UNIVERSITE LIBANAISE
FACULTE DE GENIE
BRANCHE 2

Département de Génie Mécanique
Semestre 8
Projet d'Automatique 2

Le but de ce projet est d'établir un ensemble de fonctions sur Matlab, permettant d'étudier différents aspects d'un système échantillonné: stabilité, comportement fréquentiel, réponses impulsionnelle et indicielle et fonction de transfert composée.

1- Stabilité d'un système échantillonné.

Dans Matlab, la fonction **roots** retourne les racines d'un polynôme. Elle prend un vecteur comprenant les coefficients du polynôme. Pour un polynôme de degré *n* avec la variable *x*, *n+1* coefficients doivent être spécifiés, commençant par celui de x^n et se terminant par celui de x^0. Par exemple, taper dans Command Window : roots([1 2 3]) pour retrouver les racines de $x^2 + 2x + 3$.
Aussi dans Matlab, la fonction **abs** prend un nombre complexe et retourne son module. Par exemple, taper abs(-1+2i) et observer le résultat.

a) Établir une fonction sur Matlab, qui prend les coefficients d'un polynôme, et retourne les modules de ses racines, rassemblés dans un vecteur.

Rappel : pour établir une fonction sur Matlab, il faut ouvrir un script (Home → New → Script) et le commencer par une première ligne comme suit : **function [outputs] = function_name(inputs)** où outputs est l'ensemble des éléments retournés par la fonction, et inputs est l'ensemble des paramètres qu'elle requiert pour fonctionner. Par exemple, le code suivant établit une fonction qui s'appelle p et qui prend un nombre et retourne deux choses: sa valeur absolue et sa valeur multipliée par 2 :

```
function [b,c] = p(a)
b = abs(a) ;
c = 2*a ;
```

Un code d'une fonction doit être sauvegardé sous le format .m .
Noter que, selon le polynôme, il peut y avoir plusieurs racines et la fonction roots retournera donc un vecteur comportant autant d'éléments que de racines. Ce vecteur peut être traité comme tout autre, et l'accès à ses éléments se fait selon leurs indices. Par exemple, pour un vecteur *a*, *a(1)* signifie son premier élément.
Il serait utile de réviser les boucles dans Matlab: **for** ou **while**, ainsi que la fonction **size**.

b) Considérons un sysème caractérisé par une fonction de transfert $H(z) = \frac{N(z)}{D(z)}$. Etablir un code qui demande à l'utilisateur de donner les coefficients de N(z), puis ceux de D(z), et répond en spécifiant si le système en question est stable ou pas.

Rappel : la fonction **input** permet à la fois d'afficher sur l'écran un certain texte, et de prendre des valeurs de l'utilisateur. Par exemple, taper : a = input('donner les coefficients du numerateur') et répondre en tapant [1 2 3] puis taper a pour vérifier.
Il serait utile de réviser le syntaxe de la vérification de conditions dans Matlab en utilisant **if**.

2- Comportement fréquentiel d'un système échantillonné

Dans Matlab, la fonction **freqz** retourne le comportement fréquentiel d'un système numérique. Elle requiert la fonction de transfert du filtre, et la fréquence d'échantillonnage, et fournit un graphique comportant deux parties : gain en dB et phase. Nous nous intéressons au gain uniquement. La fonction de transfert est donnée sous forme de deux vecteurs: le premier comporte les coefficients du numérateur, et le second ceux du dénominateur.

Par exemple, pour la fonction de transfert $H(z) = \frac{z^2+1}{z}$ (un exercice déjà fait en classe), il suffit de taper freqz([1 0 1], [1 0]) pour obtenir le tracé de son comportement fréquentiel.

a) Observer, commenter et expliquer le tracé obtenu en tapant freqz([1 0 1], [1 0]).
b) Etablir un code qui demande à l'utilisateur les coefficients du numérateur et du dénominateur et lui affiche le comportement fréquentiel de la fonction de transfert en question.

3- Réponses impulsionnelle et indicielle d'un système échantillonné

Les fonctions **impulse** et **step** donnent respectivement les tracés des réponses impulsionnelle et indicielle d'un système. La fonction de transfert du système doit d'abord être établie en utilisant la fonction **tf** qui prend 3 paramètres: un vecteur comportant les coefficients du numérateur, un vecteur comportant les coefficients du dénominateur, et la période d'échantillonnage en secondes. Taper par exemple :

```
Hz = tf([0.72],[1,-0.1],0.2)
step(Hz)
```

a) Observer et commenter le résultat obtenu en tapant les deux lignes de code précédentes.
b) Les fonctions step et impulse peuvent aussi retourner à l'utilisateur les valeurs des réponses ainsi que les instants d'échantillonnage correspondants. Par exemple, taper :

```
Hz = tf([0.72],[1,-0.1],0.2)
[values, times] = step(Hz) ;
values
times
plot(times, values)
```

remarquer que la durée de l'étude de la réponse est choisie automatiquement, donc pour deux fonctions de transfert différentes, on pourra avoir deux durées d'étude différentes.
Observer et commenter la différence entre les tracés obtenus dans la partie a et la partie b.

c) Considérons pour des raisons de pratique que la dernière valeur donnée dans le vecteur values est la valeur finale de la réponse. En se basant sur cette supposition, établir un code qui demande à l'utilisateur les coefficients du numérateur d'une fonction de transfert, ses coefficients du dénominateur, et la période d'échantillonnage, et trace sur l'écran sa réponse indicielle, pui affiche une estimation de son temps de montée et de son erreur de position.
Définissons ici le temps de montée comme le temps au bout duquel le signal de sortie franchit pour la première fois son asymptote s'il la franchit, et il est indéfini sinon.
Rappel : la fonction **disp** permet d'afficher sur l'écran une chaîne de caractères sans attendre de réponse de l'utilisateur. La fonction **num2str** prend un nombre et le convertit en chaîne de caractères. Par exemple, disp(['il fait ', num2str(30), ' degres']) affiche « il fait 30 degres ».

4- Combinaison de fonctions de transfert

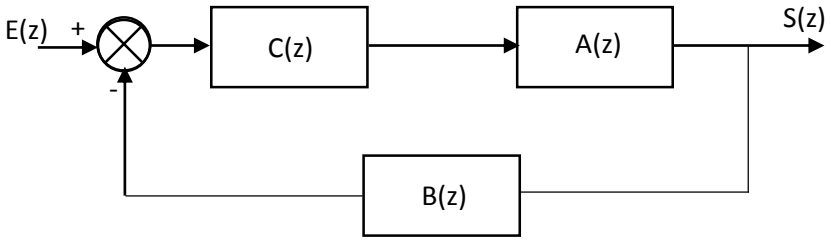
Pour multiplier deux polynômes dans Matlab, on peut avoir recours à la convolution de deux vecteurs comportant leurs coefficients, utilisant la fonction **conv**. Cette fonction prend les deux vecteurs de coefficients et retourne un vecteur comportant les coefficients du produit des polynômes.

a) Essayer la fonction conv avec trois exemples de votre choix, et vérifier les résultats obtenus.

b) Etablir une fonction qui prend en paramètres quatre vecteurs comportant les coefficients des numérateurs et des dénominateurs de deux fonctions de transfert $H_1(z)$ et $H_2(z)$, et retourne deux vecteurs comportant les coefficients du numérateur et du dénominateur de la fonction de transfert *H(z)* obtenue en multipliant les deux fonctions de transfert d'entrée : $H(z) = H_1(z)H_1(z)$.

5- Etude et correction d'un système

Considérer un schéma d'asservissement comme le suivant:



Etablir un code Matlab qui utilise les fonctions déjà établies et en ajoute pour:

- Demander à l'utilisateur la fonction de transfert A(z) et la période d'échantillonnage.
- Montrer à l'utilisateur le comportement fréqentiel de A(z), sa réponse indicielle et sa réponse impulsionnelle, son temps de montée et son erreur de position.
- Demander à l'utilisateur les fonctions de transfert B(z) et C(z).
- Calculer la fonction de transfert en boucle fermée.
- Refaire le deuxième point pour la fonction de transfert en boucle fermée.

Travail demandé

Il est demandé à chaque groupe de rendre un rapport tapé sur ordinateur, comportant les réponses à tous les points de ce projet. Le rapport doit comporter et expliquer également les codes Matlab établis et montrer comment ces codes ont été testés et évalués. Il doit montrer aussi des opérations faites préalablement à la main afin d'aider à établir les codes. Des exemples propres à chaque groupe doivent être inclus dans chaque partie, pour tester et valider les codes établis. La manière dont seront présentées les réponses, leur complétude et leur clarté sont des facteurs qui affectent la note attribuée au rapport.

Les rapports doivent être soumis par mail d'ici le 28 Juin 2019 (kareem-youssef@hotmail.com).

Des présentations orales notées se feront le 1 Juillet 2019 (questions sur le projet). Elles commenceront à 9h et se feront groupe par groupe, avec 10 à 15 minutes par groupe. Tous les membres de chaque groupe devront être présents. Un ordinateur portable devra être apporté avec chaque groupe, avec Matlab et les codes établis.

UNIVERSITE LIBANAISE
Faculté de génie - Branche 2 – Roumieh

Projet Automatique II

Présenté á
Dr. YOUSSEF Kareem

Préparé par :
RIZK Jean
GABRIELIAN Jhonny
KOLANDJIAN Anna Christina

Table des matières

1 STABILITE D'UN SYSTEME ECHANTILLONNE : 3

1.1 FONCTION ''ABSROOTS'' 3
1.2 CODE ''STABLE '' 3

2 COMPORTEMENT FRÉQUENTIEL D'UN SYSTÈME ÉCHANTILLONNÉ : 4

2.1 TRACE DE '' FREQZ([1 0 1] , [1 0]) '' 4
2.2 CODE '' FREQUENTIEL '' 4

3 REPONSES IMPULSIONNELLE ET INDICIELLE D'UN SYSTEME ECHANTILLONNE 5

3.1 REPONSE INDICIELLE DE LA FONCTION DE TRANSFERT DONNEE 5
3.2 FONCTION '' STEP '' VS FONCTION '' PLOT '' 5
3.3 CODE '' REPIND '' 6

4 COMBINAISON DE FONCTIONS DE TRANSFERT 7

4.1 FONCTION '' CONV '' 7
4.2 FONCTION '' CONVDEUXPOL'' 7

5 ETUDE ET CORRECTION D'UN SYSTEME 8

1 Stabilité d'un système échantillonné :

1.1 Fonction ''absroots''

Dans cette partie, on a été demandé de créer une fonction qui prend comme argument un vecteur représentant les coefficients d'une équation polynomiale et de donner en sortie le module des racines de ce polynôme sous forme de vecteur.
Pour cela, on s'est aidé des fonctions prédéfinies roots et abs : roots prend comme argument un vecteur des coefficients du polynôme et donne en sortie un vecteur contenant les racines du polynôme, puis abs calcul le module de chaque racine et les insèrent dans un vecteur en s'aidant d'une ''loop for'' pour le calcul de chaque module. La fonction totale a pris le nom de '' absroots ''.
Exemple comme test : x^3-2x^2+5x+3 a donné les modules : 2.4903, 2.4903 et 0.4838.

1.2 Code ''stable ''

Dans cette partie, on a été demandé de créer un code (script) qui prend comme demande de l'utilisateur de rentrer le numérateur et le dénominateur de la fonction de transfert et retourne une réponse indiquant si le système est stable ou pas.

On a procédé de la sorte :

- Demande de l'utilisateur de rentrer les coefficients du numérateur et du dénominateur sous forme de vecteurs.
- Création d'une fonction nommée '' stabilité '' qui prend ces 2 vecteurs et retourne un indicateur qui indique si la fonction est stable ou pas.
- Enfin, selon la valeur de l'indicateur, une phrase qui dit que le système est stable ou ne l'est pas apparait sur ''command window'' à l'aide de la fonction '' disp ''.

Pour créer la fonction '' stabilité '', on est revenu à la définition essentielle de la stabilité : une fonction est stable si les modules des pôles de la fonction sont tous inferieur à l'unité. De plus, nous devons de même voir si un pôle instable est simplifiable avec un zéro de la fonction, dans ce cas le pôle est éliminé, ceci a été exécuté à l'aide de 2 '' loop for '' et de 2 syntaxes '' if else '' comme le montre la fonction '' stabilité ''.

Pour tester le code :

On a eu recours à 2 fonctions de transfert, la première stable d'expression :

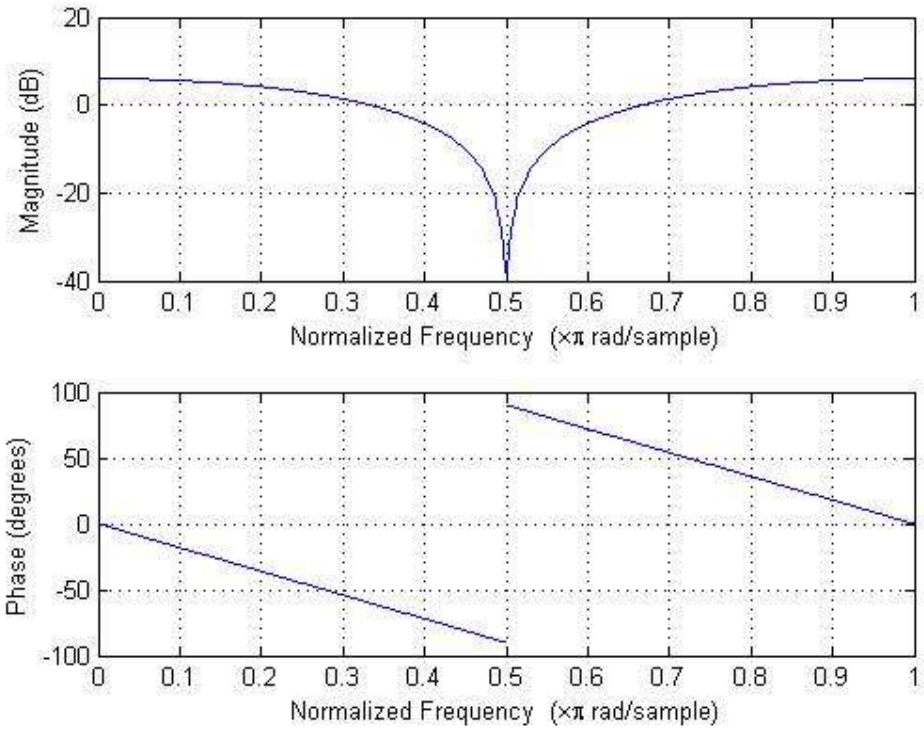
$H(z) = \frac{0.2}{z^2-1.2z+0.52}$

Et une deuxième instable d'expression : $H(z) = \frac{1}{z^2-1.2z+1.32}$

2 Comportement fréquentiel d'un système échantillonné :

2.1 Trace de '' Freqz([1 0 1] , [1 0]) ''

La fonction freqz prend 2 arguments, le vecteur coefficient du numérateur
et celui du dénominateur, et retourne un schéma fréquentiel de la fonction.
La fonction $\frac{z^2+1}{z}$ est une fonction coupe fréquence, qui associe un gain nul a une fréquence donnée pour éliminer son effet.
La courbe obtenue correspond parfaitement à celle étudié en classe :



.....

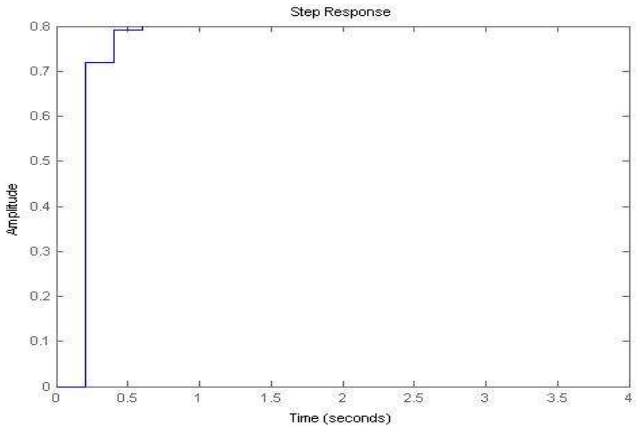
2.2 Code '' frequentiel ''

Dans cette partie , on a été demander d'ecrire un code qui affiche le comportement frequentiel d'une fonction que l'utilisateur doit donner .
La fonction est assez simple : l'utilisateur insere les vecteurs coefficients des numerateur et denominateur de la fonction de transfert et on utilise la fonction '' freqz '' en lui donnant comme arguments ces 2 vecteurs.
Le code est assez simple et l'exemple est celui de la partie a) (partie precedente).

3 Réponses impulsionnelle et indicielle d'un système echantillonné

3.1 Reponse indicielle de la fonction de transfert donnee

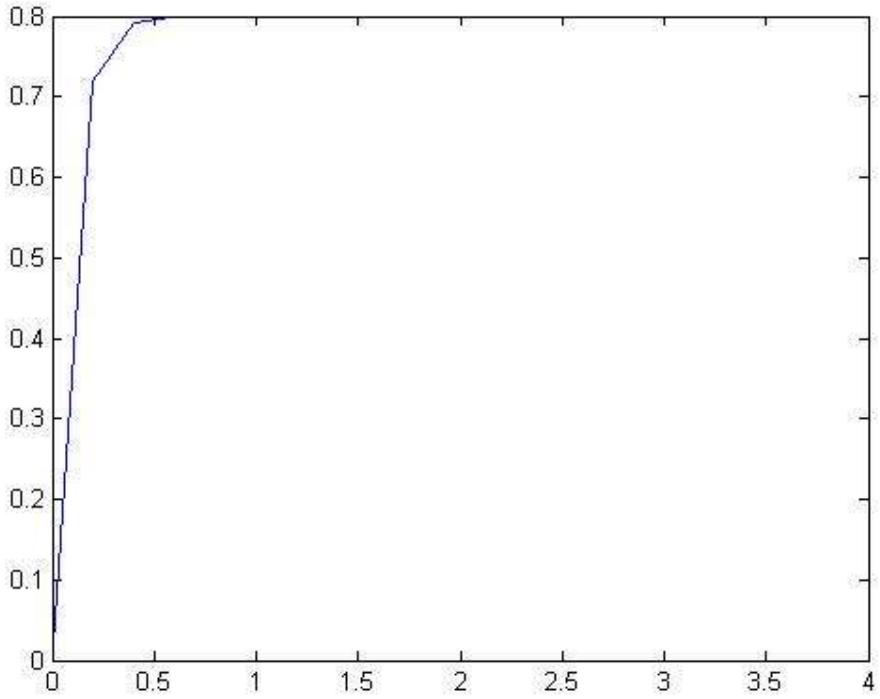
En écrivant les 2 lignes de code données, on a obtenu la réponse indicielle d'un système ayant la fonction de transfert : $\frac{0.72}{z-0.1}$



Le résultat obtenu montre que le temps de monte n'existe pas car c'est un système du 1ere ordre, tandis que la valeur finale vaut 0.8 ➔erreur de 0.2

3.2 Fonction '' step '' VS Fonction '' plot ''

En utilisant la fonction step, on a eu la réponse indicielle de la fonction de transfert, de sorte que parmi 2 temps d'échantillons la valeur de la fonction de transfert est celui du dernier échantillon calcule.
En utilisant la fonction plot, cette fonction opère en utilisant une interpolation linéaire entre 2 temps d'échantillons consécutifs.
Bien sûr, la fonction step modélise mieux la réponse indicielle d'un système discret.
Voici ce que la fonction plot donne :



3.3 Code '' repind ''

Dans cette partie, on a été demande d'écrire un script qui donnera la réponse indicielle d'une fonction de transfert donnée par l'utilisateur et de même de calculer l'erreur de position et le temps de monte (s'il existe).

Le code a été construit de la sorte :

- Dans un premier temps, on a demandé à l'utilisateur de rentrer le numérateur et le dénominateur sous forme de vecteurs de leurs coefficients, ainsi que le temps d'échantillonnage.
- Puis, on a établi la fonction de transfert à l'aide de la fonction '' tf '' et on a calculé le temps correspondant a chaque échantillon ainsi que la valeur de la fonction qui lui correspond en utilisant la fonction ''step'' , les temps et les valeurs sont chacune regroupe dans un vecteur respectivement nomme ' times ' et ' values '
- En admettant que l'asymptote a la courbe étant l'entrée unitaire du système, on a calculé l'erreur du système en prenant la dernière valeur du vecteur ' values ' (s~) et en appliquant la formule : $\text{Erreur} = \text{abs}\left(\frac{s\sim-1}{1}\right)$
- Pour le temps de monte, on a considéré que le temps de monte existe si et seulement si il existe au moins une valeur du vecteur ''values'' supérieure ou égale à 1. Par suite, à l'aide

des syntaxes 'for' et 'if', on a teste l'existence du temps de monte et on a prélevé l'indice de l'échantillon associe (s'il existe) pour enfin le calculer en multipliant l'indice par le temps d'échantillonnage.

- Finalement, tous les détails vont s'afficher sur le '' command window'' à l'aide de la fonction 'disp'.

Pour tester le code:

On a pris une fonction travaillée en classe et on a vérifié les valeurs obtenues : $\mathrm{H(z)} = \frac{0.16}{\mathrm{z^2 - 1.6z + 0.8}}$ avec Te = 0.1 s.

Similaire aux résultats du cours, on a obtenu t_m = 0.6s et erreur = 0.2.

4 Combinaison de fonctions de transfert

4.1 Fonction '' conv ''

Dans cette partie, on a testé, à partir de deux exemples, la fonction conv qui sert à multiplier deux polynômes et donne les coefficients de leur produit. Conv prend comme argument les vecteurs des coefficients des deux polynômes et donne en sortie le vecteur des coefficients de leur produit.

Exemple 1 : $(z-1) * (z^2 - 2z + 5) = z^3 - 3z^2 + 7z - 5$

Exemple 2 : $(z^2-1) * (z^2 - 2z + 5) = z^4 - 2z^3 + 4z^2 + 2z - 5$

4.2 Fonction '' convdeuxpol''

Dans cette partie, on a été demandé de créer une fonction qui multiplie 2 fonctions de transfert. La procédure suivie est très simple: on a juste multiplier les numérateurs ensemble à l'aide de la fonction conv et les dénominateurs ensemble aussi à l'aide de la fonction conv. La fonction créée, ''convdeuxpol'' donne en sortie le numérateur et le dénominateur du produit des 2 fonctions.

Pour tester le code:

On a utilisé les 2 fonctions suivantes : $\mathrm{H1(z)} = \frac{\mathrm{2z-}}{\mathrm{z-3}}$ et $\mathrm{H2(z)} = \frac{\mathrm{z^2-3z+10}}{\mathrm{z-4}}$

Leur produit vaut : $\mathrm{H(z)} = \frac{\mathrm{2z^3-11z^2+35z-50}}{\mathrm{z^2-7z+12}}$

5 Etude et correction d'un système

Dans cette partie, un code complet qui a pris le nom de ''codetotal'' a été établi.

La création de ce code n'a pas été d'une grande difficulté, puisque ce code est l'assemblage des codes déjà crée déjà dans les parties précédentes, et donc on a assemblé les codes déjà crée en adaptant les noms des variables.

La seule difficulté qu'on a réellement rencontrée était dans le calcul de H(z), fonction de transfert en boucle fermé.

Pour le calcul de H(z), on a procédé de la sorte :

- On a fait le calcul d'une fonction G(z) qui est le produit de A(z) et C(z) à l'aide de la fonction ''convdeuxpol'' déjà créée.
- Le numérateur de H(z) se calcul en multipliant le numérateur de G par le dénominateur de B à l'aide de la fonction ''conv''.
- Le dénominateur de H(z) étant égale a : $\mathrm{NgNb} + \mathrm{DgDb}$

  Et donc on a calculé chaque produit seul dans un premier temps, puis pour pouvoir sommer les deux vecteurs obtenus , ces 2 vecteurs doivent avoir une même dimension , et donc on a comparé les dimensions des deux vecteurs obtenus et on a ajouté des '0' à celui qui a la dimension la plus petite pour avoir 2 vecteurs de même dimension et pouvoir les sommer.

Pour tester le code:

On a considéré un cas simple avec :$\mathrm{A(z)} = \frac{1}{\mathrm{z}-0.1}$, $\mathrm{(z)} = 0.4(\mathrm{z}-1)$, B=1

H(z) obtenue est égale a : $\frac{0.4\mathrm{z}-0.4}{1.4\mathrm{z}-0.5}$.