**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE**

**UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE D'ORAN-M B-**

**FACULTE DE GENIE ELECTRIQUE**

**DEPARTEMENT D'AUTOMATIQUE**

**Polycopié de travaux pratiques:**

# Automatique des systèmes linéaires continus

**Présenté par : Dr. ZELMAT Mohammed El Mouloud**

**Septembre 2014**

# Avant-propos

Ce polycopié de travaux pratiques donne les éléments de base nécessaires à la compréhension des systèmes asservis linéaires. Il s'adresse aux étudiants de la troisième année préparant une licence en automatique.

## Fonctionnement général

## Organisation du cycle

Le cycle de TP se déroule sur 6 séances (Salle de TP 5110 A)
Il comprend 6 manipulations de 2 heures chacune.

Les différentes manipulations sont décrites par la liste ci-dessous.

### LISTE DES MANIPULATIONS

**TP 1** Amplificateur opérationnel 1
**TP 2** Amplificateur opérationnel 2
**TP 3** Introduction au logiciel Matlab
**TP 4** Analyse temporelle d'un système dynamique du 1$^{er}$ ordre
**TP 5** Analyse temporelle d'un système dynamique du 2$^{ème}$ ordre
**TP 6** Analyse fréquentielle, stabilité et asservissement d'un système

**Notation**
Chaque manipulation donne lieu à un compte-rendu par groupe à rendre, et qui fera l'objet d'une note qui prend en compte la préparation (contrôlée en début de séance), la participation des étudiants au déroulement du TP, les résultats obtenus et leur interprétation. La moyenne des 6 notes obtenues (une par TP) constituera la note finale associée à ce cycle d'enseignement.
En cas d'absence prévisible et préalablement justifiée par la Direction des études lors d'une séance, les étudiants doivent impérativement prévenir l'enseignant qui les encadre.
Les absences injustifiées donnent lieu à un zéro pour la séance concernée, sans aucune possibilité de rattrapage. De plus, tout retard non justifié à une séance entrainera la pondération de la note obtenue par l'étudiant à cette séance par un coefficient égal au prorata de la présence de l'étudiant durant la séance.

# TP1 : Amplificateur Opérationnel 1

## I. Objectifs pédagogiques

Etude théorique d'un amplificateur opérationnel et manipulation sur quelques montages de base :

- Amplificateur inverseur ;
- Amplificateur non inverseur ;
- Amplificateur comparateur.

## II. Matériels utilisés

- Une alimentation stabilisante $\pm$ 15V ;
- Un générateur de fonction ;
- Un oscilloscope double trace ;
- Une plaque d'essais universelle ;
- Un amplificateur, des résistances et des condensateurs ;
- Un potentiomètre ;
- Jeu de lignes de connexion des connecteurs.

## III. Rappels théoriques

L'amplificateur opérationnel (A.O) est un circuit intégré associée à des éléments passifs ou actifs, permet de réaliser des opérations sur des grandeurs analogiques (addition, multiplication, dérivation, intégration…etc.). Pour fonctionner, il doit donc être polarisé (entre -15V et +15V).

A étant très élevée (~100000), $U_D$ est très faible en régime linéaire : $U_D < 1\mu V$.

Pour un amplificateur opérationnel parfait (idéal) fonctionnant en régime linéaire :

$U_D = E^+ - E^- \equiv 0$ $I^+ = I^- = 0$

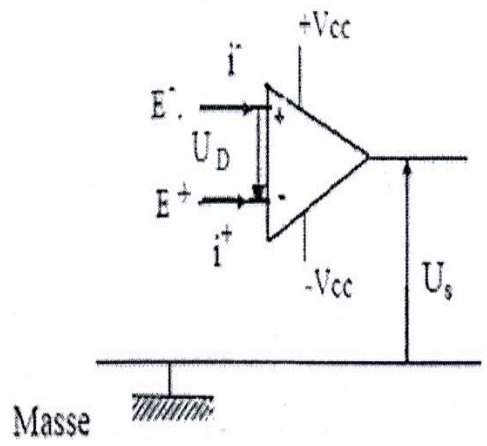


Fig.1 Schéma électrique de A.O

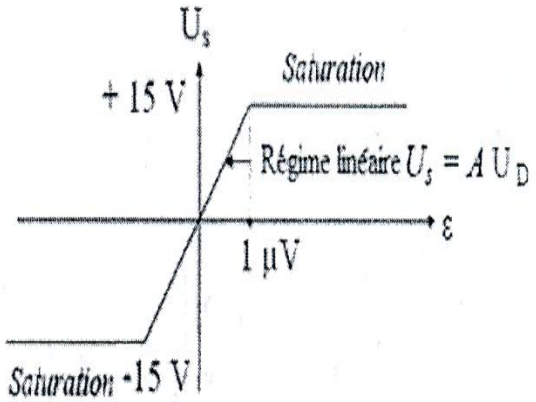


Fig.2 Caractéristique d'un A.O

## IV. Montages fondamentaux de l'amplificateur opérationnel

Les amplificateurs opérationnels sont souvent exploités avec la contre réaction. Une partie de la tension de sortie est ramenée à l'entrée $E^-$ par la résistance $R_g$.

### 1. Montage Inverseur

Dans ce montage Fig. 3, la tension d'entrée $U_E$ est appliquée par la résistance $R_1$.

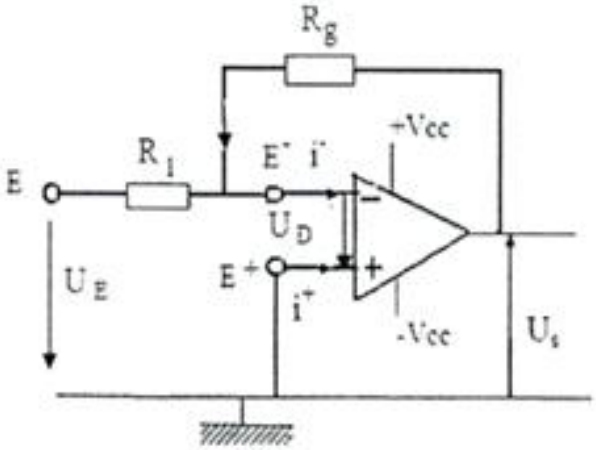


Fig.3 Amplificateur inverseur

$$\boldsymbol{U_s(t) = -\frac{R_g}{R_1}\, U_E(t)}$$

L'amplificateur : $\boldsymbol{V = -\frac{R_g}{R_1}}$

## 2. Montage non Inverseur

Dans ce montage Fig. 4, la résistance $R_1$ forme un diviseur de tension avec la résistance $R_g$.

$$\boldsymbol{U_s(t) = -\frac{R_1+R_g}{R_1}\, U_E(t)} \quad .$$

L'amplificateur : $V = 1 + \frac{R_g}{R_1}$

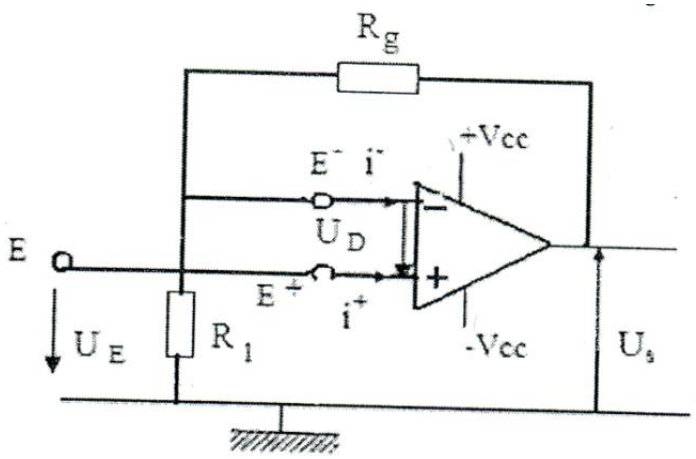


**Fig.4** Amplificateur non inverseur

## 3. Montage Comparateur

Ce circuit Fig. 5 sert à comparer les deux tensions $V_1$ et $V_2$.

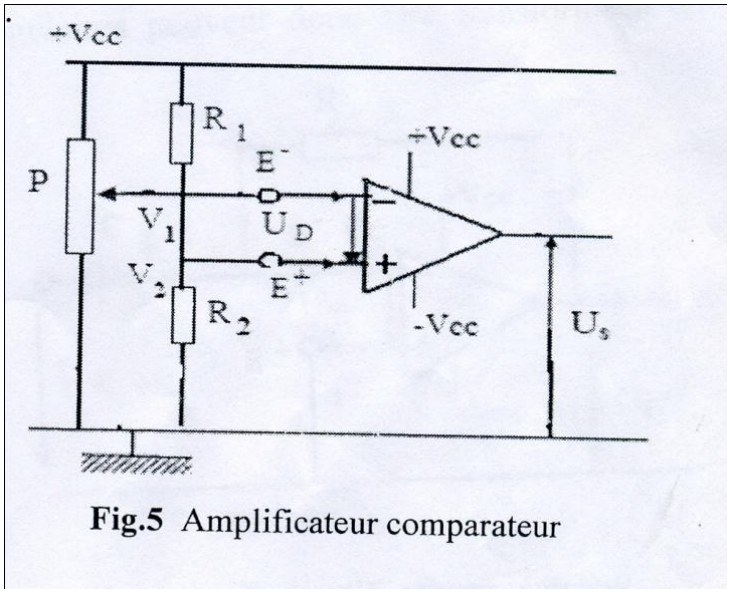


**Fig.5** Amplificateur comparateur

**$V_1 = U_E$** (Tension d'entrée) ; **$V_2 = V_{ref}$** (Tension de référence)

**$U_s(t) = + V_{cc}$ si $V_1 < V_2$ ;** **$U_s(t) = - V_{cc}$ si $V_1 > V_2$**

# V. Travail expérimental

**Remarques importantes à suivre avant toute mise en œuvre :**

- **Vérifier que l'alimentation est coupée avant chaque manipulation ;**
- **Alimenter d'abord le circuit avant d'injecter un signal d'entrée.**

## 1. Application du montage inverseur :

**1.1** Construire le montage **Fig.3,** avec $R_1$=1 KOhm et $R_g$ = 2,2 KOhm ;

- Sélectionner une tension d'entrée sinusoïdale d'amplitude 4V et de fréquence $f$ = 1Khz ;
- Identifier les deux tensions observées sur l'oscilloscope ;
- Tracer les courbes $U_E$ = f(t) et $U_s$ = f(t). Bien noter leurs valeurs maximales ainsi que la période T du signal ;
- Calculer l'amplification du montage $V = -\frac{U_s}{U_E}$
- Faites varier la tension d'entrée et mesurer les tensions de sortie correspondants :

| $U_E$(V) | 0.5 | 1 | 2 | 3 | 4 | 4.5 | 5 |
|---|---|---|---|---|---|---|---|
| $U_s$(V) | | | | | | | |

- Tracer $U_s = f(U_E)$ ;
- Vérifier que V= $-\frac{Us}{UE} = -\frac{Rg}{R1}$

**1.2** Modifier le circuit extérieur ($R_1$ =10 KOhm, $R_g$ =10 KOhm) ensuite ($R_1$ = 22 KOhm, $R_g$ = 10 KOhm) et répondre aux questions décrites dans **1.1**.

## 2. Application du montage non inverseur

Construire le montage Fig.4, et répondre aux questions décrites dans la manipulation précédente avec la prise en considération de l'amplification correspondante à ce montage.

## 3. Application du montage comparateur

Construire le montage **Fig.5,** avec $R_{1=}R_2$ = 1 k0hm et P =10 k0hm,

- Pour cette expérience, $V_2$ est la tension de référence à valeur fixe comparée a $V_1$variable représentant la tension d'entrée ;
- Mesurer $V_2$ et sélectionner une tension d'entrée continue ;
- Identifier les deux tensions observées sur l'oscilloscope ;
- Faire varier la tension d'entrée et mesurer les tensions de sortie correspondantes.

| $U_E$(V) | 0 | 2.5 | 5 | 7.5 | 8 | 10 | 12.5 | 15 |
|---|---|---|---|---|---|---|---|---|
| $U_S$(V) | | | | | | | | |

- Que peut –on conclure?

## VI. Conclusion générale

Faites vos conclusions sur les montages que vous avez réalisé.

# TP2 : Amplificateur Opérationnel 2

## I. Objectifs pédagogiques

Etude théorique et manipulations sur quelques montages réalisés à base d'amplificateur opérationnel :

- Sommateur ;
- Intégrateur;
- Dérivateur.

## II. Matériels utilisés

- Une alimentation stabilisante ± 15V ;
- Un générateur de fonction ;
- Un oscilloscope double trace ;
- Une plaque d'essais universelle ;
- Un amplificateur, des résistances et des condensateurs ;
- Un potentiomètre ;
- Jeu de lignes de connexion des connecteurs.

## III. Rappels théoriques

### III.1 Montage Sommateur

Ce circuit Fig. 6 sert à additionner les deux tensions d'entrée $U_{E1}(t)$ et $U_{E2}(t)$

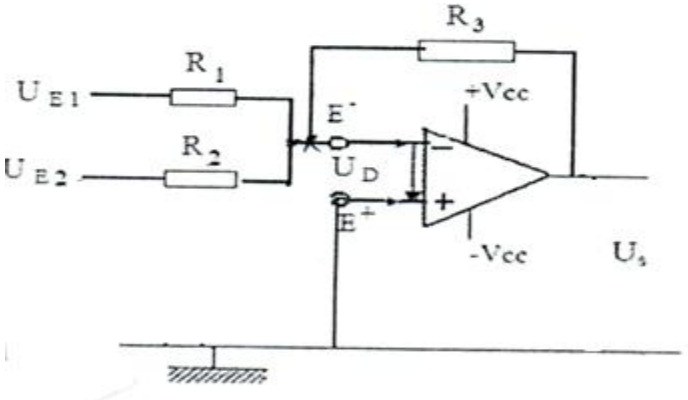


**Fig.6** Amplificateur sommateur

### III.2 Montage Intégrateur

Un intégrateur Fig. 7 est un circuit dont la tension de sortie est proportionnelle à la tension d'entrée lorsque celle-ci est appliquée. On peut ainsi convertir des tensions rectangulaires en tensions triangulaires avec différentes pentes.

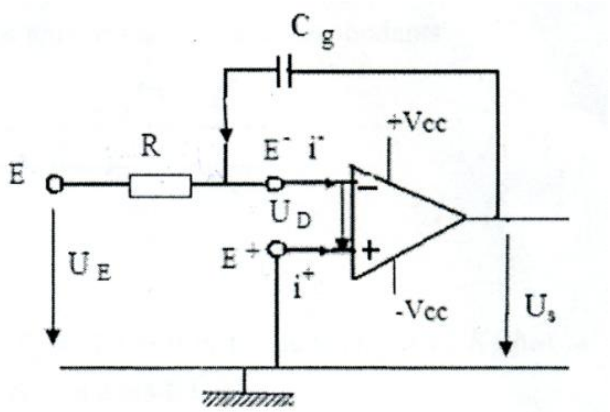


**Fig.7** Amplificateur intégrateur

$$U_s(t) = -\frac{1}{RC}\int U_E(t)dt$$

### III.3 Montage dérivateur

Un dérivateur est un circuit dont la tension de sortie est proportionnelle à la vitesse à laquelle varie la tension d'entrée. Des tensions rectangulaires peuvent donc être transformées en impulsions en dents.

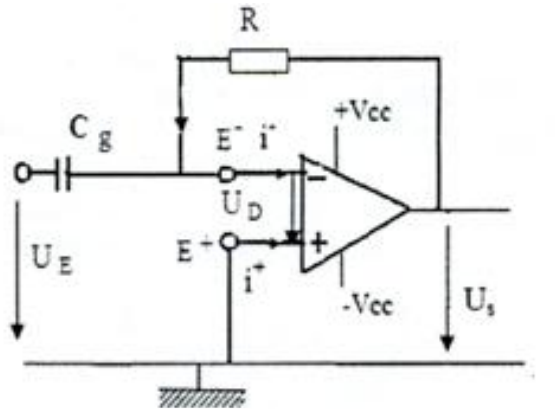


**Fig.8** Amplificateur dérivateur

# IV. Travail expérimental

## IV.1 Application du montage Sommateur

- Construire le montage Fig.6 ; avec $R_1 = R_2 = R_3$ ; après avoir calculé leurs valeurs pour que $U_s = 8V$, $U_{E1} = U_{E2} = 4V$ et la fréquence $f = 1Khz$ ;
- Identifier les tensions observées sur l'oscilloscope ;
- Tracer les courbes $U_E = f(t)$ et $Us = f(t)$. Bien noter leurs valeurs maximales ainsi que la période T du signal ;
- Que se passe-t-il si on prend $R_1 = R_2$ et $R_3 = 47$ KOhm ?

## IV.2 Application du montage intégrateur

**IV.2.1** Construire le montage Fig.7 ; avec R= 10 Ohm et $C_g = 50$ nF (2×0.1µF en série).

- Sélectionner une tension d'entrée rectangulaire d'amplitude 50 mV et de fréquence $f = 1Khz$
- Identifier les deux tensions d'entrée et de sortie observées sur l'oscilloscope ;
- Tracer les courbes $U_E = f(t)$ et $U_s = f(t)$. Bien noter leurs valeurs maximales ainsi que la période T du signal.

**IV.2.2** Varier la capacité de la contre réaction $C_g = 0.2\mu F$ (2×0.1µF en parallèle) et répéter les mesures de l'expérience IV.2.1 et tracer les courbes visualisées sur l'oscilloscope.

**IV.2.3** Comparer les courbes de la tension de sortie des expériences IV.2.1 et IV.2.2 et interpréter les résultats.

## IV.3 Application du montage dérivateur

**IV.3.1** Construire le montage Fig.8 ; avec R= 10 Ohm et $C_g = 50nF$ (2×0.1µF en série) ;

- Sélectionner une tension d'entrée rectangulaire d'amplitude 1V et de fréquence $f = 1Khz$ ;
- Tracer les courbes $U_E = f(t)$ et $U_s = f(t)$. Bien noter leurs valeurs maximales ainsi que la période T du signal.

**IV.3.2** Varier la capacité de la contre réaction $C_g = 0.2\mu F$ (2×0.1µF en parallèle), répéter les mesures de l'expérience IV.3.1 et tracer les courbes visualisées sur l'oscilloscope.

**IV.3.3** Comparer les courbes de la tension de sortie des expériences IV.3.1 et IV.3.2 et interpréter les résultats.

# V. Conclusion générale

Faites vos conclusions sur les montages que vous avez réalisé.

# TP N°3: Introduction au logiciel Matlab

## I. Objectifs pédagogiques

Etude des commandes de base du logiciel Matlab et réalisation de programmes Matlab pour la simulation numérique des systèmes asservis.

## II. Introduction

**Matlab** (abréviation de « **Mat**rix **Lab**oratory ») est un environnement informatique conçu pour manipuler aisément des matrices à l'aide de fonction préprogrammées (addition, multiplication, inversion, décompositions, déterminants…), en s'affranchissant des contraintes des langages de programmation classique :

- Plus de déclarations de variables.
- Plus de phase d'édition-compilation-exécution.

L'élément de base est une matrice dont la dimension n'a pas à être fixée. **Matlab** est un outil puissant qui permet la résolution de nombreux problème en beaucoup moins de temps qu'il n'en faudrait pour les formuler en C ou en Pascal.

S'il est parfaitement adapté à l'**Automatique** et au Traitement du Signale, sa facilité d'emploi avec des nombres complexes et ses possibilités d'affichages graphiques en font un outil intéressant pour bien d'autres type d'applications. De plus, des « toolboxes » (boite à outils) sont disponibles dans de nombreux domaines (Contrôle, Robotique, traitement de signal, traitement d'image, optimisation,…). Cet aspect modulaire est l'un des plus grands atouts de **Matlab** : l'utilisateur peut lui même définir ses propres fonctions, en regroupant des instructions **Matlab** dans un fichier portant le suffixe ".m".

**Matlab** peut être donc considéré comme un langage de programmation au même titre que C, Pascal ou Basic.

C'est un langage interprété, c'est-à-dire, que les instructions sont exécutées immédiatement après avoir été tapées.

**Matlab** permit de développer des algorithmes rapidement, de visualiser des données (sous forme de graphiques 2D ou 3D et d'images, avoir de séquence d'image), et de réaliser des interfaces graphiques.

## III. Généralités et prise en main

### III.1 Démarrage, quitter

Pour démarrer **Matlab**, il suffit de cliquer dans l'icône "**Matlab**" si vous êtes sous Windows. L'espace de travail de **Matlab** se présente alors sous la forme d'une fenêtre affichant un prompt ">>" à la suite duquel vous pouvez taper une commande qui sera exécutée après avoir tapé sur la touche "return" (retour chariot).

En haut de cette fenêtre se trouve une barre de menu qui vous permet d'ouvrir un fichier texte, de définir certaines variables de travail et surtout d'accéder à l'ensemble des fichiers d'aides.

Vous pourrez quitter la session avec la commande *quit*.

### III.2 L'espace de travail (workspace)

Comme tout langage de programmation, **Matlab** permet de définir des données variables. Les variables sont définies au fur et à mesure que l'on donne leurs noms (identificateur) et leurs valeurs numériques ou leurs expressions mathématiques. **Matlab** ne nécessite pas la déclaration de type ou de dimension pour une variable. Les variables sont stockées dans l'espace de travail (ou workspace) et peuvent être utilisées dans les calculs subséquents.

### III.3 Langage interprété

**Matlab** est un langage interprété. Il n'est pas nécessaire de compiler un programme avant de l'exécuter et toute commande tapée dans la fenêtre de commande est immédiatement exécutée (après la frappe de return).

```
>> 2+2
ans = 4
>> 5^2
ans = 25
```

La réponse est affichée et stockée dans la variable **ans.**

La plupart des fonctions mathématiques usuelles sont définies dans **Matlab**, et ceci sous une forme naturelle (sin, cos, exp, ...) de même que certaines constantes (pi ...).

```
>> 2*sin(pi/4)
ans = 1.4142
```

### III.4 Variables

**Matlab** considère trois types de variables (réel, complexe et chaînes de caractères). En fait, toute variable de **Matlab** est une matrice (scalaire : matrice 1x1, vecteur : matrice 1xN ou Nx1).

On peut indiquer le nom de la variable dans laquelle le résultat doit être stocké (commence par une lettre, moins de 19 caractères).

Attention, **Matlab** prend en considération les majuscules (x est différent de X).

```
>> x = pi/4
x = 0.7854
```

Le nom de cette variable ainsi que le résultat sont affichés.

Un point virgule (;) à la fin de la ligne permet de ne pas afficher ce résultat. On peut taper plusieurs commandes par ligne, séparées par un point virgule.

```
>> x = pi/2; y = sin(x);
```

Lorsqu'une ligne de commande est trop longue on peut l'interrompre par trois points (...) et la poursuivre à la ligne suivante, on peut aussi mettre des commentaires dans une ligne de commande à l'aide du signe " % ".

### III.4.1 Effacement et liste des variables

La commande *clear* permet d'effacer une partie ou toutes les variables définies jusqu'à présent. Il est conseillé de placer cette commande au début de vos fichiers de commandes, en particulier lorsque vous manipulez des tableaux. Syntaxe :

```
>>clear var1 var2 var3 . . .
```

Si aucune variable n'est spécifiée, toutes les variables seront effacées.

La commande *who* affiche les noms de toutes les variables en cours.

### III.4.2 Variables complexes

**Matlab** travaille indifféremment avec des nombres réels et complexes. Par défaut les variables i et j sont initialisées à la valeur complexe. Naturellement si vous redéfinissez la variable i ou j avec une autre valeur elle n'aura plus la même signification.

```
>> z = 3 + 2*i
z = 3.0000 + 2.0000i
```

Les fonctions usuelles de manipulation des nombres complexes sont prédéfinies dans **Matlab** : real, imag, abs, angle (en radian), conj.

```
>> r = abs(z);
>> theta = angle(z);
>> y = r*exp(i*theta);
```

### III.4.3 Chaînes de caractères

Comme on l'a vu précédemment, toute variable de **Matlab** est une matrice. Cependant, il peut être utile de conserver des chaînes de caractères dans des variables, et cela se fait de façon tout à fait naturelle :

```
>> message = 'bienvenue sur Matlab'
```

On peut alors réaliser des manipulations de même type que pour les vecteurs

```
>> message = [message, ' version 4.1']
message = bienvenue sur Matlab version 4.1
```

On peut convertir des nombres en chaîne de caractères à l'aide des fonctions num2str, int2str, sprintf. Par exemple :

```
>> message = ['pi vaut ', num2str(pi)]
message = pi vaut 3.142
```

**III.5 Vecteurs, matrices et leur manipulation**

On peut spécifier directement une matrice sous la forme d'un tableau, l'espace ou la virgule sépare deux éléments d'une même ligne, les points virgules séparent les éléments de lignes distinctes.

```
>> A = [ 1, 2, 3 ; 4, 5, 6 ; 7, 8, 9 ]
   A = 1 2 3
       4 5 6
       7 8 9
```

Les éléments d'une matrice peuvent être n'importe quelle expression de **Matlab** :

```
>> x = [ -1.3, sqrt(3), (1+2+3)*4/5 ]
   x = -1.3000 1.7321 4.8000
```

Les éléments d'une matrice peuvent être référencés par leurs indices :

```
>>x(2)
ans = 1.7321
>>x(5) = abs(x(1))
x = -1.3000 1.7321 4.8000 0.0000 1.3000
```

On peut remarquer que la taille du vecteur x a été ajustée en remplissant les éléments non précisés par 0.

On peut aussi créer des matrices avec les fonctions *zeros, ones* et *eye*, ces fonctions créent des matrices de la taille précisée, respectivement remplies de zéros, de un, et de un sur la diagonale et des zéros ailleurs (eye = prononciation anglaise de I comme identité).

```
>> eye(2,3)
ans = 1 0 0
      0 1 0
>> ones(1,5)
ans = 1 1 1 1 1
```

On peut avoir des informations sur la taille d'une matrice :

```
>> size(x)
ans = 1 5
>> length(x)
ans = 5
```

On peut ajouter des lignes et des colonnes à des matrices déjà existantes.

```
>> r1 = [10, 11, 12];
>> A = [A ; r1]
A = 1   2   3
    4   5   6
    7   8   9
   10 11 12
```

```
>> r2 = zeros(4,1);
>> A = [A, r2]
  A = 1 2 3 0
      4 5 6 0
      7 8 9 0
      10 11 12 0
```

**III.6 L'opérateur " : "**

Cet opérateur, sous **Matlab**, peut être considéré comme l'opérateur d'énumération. Sa syntaxe usuelle est : *deb:pas:fin*

Il construit un vecteur dont le premier élément est *deb* puis *deb+pas, deb+2*pas...* jusqu'à *deb+n*pas* tel que *deb+n*pas* £ *fin* < *deb+(n+1)*pas*.

```
>> x = 0.5:0.1:0.85
   x = 0.5000 0.6000 0.7000 0.8000
```

Le pas d'incrémentation peut être omis, 1 est alors pris par défaut :

```
>> x =1:5
   x = 1 2 3 4 5
```

On peut aussi utiliser le " : " pour sélectionner des éléments d'un vecteur ou d'une matrice :

```
>> A(1,3) % Troisième élément de la première ligne de A
>> A(1,1:3) % Premier, deuxième et troisième éléments de la première ligne de A
>> A(1,:) % Tous les éléments de la première ligne
>> A(:,3) % Tous les éléments de la troisième colonne
>> A(:) % Vecteur colonne contenant tous les éléments de A lus colonne par colonne.
```

### III.7 Exemple d'extraction de sous-tableaux

*tableau(début :fin, début :fin)*

```
>> M = [1 2 3;11 12 13;21 22 23]
>> M(1:2,2:3)% permet d'extraire la matrice M=[2,3 ;12,13]
  ans = 2   3
        12 13
```

Si on utilise le caractère : seul, ça veut dire prendre tous les indices possibles. Exemple :

```
>> M(1:2,:)
ans =1     2  3
       11 12 13
```

C'est bien pratique pour extraire des lignes ou des colonnes d'une matrice. Par exemple pour obtenir la deuxième ligne de M :

```
>> M(2,:)
ans = 11 12 13
```

### III.8 Construction de tableaux par blocs

$$N = \left[\begin{array}{c|c} M & V \\ \hline U & 0 \end{array}\right]$$

Vous connaissez ce principe en mathématiques. Par exemple, à partir des matrices et vecteurs précédemment définis, on peut définir la matrice (4x4)

Pour faire ça sous Matlab, on fait comme si les blocs étaient des scalaires, et on écrit :

```
>> N=[M V
      U 0]
N =
     1   2   3  11
    11  12  13  12
    21  32  23  13
     1   2   3   0
```

Ou bien en utilisant le caractère ;

```
>> N=[M V; U 0]
```

Cette syntaxe est très utilisée pour allonger des vecteurs ou des matrices, par exemple si on veut ajouter une colonne à M, constituée par V :

```
>> M=[M V]
M =
     1   2   3  11
    11  12  13  12
    21  32  23  13
```

Si on veut lui ajouter une ligne, constituée par U :

```
>> M = [M;U]
M =
     1   2   3
    11  12  13
    21  32  23
     1   2   3
```

## III.9 Affichages alphanumériques et graphiques

### III.9.1 Affichage alphanumérique

On peut afficher des chaînes de caractères dans la fenêtre de commande :

```
>> disp(pi vaut 3.142)
```

Les fonctions sprintf et fprintf existe également (même syntaxe qu'en langage C).

```
>> fprintf('pi vaut %f\n',pi)
   pi vaut 3.141593
```

On peut aussi demander des valeurs à l'utilisateur :

```
>> var = input('Nombre d''itération de l''algorithme : ');
```

Matlab affichera la chaîne de caractère entrée en paramètre et attendra une réponse de l'utilisateur.

### III.9.2 Affichages graphiques 2D

Matlab permet un grand nombre de types d'affichage 2D et 3D, seuls les plus courants seront décrits ici. Utiliser help et les autres commandes d'aide pour affiner vos connaissances et vos graphismes.

La commande ***plot*** permet l'affichage d'une courbe 2D :

```
>>clear all ;clc
>> t= -pi:0.01:pi; % l'intervalle de temps
 >>y = sin(t);z=cos(t) ;
>>figure(1)      %Ouvrir une figure numéro 1
>>plot(t,y)      % Tracer la fonction y en fonction de t dans l'intervalle [-π,π] avec un pas
                 %de  0.01.
>>hold on       %Maintenir le graphe courant
>>plot(t,z)     %  tracer la fonction z en fonction de x dans le même graphe
>>title('Fonctions sin et cos')      % Afficher le titre du graphe
>>xlabel('Temps'); ylabel('y,z')     % Définir les axes x et y
>>grid                               %Affiche un quadrillage sur la courbe
 >>legend('sinus', 'cosinus')       % Afficher les légendes de x et y
 >>hold off                         %Pour revenir en mode normal
>>figure (2)                        %Ouvrir une figure numéro 2
>>plot(x,sin(x),x,cos(x),'r-.')     %Tracer deux courbe(sin(x) et cons(x) dans la même
                                    %figure(2)
>>legend('cosinus','sinus','racine')
 >>clf % effacer le fenêtre graphique et annule toutes les commandes de traçage (plot)
```

On peut tracer la courbe en *semilog* ou en log avec les fonctions *semilogx, semilogy* et *loglog.*

Pour attribuer des couleurs ou des styles du trait et des points de traçage des courbes vous taper ***help plot*** pour avoir de l'aide.

```
>> plot(x,cos(x),':',x,sin(x),'r-.',x,sqrt(x),'g--')
```

***Remarque:***

Les commandes **gtext** et **text** permettent le positionnement et l'écriture dans la zone graphique.

```
>>gtext({'This is the first line','This is the second line'})
```

L'instruction axis ([xmin xmax ymin ymax]) permet de définir les dimensions des axes x et y

```
>> axis ([-pi pi 0 10])
```

- **Afficher plusieurs graphiques (subplot)**

On peut effectuer plusieurs affichages sur une même figure en utilisant la commande subplot qui subdivise la fenêtre graphique. Sa syntaxe est :

*subplot(nombre_lignes,nombre_colonnes,numéro_subdivision)*

Les subdivisions sont numérotés de 1 à nombre_lignes*nombre_colonnes, de la gauche vers la droite puis de haut en bas.

**Exemple** :

```
>> subplot(3,2,1)
>> plot(x,y)
>> subplot(3,2,2)
 >> plot(x,y.^2)
```

| | |
|---|---|
| **1** | **2** |
| **3** | **4** |
| **5** | **6** |

### III.10 Instructions de contrôle

Comme dans la plupart des langages il existe des instructions de contrôle de la forme **for**, **while** ou **if**.

- **Instructions conditionnelles *if***

```
if condition logique
instructions ;
elseif condition logique
instructions ;
...
else
instructions
end
```

- **Structures de contrôle boucles *for***

```
for variable = valeur début: pas: valeur fin
instructions ;
end
```

- **Structures de contrôle boucles *while***

```
while condition logique
instructions ;
end
```

### III.11 Fichier Scripts

Il est parfois (souvent) souhaitable, pour ne pas avoir à taper plusieurs fois une même séquence d'instructions, de la stocker dans un fichier. Ainsi on pourra réutiliser cette séquence dans une autre session de travail. Un tel fichier est dénommé script.

Il suffit d''ouvrir un fichier avec le menu *file*, *new*, et de taper la séquence de commande dans ce script ensuite sauvegarder le fichier avec une extension " *.m* " (*nom_fich.m*). En tapant le nom du fichier sous Matlab (Workspace), la suite d'instructions s'exécute.

Les variables définies dans l'espace de travail sont accessibles pour le script. De même les variables définies (ou modifiées) dans le script sont accessibles dans l'espace de travail.

On peut (doit) mettre des commentaires à l'aide du caractère " % ". Toute commande située après "% " n'est pas prise en compte par Matlab, jusqu'à la ligne suivante.

## IV. Résumé des commandes MatLab

On a regroupé par sujet d'intérêt les différentes commandes et fonctions de MatLab. Il en existe d'autres que l'on pourra trouver dans les différentes boîtes à outils (toolbox) qui viennent compléter le logiciel de base.

| Opérateurs sur les Matrices | | Opérateurs sur les Tableaux | |
|---|---|---|---|
| + | addition | + | addition |
| - | soustraction | - | soustraction |
| * | multiplication | .* | multiplication |
| ^ | puissance | .^ | puissance |
| / | division à droite | ./ | division à droite |
| \ | division à gauche | .\ | division à gauche |
| ' | transpose conjugué | | |
| .' | transpose | | |
| kron | produit de Kronecker | | |

| Gestion des variables et de l'espace de travail | |
|---|---|
| who | affiche les variables courantes |
| whos | affiche les variables courantes, format long |
| save | Sauve l'espace de travail sur disque |
| load | restaure l'espace de travail à partir du disque |
| clear | efface les variables et fonctions de la mémoire |
| pack | réorganise la mémoire |
| size | renvoie la taille d'une matrice |
| length | renvoie la longueur d'un vecteur |
| disp | affiche une matrice de texte |

| Caractères spéciaux | |
|---|---|
| = | assignation |
| [ ] | définition de matrices ou vecteurs; |
| ( ) | gère la priorité des opérations arithmétique |
| . | point décimal |
| .. | directory parent |
| ... | indique une ligne suite |
| , | séparateur d'arguments ou d'instructions |
| ; | fin de lignes (matrices) ou suppression de l'affichage |
| % | commentaires |
| : | manipulation de sous matrices ou génération de vecteurs |

| Instructions spécifiques | |
|---|---|
| input | indicateur d'attente d'entrée |
| keyboard | considère le clavier comme un fichier script |
| menu | génère un menu de choix pour l'utilisateur |
| pause | attente |
| function | définition de fonction |
| eval | exécute un chaîne de caractère |
| feval | exécute une fonction définie dans une chaîne |
| global | définit les variables comme globales |
| nargchk | valide le nombre d'arguments d'entrée |

| Opérateurs Relationnels | | Opérateurs Logiques | |
|---|---|---|---|
| < | inférieur à | & | et |
| > | supérieur à | \| | ou |
| <= | inférieur ou égal à | ~ | non |
| >= | Supérieur ou égal à | xor | ou exclusif |
| == | égal à | | |
| ~= | différent de | | |

| Fonctions trigonométriques | |
|---|---|
| sin, asin, sinh, asinh | |
| cos, acos, cosh, acosh | |
| tan, atan, tanh, atanh | |
| cot, acot, coth, acoth | |
| sec, asec, sech, asech | 1./cos(z), acos(1./z), 1./cosh(z), acosh(1./z) |
| csc, acsc, csch, acsch | 1./sin(z), asin(1./z), 1./sinh(z), asinh(1./z) |

| Fonctions mathématiques élémentaires | |
|---|---|
| abs | valeur absolu ou module |
| angle | argument d'un complexe |
| sqrt | racine carrée |
| real | partie réelle |
| imag | partie imaginaire |
| conj | complexe conjugué |
| gcd | PGCD |
| lcm | PPCM |
| round | arrondi à l'entier le plus proche |
| fix | troncature |
| floor / ceil | arrondi vers -∞ / vers +∞ |
| sign | signe de |
| rem | reste de la division |
| exp | exponentiel |
| log / log10 | log népérien / log décimal |

| Variables prédéfinies | |
|---|---|
| ans | réponse à une expression sans assignation |
| eps | précision de la virgule flottante |
| realmax | plus grand nombre flottant |
| realmin | plus petit nombre flottant positif |
| pi | π |
| i, j | [√(-1)] |
| inf | ∞ |
| NaN | Not a Number |
| flops | nombre d'opérations flottantes par seconde |
| nargin | nombre d'arguments d'entrée d'une fonction |
| nargout | nombre d'arguments de sortie d'une fonction |
| computer | type du calculateur |

| Gestions des commandes et des fonctions | |
|---|---|
| help | aide |
| what | Listing du nom des M_files présents |
| type | impression d'un M_file |
| lookfor | recherche d'une entrée dans le help |
| which | localise les fonctions et fichiers |
| demo | lance la démonstration |
| path | Défini les chemins d'accès aux fichiers et fonctions |
| cedit | paramètres d'édition d'une ligne de commande |
| whatsnew | affiche les fichiers README de la toolbox |
| info | information sur MATLAB et The MathWorks |
| why | renvoie une réponse aléatoire non 'neutre' |

| Analyse de données par colonne | |
|---|---|
| max | valeur max |
| min | valeur min |
| mean | valeur moyenne |
| median | valeur médiane |
| std | écart type |
| sort | tri en ordre croissant |
| sum | somme des éléments |
| prod | produit des éléments |
| cumsum | vecteur des sommes partielles cumulées |
| cumprod | vecteur des produits partiels cumulés |
| hist | histogramme |

# TP N°4: Analyse temporelle d'un système dynamique du 1er ordre

## I. Objectifs du TP

L'étudiant, après une étude théorique des systèmes proposés (1er ordre), devra réaliser un fichier exécutable sous Matlab afin d'analyser les réponses de ces systèmes à des entrées différentes et de déterminer leurs performances.

**Remarque :** Pour la programmation Matlab utiliser les fonctions indiquées ci-dessous en plus des autres instructions étudiées précédemment.

Soit un système du 1er ordre dont la fonction de transfert est $T(s) = \frac{k}{1+\tau s}$.

```
%%%%%%%% Analyse temporelle d'un système de 1er ordre %%%%%%%%%%%%%%%%%%%%%%%%

t=init:pas:fin ;                    % Définir un vecteur de fin points linéairement
                                    %  espacé d'un pas entre init et fin
%Détermination de fonction de transfert
num=[ k ] ;den=[ τ  1] ;
sys=tf(num,den) ;                   %  Création de la fonction de transfert

%Réponse impulsionnelle d'un système de premier ordre
y1=impulse(sys,t) ;                 %  Réponse à une impulsion unité
figure(1)                           % Ouverture d'une figure
plot(t,y1)                          %Traçage de la réponse impulsionnelle

%Réponse indicielle d'un système de premier ordre
y2=step(sys,t)                      %  Réponse indicielle du système
figure(2)
plot(t,y2) ;

%Détermination de la constante de temps graphiquement (tau) à partir de la réponse indicielle

hold on                             %Maintenir les graphes courants
ym=max(y2 ) ;                       %  Détermine la valeur maximale de y2(Rep.indicielle)
yt=0.63*ym ;                        %  Détermine la valeur de yt correspondante au temps de réponse
max_idx=min(find(y2>=yt)) ;         %  find : permet de trouver l'indice de l'élément auquel y2>=yt
tau=t(max_idx) ;                    %  tau est le temps correspondant à cet indice dans le vecteur t
plot([tau tau],[0 yt],'m--')        %  Tracer une ligne verticale à tr
str=sprintf('tau=\n%3.2f s',tau)    %  Créer l'étiquette volante
gtext (str)                         %  Permet de cliquer dans la fenêtre pour positionner le texte
hold off

%Réponse à une rampe d'un système de premier ordre
u=a*t ;                             %Détermination de la rampe u de pente a
y3=lsim(sys,u,t)                    % Réponse du système à une entrée u quelconque (t :temps)
figure(3)
plot(t,y3) ;
```

## II. Etudes théoriques

Soit le circuit RC donné par Fig.1, avec $E = 40V, R = 50\Omega$ et $C = 63\mu F$ (les conditions initiales sont nulles)

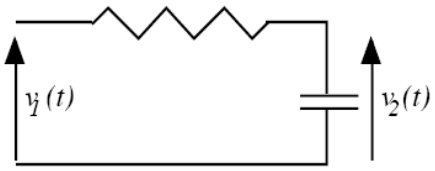


**Fig.1.** Circuit RC

On Considère la tension $v_1$ comme entrée $e(t)$, et la tension $v_2$ comme sortie $s(t)$.

1. Etablir l'équation différentielle qui régit le fonctionnement de ce circuit en appliquant la loi de Kirchhoff des tensions.

2. Donner la fonction de transfert de ce système.

3. Calculer la constante de temps et le gain statique correspondant au système.

4. Déterminer la réponse du système à l'application d'un échelon d'amplitude $E_0$. Tracer cette réponse d'une façon succincte. Calculer le temps de montée et le temps de réponse.

5. Déterminer la réponse du système à l'application d'une rampe de pente $a$. Tracer cette réponse d'une façon succincte. Calculer l'erreur de trainage (erreur de vitesse).

6. Tracer la réponse du système lorsque nous appliquons une impulsion, $e(t) = E_0.\delta(t)$ comme entrée du système.

## III. Analyse sous Matlab

En se basant sur le programme donné ci-dessus, écrire un fichier Matlab permettant de :

a. Calculer la fonction de transfert du circuit RC donné par Fig.1 ;

b. Tracer sa réponse à une impulsion unité ;

c. Tracer sa réponse indicielle à un échelon *e(t)=1* ;

d. Retrouver graphiquement la constante de temps tau à 63% de la réponse indicielle ;

e. Calculer le temps de montée et le temps de réponse. Afficher les dans la commande Windows et sur la figure de la réponse indicielle ;

f. Faites varier la constante de temps ( $\tau = 0.0016, \tau = 0.0032, \tau = 0.0064.$ ) du système et tracer les réponses indicielles correspondantes dans la même figure.

Faite vos conclusion sur l'influence de la constante de temps sur les allures obtenues ;

g. Tracer la réponse du système à une rampe de pente $a = 2$, et reprendre le même travail (étapes f). Calculer l'erreur de trainage ;

Commenter votre fichier Matlab et donner vos conclusions.

# TP N°5: Analyse temporelle d'un système dynamique du 2ème ordre

## I. Objectifs du TP

L'objectif de ce TP est d'exploiter les connaissances acquises, pendant le cours sur l'analyse temporelle d'un système dynamique du 2eme ordre. L'étudiant, après une étude théorique des systèmes proposés, devra d'abord réaliser un fichier exécutable sous Matlab afin d'analyser les réponses de ces systèmes à des entrées différentes et de déterminer leurs performances. La représentation d'un modèle sous Simulink va permettre ensuite de visualiser les réponses du système étudié.

## II. Rappels théoriques

Soit un système du 2eme ordre dont la fonction de transfert est donnée par :

$$T(s)=\frac{S(t)}{E(t)}=\frac{k}{\frac{s^2}{w_n^2}+\frac{2\xi}{w_n}s+1}$$

Avec $k$ : est le gain statique du système.

$w_n$ : est la pulsation naturelle(en rad/s).

$\xi$ : est le coefficient d'amortissement.

## III. Etudes théoriques

**Exemple 1 :**

On considère le circuit RLC série alimenté par une source de tension continue $v_1(t)$ (Fig.1), avec $R=20\Omega, L=3mH, et\ C=5\mu F$ (les conditions initiales sont nulles).

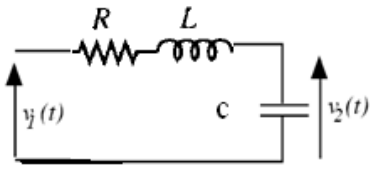


**Fig.1.** Circuit RLC

On Considère la tension $v_1(t)$ comme entrée $e(t)$ et la tension $v_2(t)$ comme sortie $s(t)$.

1. Déterminer la fonction de transfert de ce système (réseau électrique) ;
2. Mettre l'équation trouvée sous la forme de l'équation (1), et définir chaque terme ($w_n, \xi, k$) en fonction des éléments constitutifs du circuit RLC.
3. Donner les réponses du système à un échelon unitaire, à une rampe de pente a=2 et à une impulsion unité.

**Exemple 2 :**

On considère le système mécanique donné par (Fig.2). L'application d'une force $f(t)$ (entrée du système) sur le système provoque des oscillations de la masse $m$ suivant la direction $x(t)$ (la sortie du système). Le système est constitué, en plus de la masse $m$, d'un ressort de raideur $k$ et d'un amortisseur de coefficient $b$.

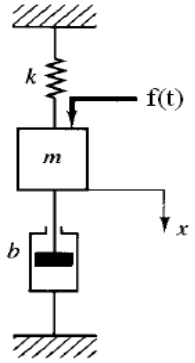


**Fig.2.** Système mécanique en vibration

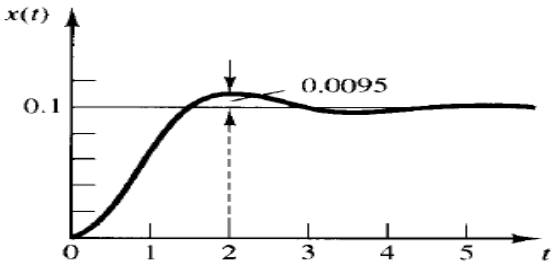


**Fig.3.** Réponse indicielle du système mécanique

L'équation différentielle correspondante à ce système est la suivante :

$$mx'' + bx' + kx = f \qquad \textbf{(2)}$$

1. Déterminer la fonction de transfert de ce système ;
2. A partir de la Fig.3, illustrant la réponse du système à un échelon $f(t) = 2N.m$, déterminer les valeurs numériques des paramètres du système $(k, m, b)$.

**Note :** le dépassement $D = 9,5\%$ correspondant à un amortissement $\xi = 0,6$.

## IV. Analyse sous Matlab

**Exemple 1 :**

1. Ecrire un programme sous un fichier Matlab, nommé tp3_1.m, qui permet de :
   a. Déterminer la fonction de transfert du circuit donné par Fig.1
   b. Tracer sa réponse à une impulsion unité ;
   c. Tracer sa réponse indicielle à un échelon unitaire;
      c1. Déterminer et afficher les performances du circuit $(t_r, \tau, t_m, t_p, D, \xi)$ et indiquer les sur la figure correspondante.
      c2. Pour qu'elle valeur de $R$ a-t-on $\xi = 1$ (amortissement critique) ;
      c3. Pour R= 0, 20, 60 Ω, tracer dans une même figure les trois réponses du système à un échelon unitaire et indiquer la valeur de $\xi$ pour chaque courbe.
   d. Tracer la réponse du système à une rampe pour $R = 60\Omega$.
      d1. Trouver l'erreur de trainage du système.
2. Commenter votre fichier Matlab et donner vos conclusions.

**Exemple 2 :**

1. Ecrire un programme sous un fichier Matlab, nommé tp3_2.m, qui permet de :

   a. Déterminer la fonction de transfert du système mécanique donné par Fig.2
   b. Tracer sa réponse à une impulsion unité ;
   c. Tracer sa réponse indicielle à un échelon $f(t) = 2N.m$;

      c1. Déterminer et afficher les performances du système $(t_r, \tau, t_m, t_p, D, \xi)$ et indiquer les sur la figure correspondante.

      c2. Tracer la réponse indicielle dans la même figure pour $\xi = 1, 0.4, 0.7$. Indiquer sur cette figure la valeur de $\xi$ pour chaque courbe.

   d. Tracer la réponse du système à une rampe de pente 2 ;

      d1. Trouver l'erreur de trainage du système.

2. Commenter votre fichier Matlab et donner vos conclusions.

## V. Analyse sous Simulink

Dans Simulink représenter le processus de **l'Exemple 2** par le bloc *Transfer Fcn*. Sauvegarder le sous le nom *tp3.mdl*

1. Observer sur l'oscilloscope l'évolution de sa réponse à un échelon d'amplitude 5, lorsque le paramètre d'amortissement $b$ varie. Pour cela, affecter le numérateur et le dénominateur avec les valeurs littérales : $[1/k]$ et $[m/k, b/k, 1]$.

2. Dans un fichier Matlab script, sauvegarder sous le nom *tp3param*, écrire les paramètres du système : $k = 3$ et $m = 5$ qui resteront fixes et $b = 2$ qui sera modifié avant chaque simulation, à savoir : $b = 2$ puis b 4, puis 6.

3. Dans **command windows** du Matlab, taper *tp3param* puis lancer la simulation après avoir choisi les paramètres de simulation : solver **ode45**, la durée de simulation (20s) et Diagnostic : Automatic solver parameter selection, none. Observer les résultats dans l'oscilloscope et tirer les performances du système.

4. De manière similaire laisser $k = 3$ et $b = 3$, mais faire évoluer la masse $m$ de 2.5 à 5. Expliquer les résultats obtenus.

5. Changer l'entré échelon par une rampe de pente 2 et visualiser les résultats.

# TP N°6: Analyse fréquentielle, stabilité et asservissement d'un système

## I. Objectifs du TP

- Observer et analyser la réponse fréquentielle d'un système asservi en utilisant les différentes présentations graphique : le plan de **Bode** et de **Nyquist**;

- Etudier la stabilité d'un système asservi en se basant sur son comportement fréquentielle en boucle ouvert et en utilisant le critère de **Routh** en cas de boucle fermée ;

- Améliorer les performances d'un système asservi en faisant une régulation Proportionnelle, Intégrale, Dérivée (correcteur PID) qui réunit les trois avantages, à savoir, la rapidité, la stabilité, et la précision.

## II. Rappels théoriques

- **Lieux de transfert d'un système dynamique linéaire**

Considérons un système linéaire d'ordre quelconque avec une entrée sinusoïdale ($e(t) = E_0 \sin(\omega t)$) et une sortie ($s(t) = S_0 \sin(\omega t + \varphi)$).

L'analyse harmonique (fréquentielle) de ce système consiste à faire le lien entre la fonction de transfert et la réponse de ce système à une sinusoïdale. Cette réponse sera caractérisée par deux paramètres : $Gain = \frac{S_0}{E_0}$ $dephasage : \varphi$

Ces deux paramètres dépendent de la pulsation $\omega$ de l'entrée :

$$\frac{S_0}{E_0} = |T(j\omega)| \qquad \varphi = \arg(T(j\omega)) \qquad \textbf{(1)}$$

Où $|T(j\omega)|$ est l'expression de la fonction de transfert du système dans laquelle on remplace la variable de Laplace $s$ par $j\omega$. Il existe trois types de représentation graphique du gain et du déphasage, Equ.1, en fonction de la pulsation $\omega$ :

***BODE*** se présente sous la forme de deux courbes :

$|T(j\omega)|_{dB}$ en fonction de $\omega$ (abscisses logarithmiques) : $|T(j\omega)|_{dB} = 20\log|T(j\omega)|$ **(2)**

$\varphi = \arg(T(j\omega))$ en fonction de $\omega$ (abscisses logarithmiques).

**BLACK** aussi appelé **NICHOLS** représente $|T(j\omega)|_{dB}$ en fonction de $\omega$. La courbe est graduée en $\omega$.

**NYQUIST** représente $T(j\omega)$ dans le plan complexe. La courbe est graduée en $\omega$.

- **Evaluation de la stabilité d'un système dynamique linéaire en régime sinusoïdale**

Cette partie montre que nous pouvons prévoir la stabilité d'un système en BF à partir de la représentation graphique du gain et du déphasage en BO.

Le critère graphique consiste à étudier la position de la courbe de la réponse harmonique en BO par rapport au point critique définie par

$$|T(j\omega)| = 1 = 0dB \qquad A\arg(T(j\omega)) = -180^\circ \qquad \textbf{(3)}$$

pour évaluer la stabilité de l'asservissement (boucle fermée).

Pour que la stabilité d'un système asservi soit assurée en toutes circonstances (perturbations comprises), il faut que sa courbe de réponse harmonique en BO passe suffisamment loin du point critique.

Les valeurs couramment admises pour assurer une stabilité suffisante sont :

Marge de gain $Gm$ : 8 à 12 dB

Marge de phase $Pm$ : de $30^\circ$ à $45^\circ$.

Ces marges de stabilité peuvent être lues directement dans les différents plans (**Bode, Black**).

**Critère du revers dans le plan de Nyquist**

Un système asservi linéaire est stable si en décrivant le lieu de **Nyquist** en BO dans le sens des fréquences croissantes, on laisse le point critique à sa gauche.

**Règle du revers dans le plan de Bode**

Soit $\omega_0$ la pulsation pour laquelle la courbe de gain coupe l'axe 0dB et $\omega_c$ la pulsation pour laquelle la courbe des phase passe par $-180^\circ$. L'asservissement est stable si $\omega_0 < \omega_c$.

***Remarque*** **:** un système instable n'a pas de marge de stabilité.

- **Les commande Matlab utilisés dans ce Tp**

Pour ce Tp vous pouvez utiliser les fonctions indiquées ci-dessus.

**Opération sur les fonctions de transfert :**

Il est possible d'assembler deux systèmes $T_1(s)$ et $T_2(s)$ de fonctions de transferts $T_1(s) = \frac{num1}{den1}$, $T_2(s) = \frac{num2}{den2}$ et soit $T(s) = \frac{num}{den}$ la fonction de transfert équivalente.

- Si les deux systèmes $T_1(s)$ et $T_2(s)$ sont en séries

$[num, den] = series(num1, den1, num2, den2)$ ou $T = T_1 * T_2$

- Si les deux systèmes $T_1(s)$ et $T_2(s)$ sont en parallèle

$$[num, den] = parallel(num1, den1, num2, den2) \text{ ou } T = T_1 + T_2$$

- Systèmes asservis avec un retour non unitaire

$$[num, den] = feedback(num1, den1, num2, den2, \pm 1) \text{ ou } T = feedback(T_1, T_2, \pm 1)$$

On met +1 si le retour est positif si non on met -1.

- Systèmes asservis avec un retour unitaire

$$[num, den] = feedback(num1, den1, 1, 1, \pm 1) \text{ ou } T = feedback(T_1, 1, \pm 1)$$

**Les fonctions correspondantes à la réponse fréquentielle de la fonction T(s) :**

```
% Le tracé de diagramme de Bode d'un système T(s)
T=tf(num,den);        % Fonction de transfert de T
bode (num,den)        % ou bode(T)
grid on

% Le calcul des marges de stabilité (gain et phase) à partir de diagramme de Bode
[gain,phase,w]=bode(T);
[MG,MP,wpi,wc]=margin(gain,phase,w);

% Ces marges sont indiquées sur le tracé de Bode à l'aide de
margin(gain,phase,w)

%Le tracé de Nyquist s'obtient
nyquist(T)  %ou [Re,Im]=Nyquist(T)
grid on
```

**Autres fonctions Matlab qui aide à déduire la stabilité du système T :**

```
roots(den)              % Permet de définir les pôles du système T
[z,p,k]=zpkdata(proc); % Déterminer les zéros, les pôles et le gain de T
p{1,1}                  % Définir le vecteur des pôles
pzmap(T);               % Placer les pôles et les zéros dans le plan complexe
```

# III. Etudes théoriques

**Exemple 1 :**

Un système du 1er ordre s'écrit de manière générale.

$$T(s) = \frac{k}{1 + \tau s} \tag{4}$$

Avec $k$ gain statique, $\tau$ : constante de temps

1. Déterminer la fonction de transfert de ce système en boucle fermée (retour unitaire);

2. Dans quels cas ce système peut il être stable en boucle ouverte et en boucle fermée (critère de Routh) ;

3. Donner l'expression du gain complexe et décibel de $T(s)$ pour $k=10$ et $\tau=1$, et déduire la pulsation de coupure $\omega_c$ ;

4. Tracer alors les allures des courbes dans le diagramme de Bode et Nyquist correspondant au système et en BO. Retrouver les valeurs de $k$ et $\tau$ ;

5. Déterminer les marges de stabilité à partir du tracé de Bode.

**Exemple 2 :**

Soit le système donné par le schéma suivant, Fig.1:

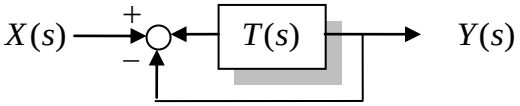


**Fig.1** Schéma fonctionnel du système en Boucle fermée.

Avec $$T(s)=\frac{k}{s(1+T_1\,s)(1+T_2\,s)} \qquad (5)$$

Ce système possède une intégration (pôle à l'origine). Il est de classe 1. on donne $T_1=0.5s$ et $T_2=1s$.

Ce système est inclus dans une boucle unitaire, munie d'un correcteur $C(s)$, Fig.2 :

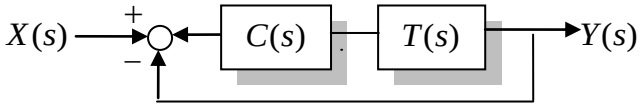


**Fig.2** Schéma fonctionnel du système avec contrôleur en boucle fermée.

On envisage un correcteur PID pour un bon suivi à une consigne variable, on désire en effet une erreur nulle (en régime permanent) à une consigne qui varie en rampe.

Le correcteur PID a pour fonction de transfert :

$$C(s)=\frac{k_c}{(1+1/T_i\,s+1/T_d\,s)} \qquad (6)$$

On prendra d'ailleurs $T_d=T_i/4$ (valeur courante adoptée par Ziegler Nichols).

1. Déterminer la fonction de transfert du système $T(s)$ en BF en fonction de k ;
2. Etudier et discuter la stabilité du système $T(s)$ en BF en utilisant le critère de Routh et cela suivant la valeur du gain k ;

3. On donne $k = 2$, $k_c = 0.8$ et $T_i = 8$, déterminer la fonction de transfert du système $(C(s)+T(s))$ en BF ;

## IV. Analyse sous Matlab

Comparez les résultats obtenus lors de l'étude théorique avec l'analyse faite sous Matlab.

**Exemple 1 :**

1. Ecrire fichier Matlab, nommé tp4_1.m permettant de :
   a. Déterminer la fonction de transfert $T(s)$ en BO et en BF (retour unitaire);
   b. Déterminer les pôles du système et déduire sa stabilité;
   c. Tracer le diagramme de Bode et Nyquist en BO, et en BF (retour unitaire);
2. Commenter votre fichier Matlab ;
3. Donner vos remarques et conclusions à partir des résultats acquises.

**Exemple 2 :**

1. Ecrire un fichier Matlab, nommé tp4_2.m permettant de:

**a- Analyser le système $T(s)$ donné par Fig.1 (sans correcteur) :**

a.1. Déterminer sa fonction de transfert en BO et en BF (retour unitaire);

a.2. Déterminer les pôles, les zéros et le gain du système en utilisant la fonction Matlab `zpkdata` ;

a.3. Analyser les pôles et les zéros du système en utilisant la fonction Matlab *pzmap,* et déduire sa stabilité ;

a.4. Tracer le diagramme de Bode et Nyquist du système en BO ;

a.5. Déterminer les marges de stabilité correspondantes. Discuter et comparer les résultats obtenus.

**b- Analyser le système $(C(s)+T(s))$ donné par Fig.2 (avec correcteur) :**

b.1. Déterminer sa fonction de transfert en BO et en BF (retour unitaire);

b.2. Analyser les pôles et les zéros du système en utilisant la fonction Matlab *pzmap,* et déduire sa stabilité ;

b.3. Tracer le diagramme de Bode et Nyquist en BO (retour unitaire);

b.4. Déterminer les marges de stabilité, discuter et comparer les résultats obtenus.

**c- Comparer les résultats obtenues de a et b pour le système $T(s)$ avec et sans correcteur en BF :**

c.1. Tracer dans la même figure les réponses indicielles (échelon unitaire) du système en BF avant et après correction. Analyser les résultats et spécifier les changements (performances) que le correcteur a apporté au système.

c.2. Tracer dans la même figure les réponses du système en BF à une rampe de pente a=1, avant et après correction. Analyser les résultats et vérifier que l'erreur statique en régime permanent est nulle.

2. Commenter votre fichier Matlab ;

3. Donner vos remarques et conclusions à partir des résultats acquises.