stabilisé.

n=longueur(y)

max(y(n/2:n))

Utilisation des modèles Simulink dans les scripts Matlab

Souvent, il est pratique de travailler avec des scripts Matlab (fichiers m), afin de sauvegarder une séquence de commandes. Il est possible d'utiliser des modèles Simulink depuis un script Matlab, en utilisant la commande sim. En utilisant la commande simset, les options de la commande sim peuvent être spécifiées.

Utilisez le modèle de l'exemple précédent. Enregistrez le modèle et nommez-le "mymodel.mdl". Créez un script Matlab nommé "mysim.m", et saisissez les commandes suivantes :

1. *tfinal = 300;*
2. *options = simset('reltol',1e-5,'refine',10,'solver','ode45');*
3. *sim('mymodel',tfinal,options);*

   %plot results
4. *figure(1)*
5. *clf*
6. *subplot(211)*
7. *plot(t,u);*
8. *ylabel('u')*
9. *subplot(212)*
10. *plot(t,y)*
11. *ylabel('y')*

Lorsque vous exécutez le script, vous devriez voir un tracé montrant l'entrée et la sortie de la fonction de transfert. Utilisez la commande help pour en savoir plus sur l'utilisation des commandes simset et sim.

Comment enregistrer des systèmes Utilisez Fichier-Enregistrer sous ou Fichier->Enregistrer.

Partie 02 : Partie Simulation (Un système de flux)

Considérez un réservoir simple comme dans le cours de contrôle de base

$$\dot{h} = \frac{1}{A}(u - q)$$

$$q = a\sqrt{2gh}$$

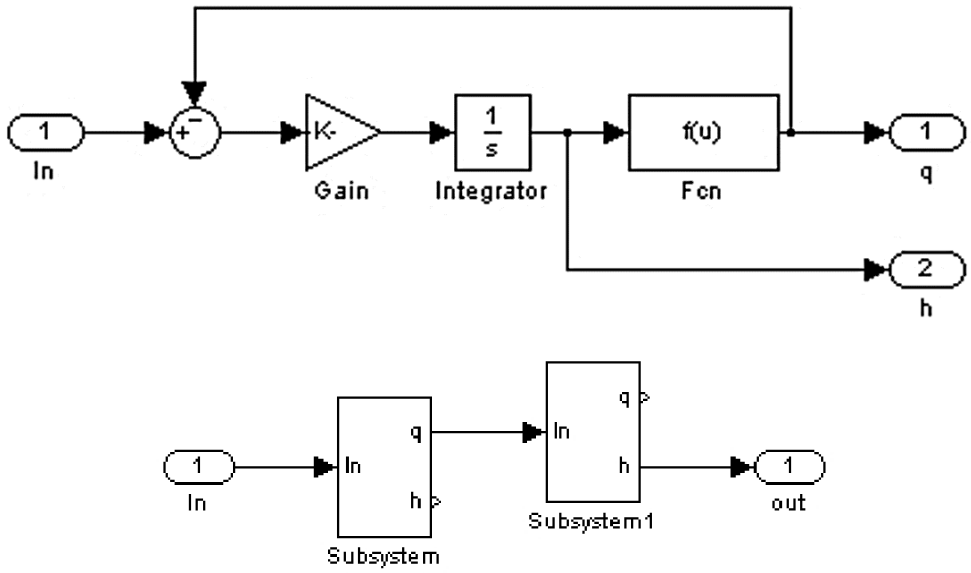
Cela peut être implémenté dans Simulink comme dans la figure ci-après. La fonction f (u) a la valeur $a * sqrt(2 * g) * sqrt(u[1])$. Le bloc Sum a reçu deux entrées avec des signes différents en attribuant la chaîne "-+|" à Somme->Liste des signes. Les blocs d'entrée et de sortie se trouvent respectivement sous les sources et les puits.

Ces blocs indiquent à Simulink ce qui doit être considéré comme des entrées et des sorties vers ce sous-système. Les titres des blocs peuvent être modifiés en cliquant dessus. Marquez l'ensemble du système en maintenant le bouton gauche de la souris enfoncé et en dessinant un carré autour. Ensuite, choisissez Edition->Créer un sous-système. Le résultat est que le système est représenté par un bloc. Utilisez Édition->Copier pour créer le système à double réservoir suivant.

Exercice 1

Utilisez la ***commande linmod*** pour trouver un modèle linéarisé du double réservoir autour de $h_1^0 = h_2^0 = 0.1$. Utilisez les paramètres $A_1 = A_2 = 2.7\ 10^{-3}$, $a_1 = a_2 = 7 \times 10^{-6}$, $g = 9.8$. Remarquez également dans la figure le bloc Simulink à partir des sources et le bloc Simulink à partir des puits, qui sont nécessaires pour le ***commande linmod***.

>> A=2.7e-3; a=7e-6; g=9.8;



>>x0 = [0.1000 0.1000]';

*>>u0 = a*sqrt(q*g*x0(1));*

>> [aa, bb,cc,dd]=linmod('flow',x0,u0);

Partie 03 : Partie expérimentale (MATLAB – SIMULINK)

Exercice 6

Construisez un modèle Simulink pour calculer les valeurs de la fonction cosinus g(t) = cos (ωt) pour t = 0…3 avec 3 000 pas incrémentiels et ω = [π, 2π, 3π, 5π, 7π] spécifié dans MATLAB. Simulez votre modèle Simulink à l'aide de MATLAB avec la commande sim() en utilisant une boucle [for ... end] ou [while .. end] pour toutes les valeurs de ω.

Exercice 7

L'accélération d'un parachutiste est déterminée par :

$$a = g(1 - \frac{v^2}{3600})$$

D'où: g = 9,81 m/sec2.

Créez un modèle Simulink pour simuler l'accélération d'un parachutiste.

Exercice 8

L'introduction de fonctions de transfert s'effectue sous plusieurs formes polynomiales, forme ZPK (zéros, pôles, gain), forme d'état. Prenons quelques types de systèmes asservis, deuxième et troisième ordre pour s'entrainer sur la déclaration de ces systèmes.

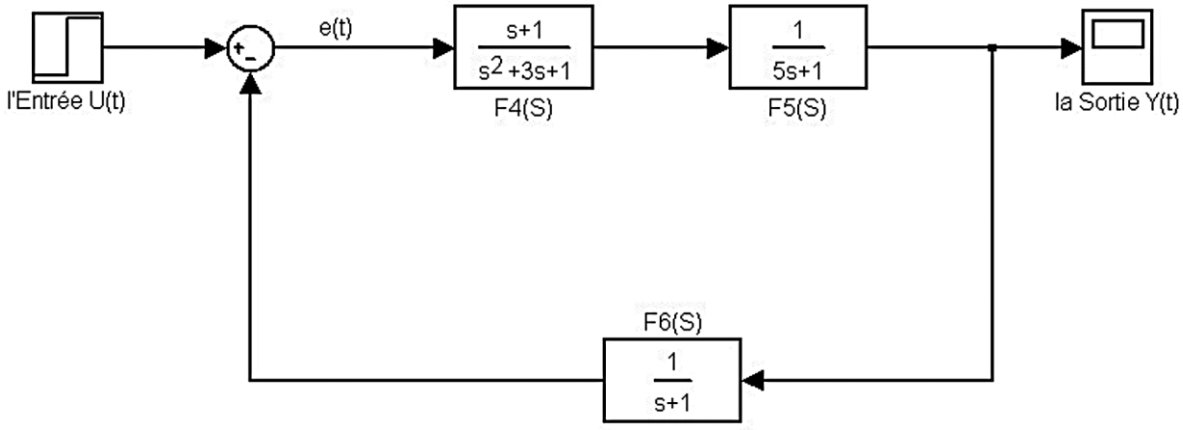
$$f_1(s) = \frac{s+5}{s^3 + 8s^2 + 17s + 10};\ f_2(s) = \frac{s+10}{2s^2 + 3s + 1}$$

$$f_3(s) = 4.25\frac{(s-2)(s+1.2)}{(s+0.3)(s+1)(s+5)}$$

1. Consulter l'aide en ligne sur les commandes Tf, ZPK, TF2SS, TF2ZP
2. Créer un fichier de commande tp3ex1.m pour les commandes MATLAB de l'exercice N°1
3. Introduire **f1(s)** et **f2(s)** sous MATLAB
4. Donner les pôles de ces deux systèmes
5. Introduire le système **f8(s)** en utilisant la commande ZPK
6. Passer de la représentation fonction de transfert de **f1(s)** et **f2(s)** vers la forme ZPK connue par la forme d'Evans **H(s)** = **K** (**s-Z(1)**)(**s-Z(2)**)...(**s-Z(n)**) (**s-P(1)**)(**s-P(2)**)...(**s-P(n)**)).
7. Passer de la représentation fonction de transfert de f9(s) et f4(s) vers la forme d'état Ẋ = AX + Bu, y = CX + Du.
8. Passer de la représentation ZPK de f8(s) vers la forme Ẋ = AX + Bu, y = CX + Du.
9. Enregistrer l'espace de travail de cet exercice dans tp3ex1.mat

Exercice 9 (Construction de Schémas Fonctionnels)

Soit le schéma fonctionnel donné par la figure ci-dessous



1. Consulter l'aide en ligne sur les commandes Series, Parallel, feedback
2. Créer un fichier de commande tp3ex2.m pour les commandes MATLAB de l'exercice N°2
3. Donner la fonction de transfert en boucle ouverte
4. Donner la fonction de transfert en boucle fermée (Y(s)/U(s))
5. Vérifier le résultat analytiquement.
6. Calculer les pôles de la fonction de transfert en boucle fermée
7. Passer de cette fonction de transfert aux différentes formes (ZPK, SS) du système en BF
8. Enregistrer l'espace de travail de cet exercice dans tp3ex2.mat

Références Bibliographiques

Références Bibliographiques

Cours / TP : Informatique 3

Selon la disponibilité de la documentation au niveau de l'établissement, Sites internet...etc.)

1. **SCILAB :**
   - **1.1. Informatique: Programmation et simulation en Scilab 2014** - Auteurs : Arnaud Bégyn, Jean-Pierre Grenier, Hervé Gras.
   - **1.2. Scilab : De la théorie à la pratique - I. Les fondamentaux**. Livre de Philippe Roux 2013.

2. **MATLAB :**
   - **2.1. Références Françaises :**
     - Introduction à MATLAB et SIMULINK, Un guide pour les élèves de l'École Nationale Supérieure d'Ingénieurs Electriciens de Grenoble, Paolino Tona.
     - C++ pour les nuls, Stephen Randy Davis**.**
   - **2.2. Références Arabiques :**
     الماتلاب للمهندسين، المهندس عدنان شاهين
   - **2.3. Références Anglaises:**
     - **Essential MATLAB for Engineers and Scientists**, Seventh Edition, Brian D. Hahn & Daniel T. Valentine.
     - **Numerical Analysis Using MATLAB and Excel**, Steven T. Karris.
     - **Matlab for Dummies**, Jim Sizemore, John Paul Mueller.

Fin

Cours / TP : Informatique 3