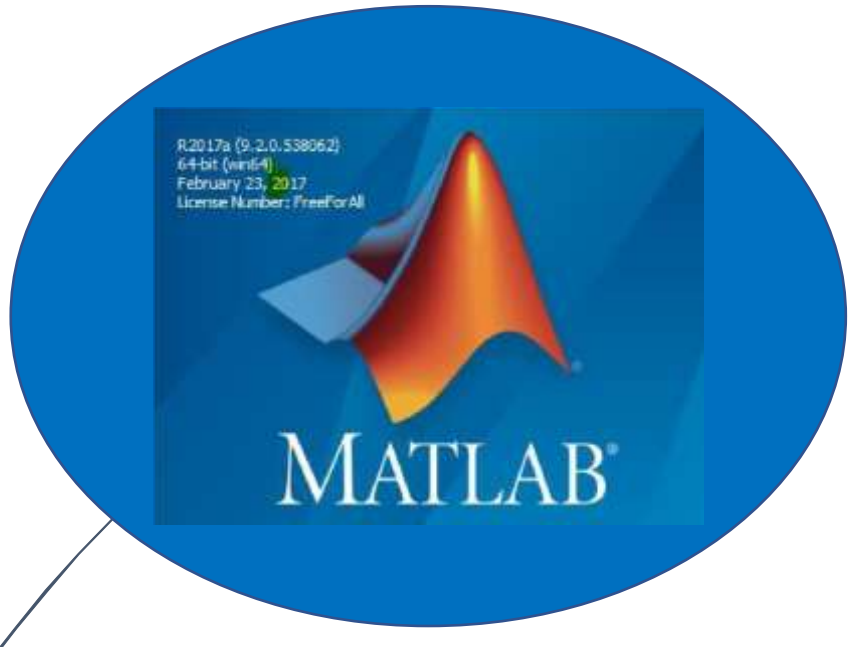
11/06/2018

GUIDE DES FONCTIONS MATLAB DE BASE EN SYSTEMES ASSERVIS



Mubikayi Kayambala Matthieu
UNIVERSITE DE KINSHASA

Table des matières

0. INTRODUCTION

Le logiciel MATLAB de la société THE MATH WORKS INC est originellement un logiciel de calcul matriciel. Il s'est spécialisé dans différentes disciplines au cours des années en fonction de la clientèle intéressée. Comme de nombreux automaticiens utilisaient MATLAB, il s'est spécialisé dans les fonctions liées à l'automatique, tant pour l'approche fréquentielle que pour l'approche temporelle. Outre le logiciel de base, l'on peut faire l'acquisition de quelques boites-à-outils et certaines d'entre elles sont spécifiques aux problèmes d'Automatique.

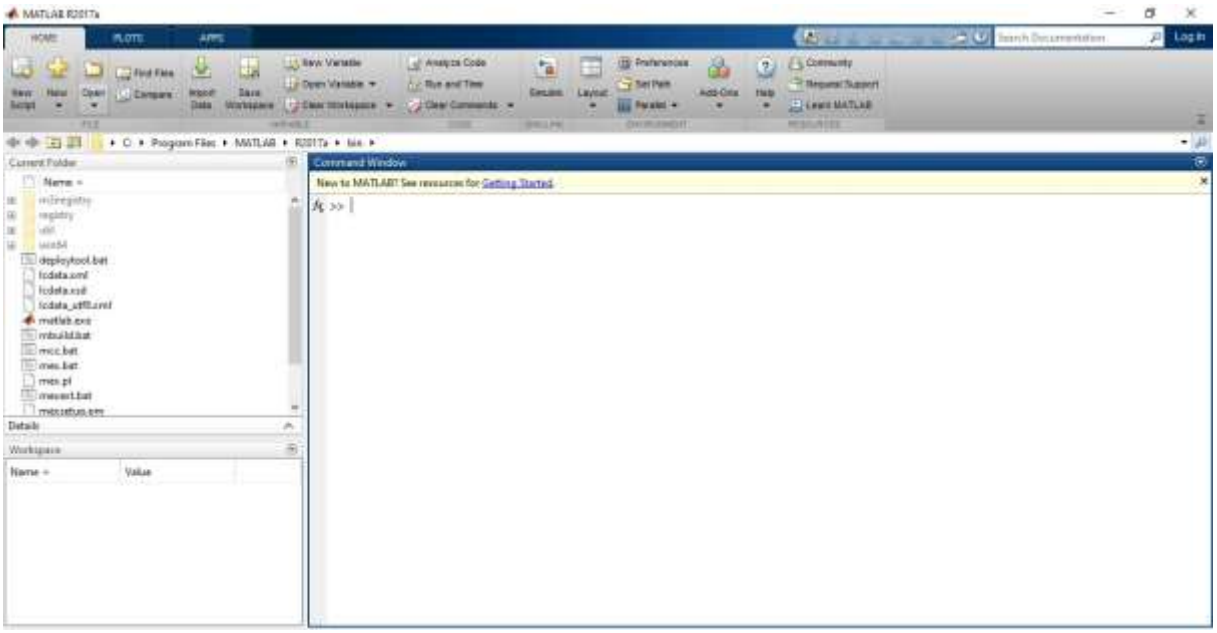
Sur ce, nous avons conçu ce guide dans le but d'aider les étudiants en génie électrique à se familiariser avec les commandes MATLAB orientées aux cours d'automatique et systèmes asservis.

Ainsi, nous aurons à présenter dans ce guide quatre chapitres, dont le premier se focalisera sur la présentation de l'environnement de travail MATLAB, et la syntaxe des fonctions mathématiques de base ; le second chapitre présentera les fonctions MATLAB pour le modèle d'état continu, et le troisième chapitre traitera des fonctions relatives à la commandabilité et observabilité, ensuite le quatrième chapitre nous présentera les fonctions relatives au modèle d'état discret.

CHAPITRE 1 : LES FONCTIONS MATHEMATIQUES DE BASES.

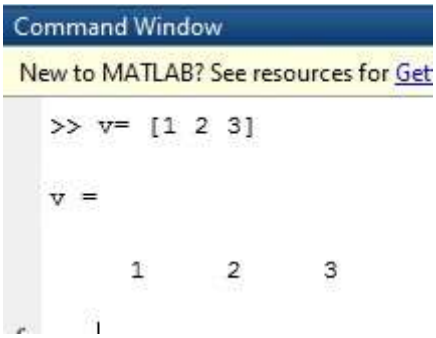
MATLAB est un logiciel de calcul matriciel, dont l'entité de base est une matrice. Aussi, les vecteurs et les scalaires ne sont vus que comme des matrices particulières. Dans ce qui suit, l'on considère uniquement des instructions tapées directement dans l'environnement de travail de MATLAB.

1.1 Présentation de l'environnement de travail



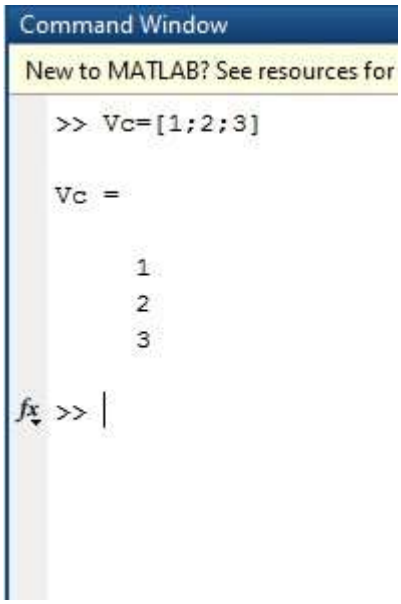
1.2 Ecriture des fonctions mathématiques de bases.

a) vecteur ligne



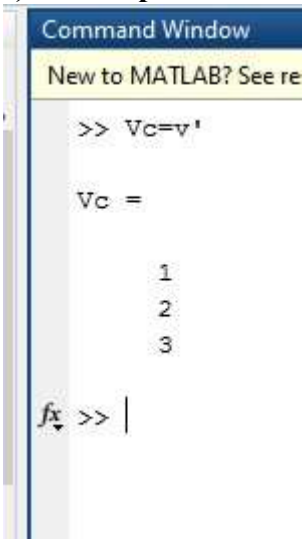
La première ligne avec le prompt ">>" correspond à l'instruction et les autres correspondent au résultat affiché par MATLAB. La variables v est alors enregistrée dans l'environnement et disponible à tout moment. La liste des variables actives est donnée par l'instruction Who.

b) vecteur colonne



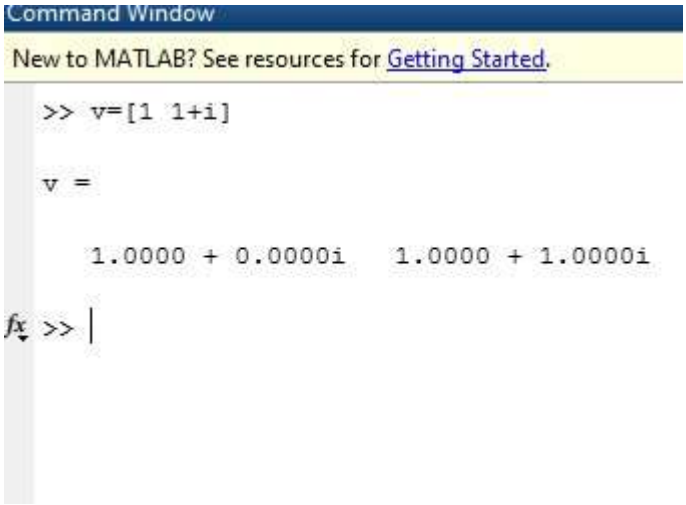
Il sied à noter que la différence entre le vecteur ligne v et le vecteur colonne Vc réside au niveau de la syntaxe.c.à.d. pour Vc on utilise les « ; » pour séparer les éléments.

c) la transposée d'un vecteur



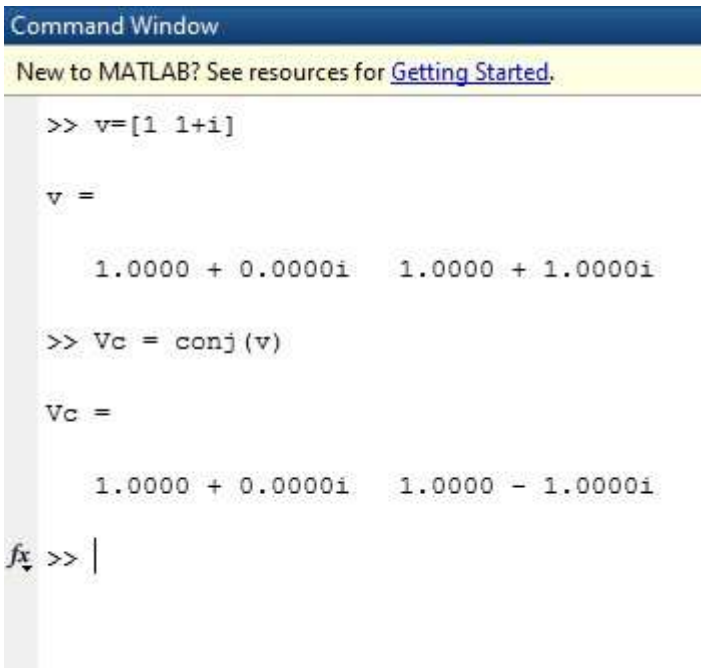
Notez que le vecteur v est déjà enregistré dans la mémoire.

d) nombre complexe



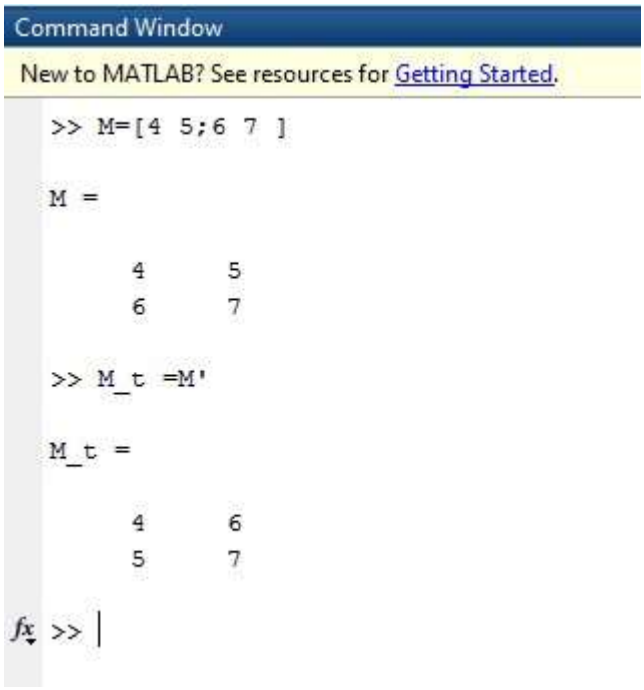
e**) conjuguer un nombre complexe**

Les composantes d'un vecteur ou d'une matrice peuvent être complexes. Ainsi, les variables `i` ou `j` sont prédéfinies dans MATLAB . Il importe donc de ne pas les écraser en redéfinissant des variables avec le même identificateur.



f**) Matrice d'odre n**

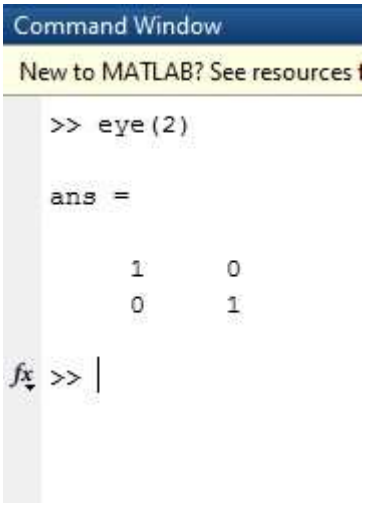
Dans cet exemple nous prenons n=2.



Avec « M_t » transposé de la matrice « M ».

g) matrice unitaire

la fonction « eye(n) » nous permet d'obtenir une matrice unitaire d'ordre n , dans notre exemple nous allons prendre n=2.

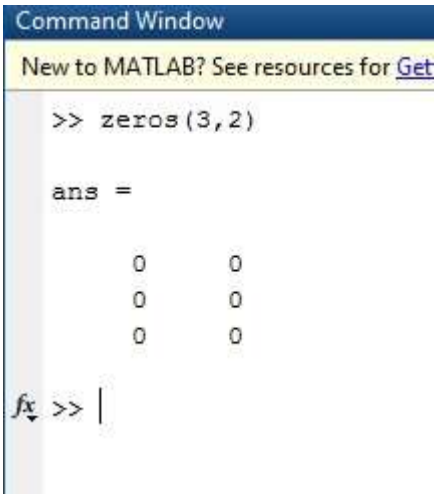


Pour plus d'infos sur la fonction tapez : `>> help eye`

h) Matrice nulle

La fonction zeros(m,n) nous permet d'obtenir une matrice nulle de m lignes et n colonnes.

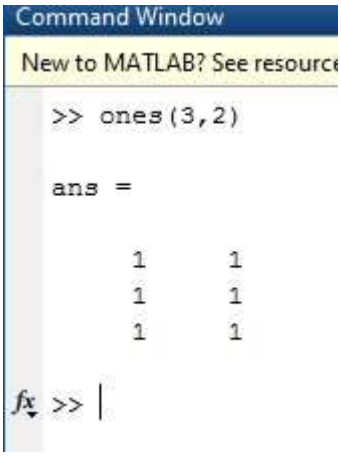
Exemple m=3, n= 2.



i)Matrice rempli des composants unitaire

La fonction ones(m,n) nous permet d'obtenir une matrice de m lignes et n colonnes.

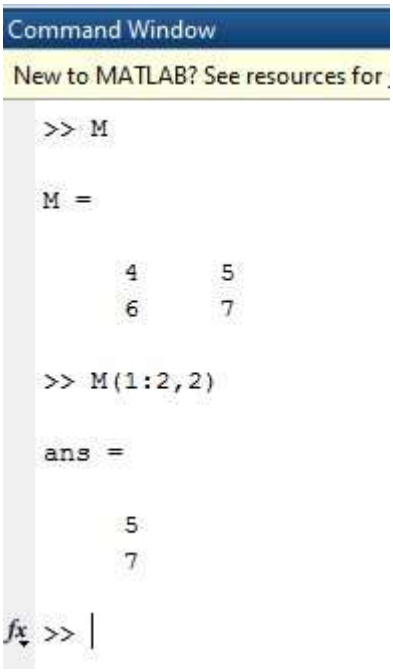
Exemple m=3, n=2



1.3 Les opérations sur les matrices

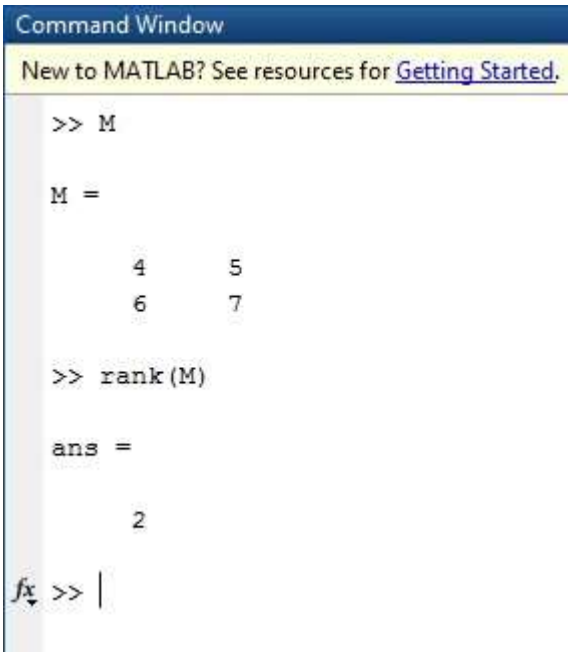
a) Extraction d'une sous matrice Exemple

:



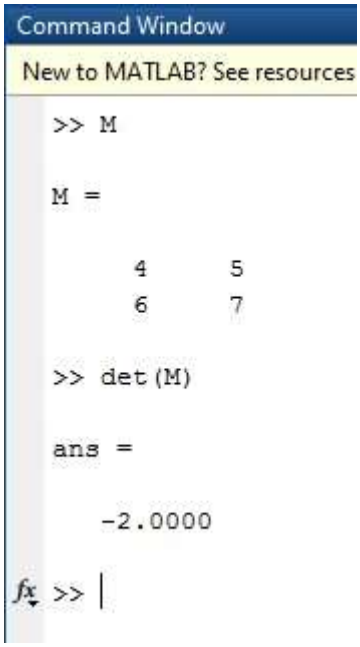
b) Rang d'une matrice

La fonction « rank( ) », nous permet de déterminer le rang de la matrice.

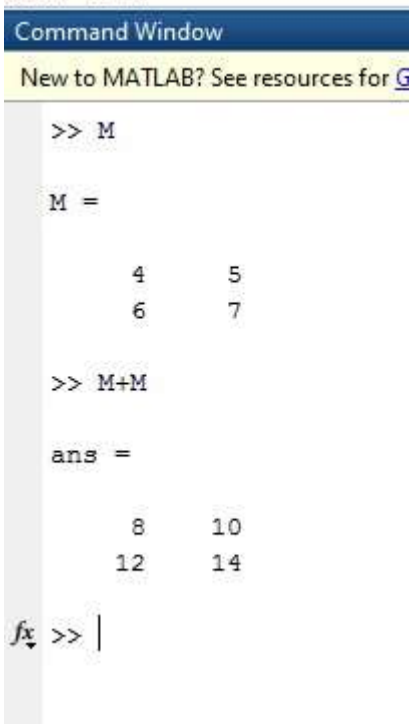


c) Déterminant d'une matrice

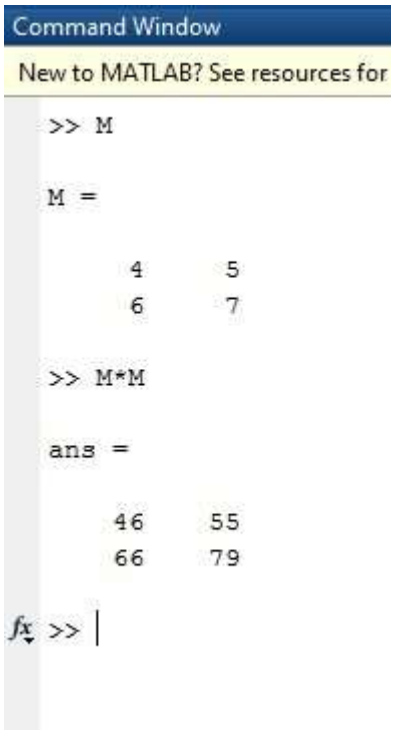
La fonction « det ( ) » nous permet de le déterminer.



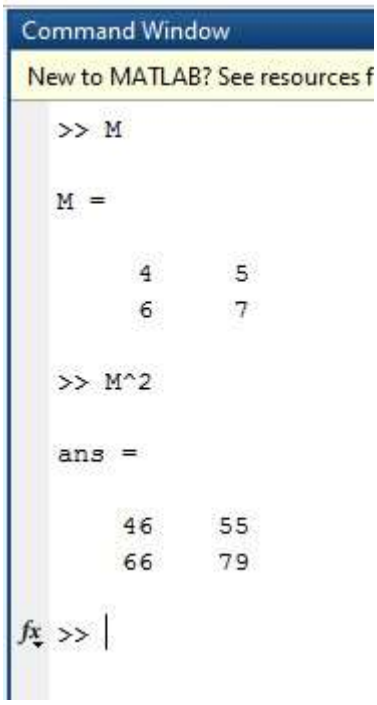
d) somme des matrices



e) Produit des Matrices

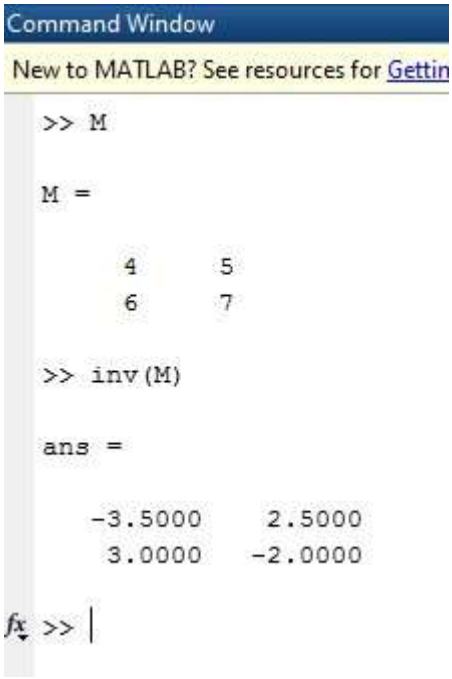


f) Puissance d'une matrice



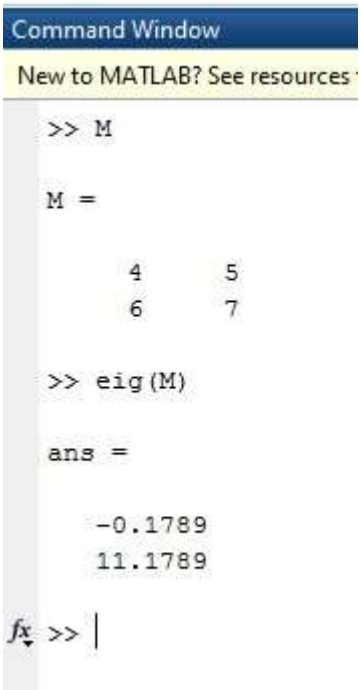
g) Inverse d'une matrice

L'inverse d'une matrice est réalisée à l'aide de la fonction « inv ( )».

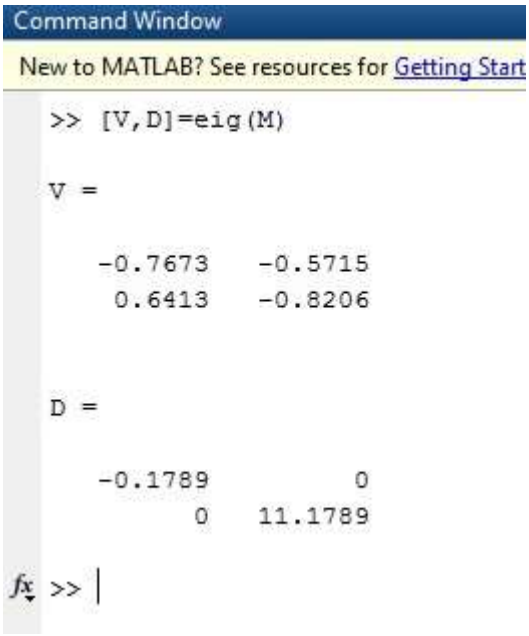


h) valeurs propres

Ils sont obtenues grâce à la fonction « eig() ».



i)Matrice Modale



D est la matrice diagonale obtenue en diagonalisant M.

j) Exponentiel d'une Matrice

Obtenu à l'aide de la fonction « expm ».

```
>> M=[4 5;6 7]

M =

     4     5
     6     7

>> expm(M)

ans =

   1.0e+04 *

    2.6346    3.1522
    3.7826    4.5259
```

1.4 Autres Fonctions

a) Calcul intégral

L'intégrale est calculé à l'aide de la fonction « int(f(x)) ».

Avec f(x) la fonction à intégrer.

```
>> syms x
>> int(x)

ans =

x^2/2
```

b) Transformée de laplace la fonction « laplace() » nous permet de calculer la transformée de laplace.

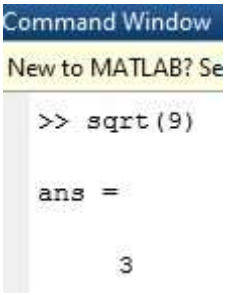
```
>> syms t
>> laplace(exp(-t))

ans =

1/(s + 1)
```

c) Transformée inverse de laplace la fonction « ilaplace() » nous permet de calculer la transformée inverse de laplace.

```
>> ilaplace(1/(s+1))

ans =

exp(-t)
```

d) calcul de la racine

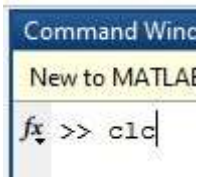
La fonction « sqrt ()» nous permet de calculer la racine d'un nombre.



NB

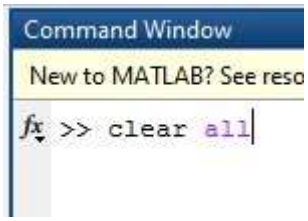
L'aide pour toute fonction peut être obtenue en tapant : help (nom_fonction).

La commande « clc »: Nous permet d'effacer toutes les commandes exécutées dans l'espace de travail.



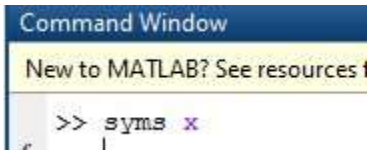
.

La commande « clear all » Nous permet d'effacer toutes les variables enregistrées.



.

La commande « syms variable » : nous permet de créer une variable sous forme de symbole.

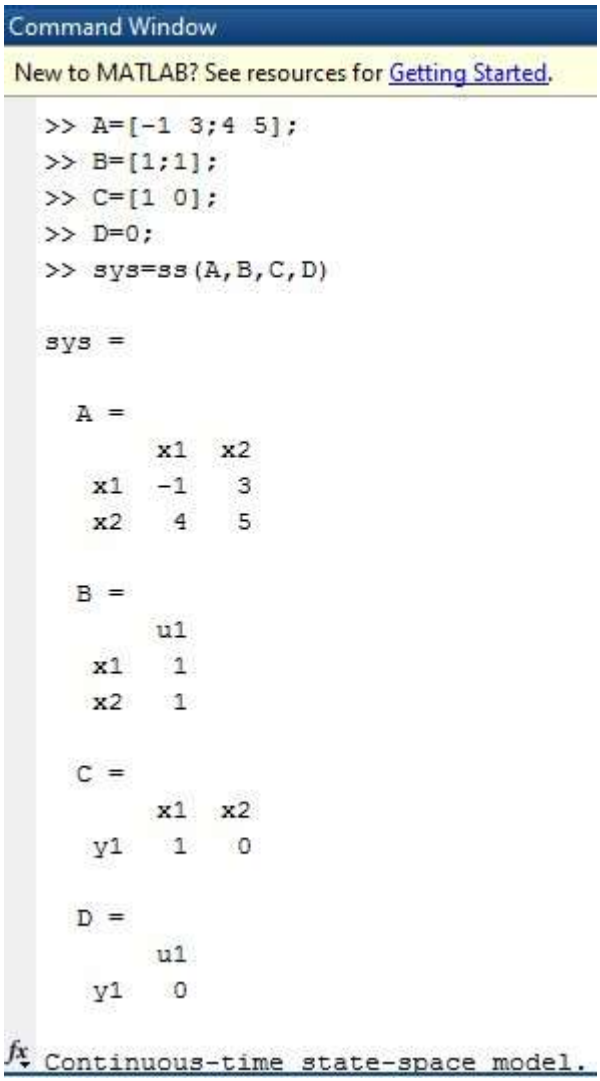


CHAPITRE2 : LES FONCTIONS RELATIVES AU MODELE D'ETAT CONTINU

Dans ce chapitre nous présenterons quelques fonctions relatives au modèle d'état pour un système continu.

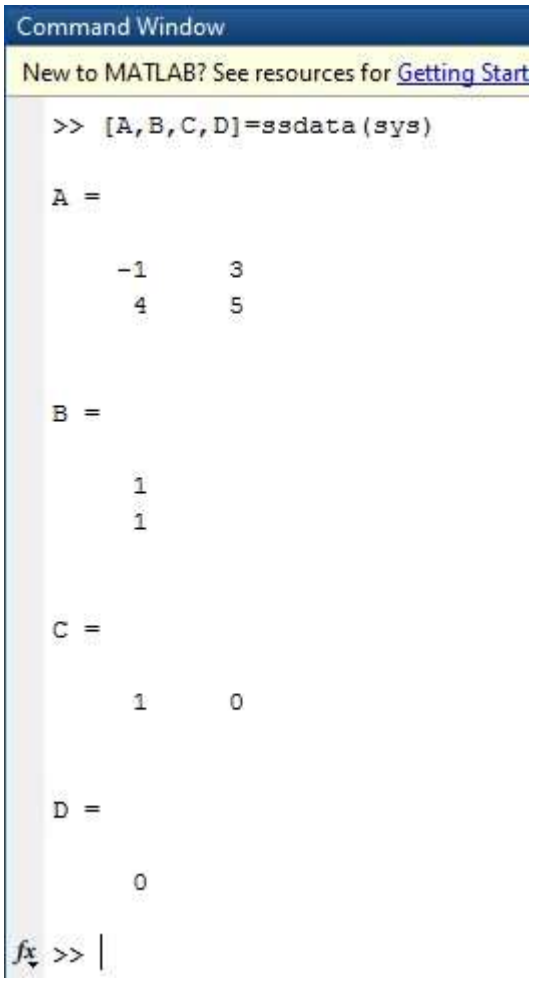
2.1 Fonction « ss() »

La fonction « ss() » ,nous permet de créer un espace-état ou un objet LTI.



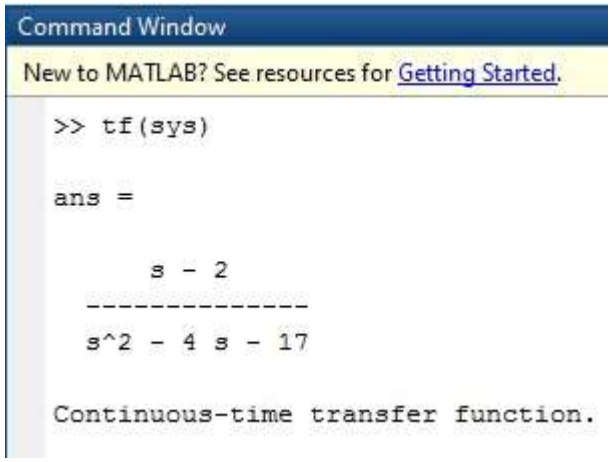
2.2 Fonction « ssdata() »

La fonction « ssdata() » est une fonction inverse de celle évoquée au point 2.1 , elle nous permet de récuperer les matrices A,B,C et D à partir de l'objet LTI crée : « sys ».



2.3 Fonction « tf ()»

La fonction « tf() »nous permet de calculer la fonction de transfert partant de l'objet LTI sys.



2.4 Fonction « ss2tf() »

Cette fonction nous permet de calculer la fonction de transfert partant des matrices A,B,C,D.

```
>> [num,den]=ss2tf(A,B,C,D)

num =

         0    1.0000   -2.0000


den =

     1    -4   -17
```

2.5 Fonction « impulse() » et « step() »

La fonction « impulse() » nous permet de tracer la réponse impulsionnelle du système.

New to MATLAB? See resources for Getting Started.

```
>> [A,B,C,D]=ssdata(sys)

A =

    -9   -10
     5     5


B =

     1
     1


C =

     1     0


D =

     0

>> impulse(sys)
```

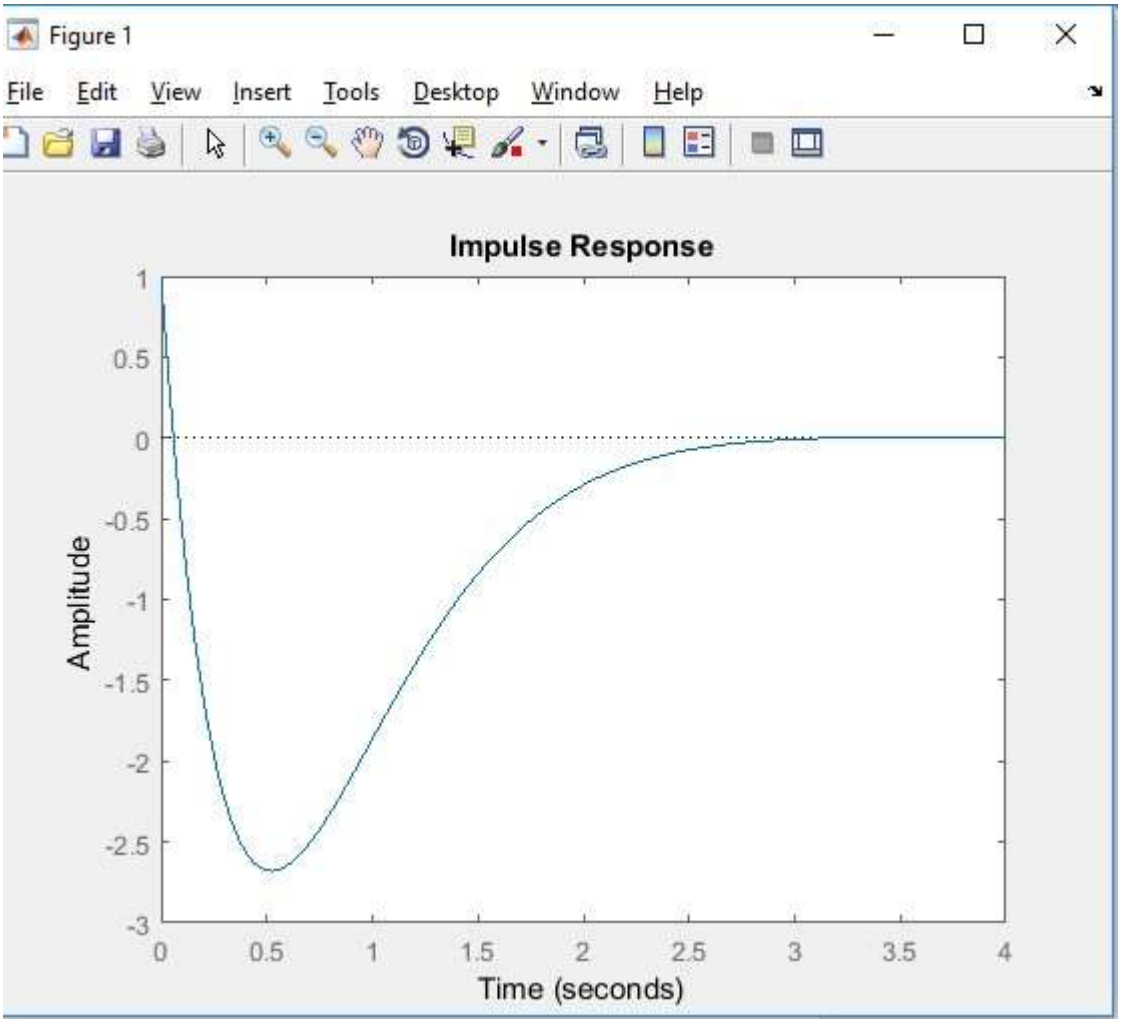


Figure 2.1 : Réponse impulsionnelle d'un système continu.

La fonction « step() » nous permet de tracer la réponse indicielle du système.

```
>> step(sys)
>>
```

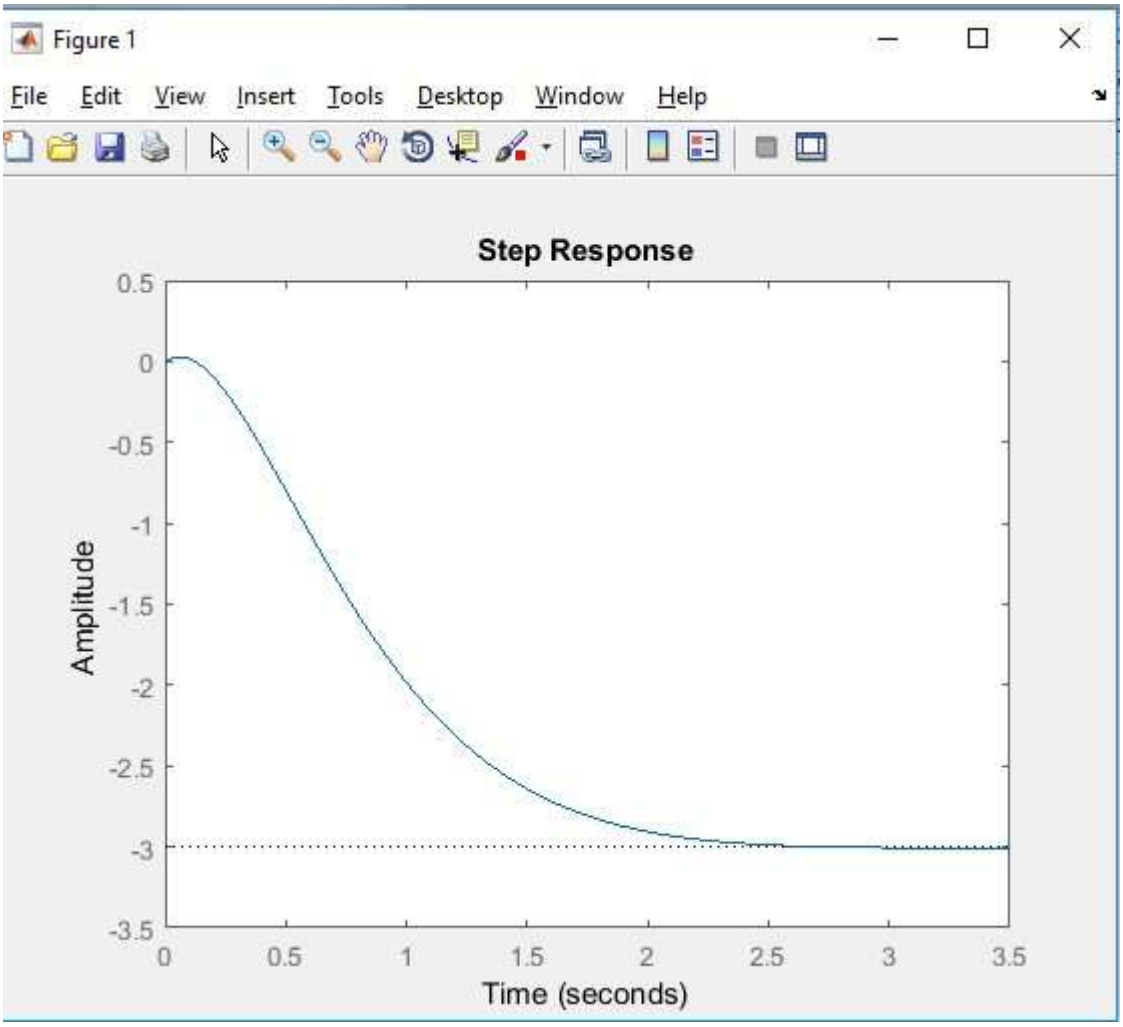


Figure 2.2 : Réponse indicielle d'un système continu.

2.6 Fonction bode()

La fonction « bode () » nous permet de tracer le diagramme de bode de notre système.

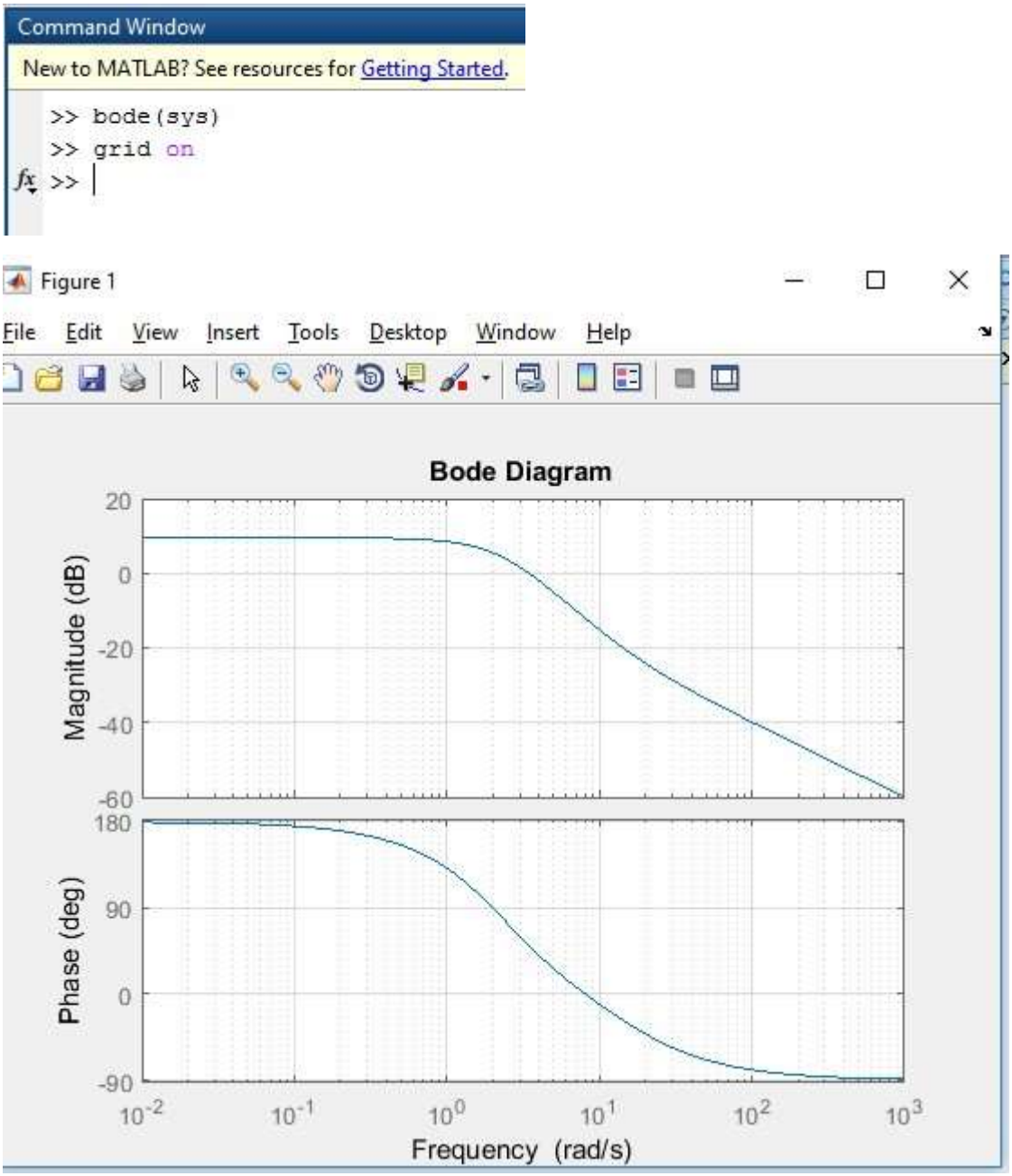


Figure 2.3 : Diagramme de bode.

2.7 Fonction nyquist()

la fonction « nyquist() » , nous permet de tracer la courbe de nyquist.

```
>> nyquist(sys)
>> hold on
>> grid on
>>
```

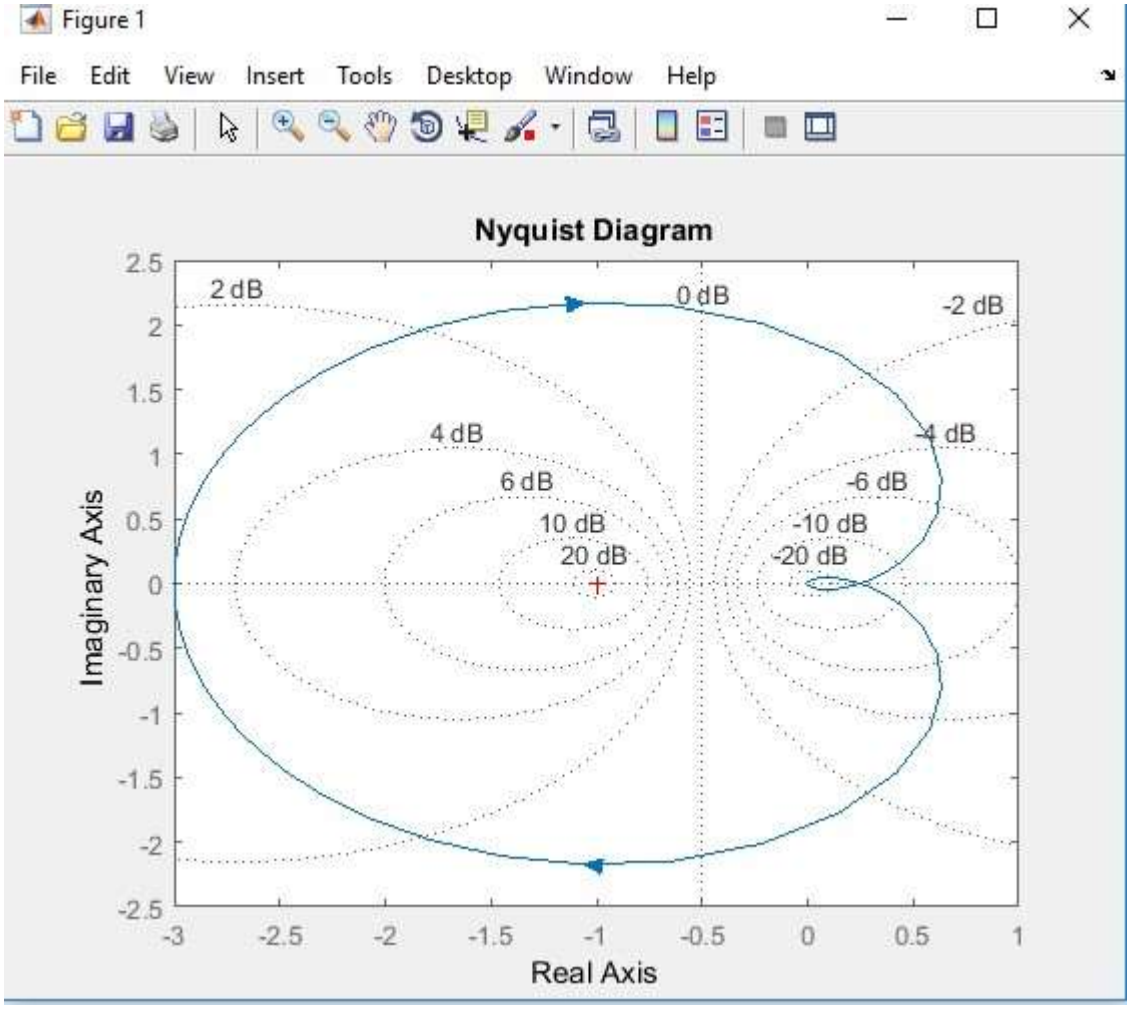


Figure 2.4 : Courbe de nyquist.

2.8 Fonction plot()

la fonction plot() nous permet de tracer toute sorte de courbe.

exemple : y= sin(x)

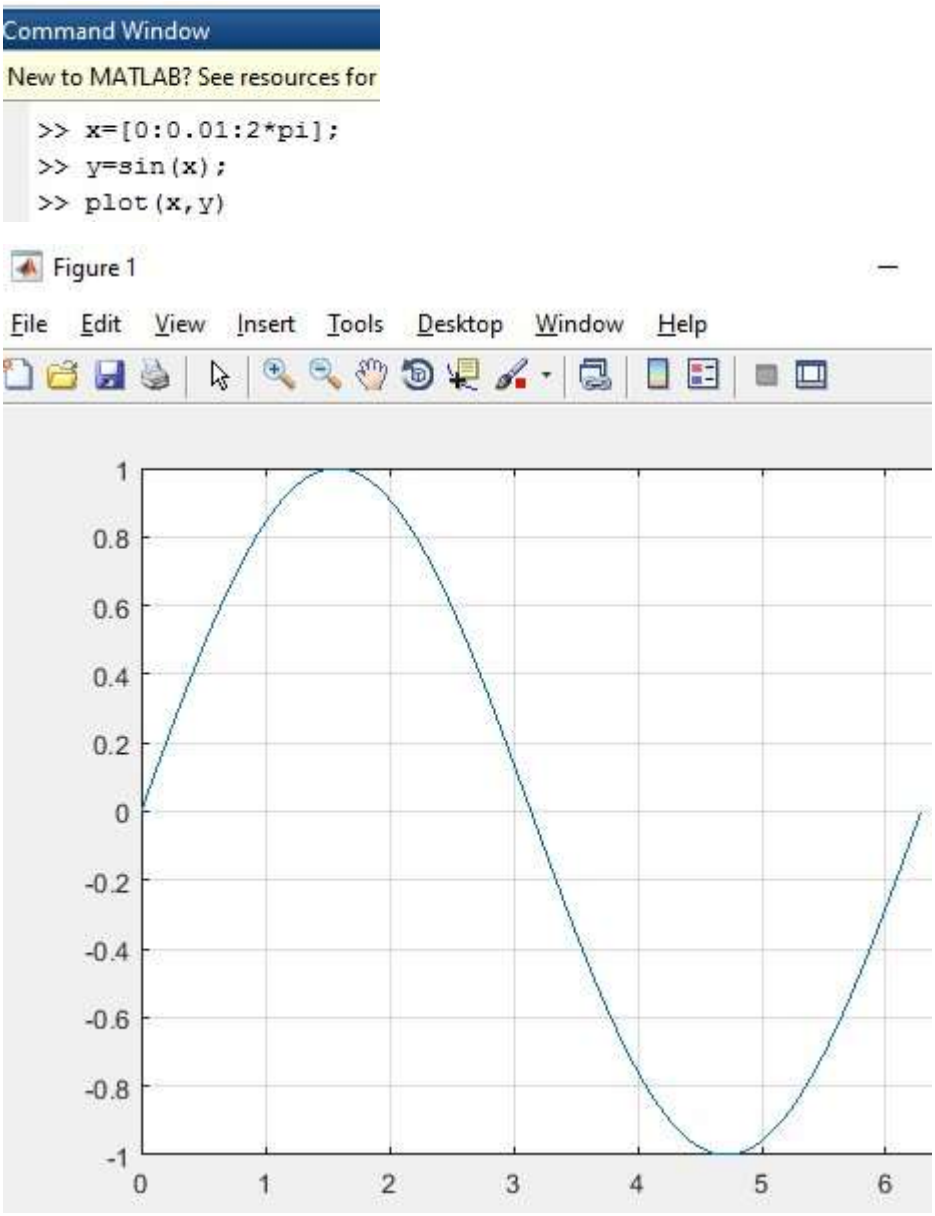
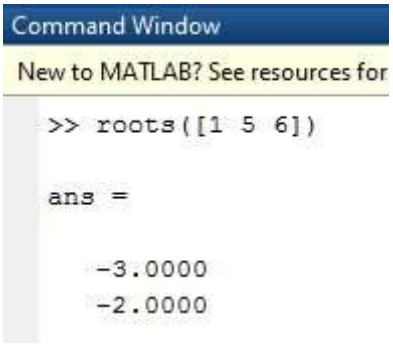


Figure 2.5 : tracé d'une sinusoïde avec la fonction plot

2.9 Fonction roots()

La fonction roots(), nous permet de calculer les racines d'un polynome

Exemple : x^2+5*x+6=0



2.10 Fonction « margin ()»

La fonction « margin() » ,nous permet de calculer la marge de gain et de phase.

```
>> [Gm,Pm,Wcg,Wcp]=margin(sys)
Warning: The closed-loop system is unstable.
> In ctrlMsgUtils.warning (line 25)
  In DynamicSystem/margin (line 65)

Gm =

    0.3333


Pm =

 -129.4032


Wcg =

     0


Wcp =

    3.4438
```

CHAPITRE3 : FONCTIONS DE COMMANDABILITE ET OBSERVABILITE

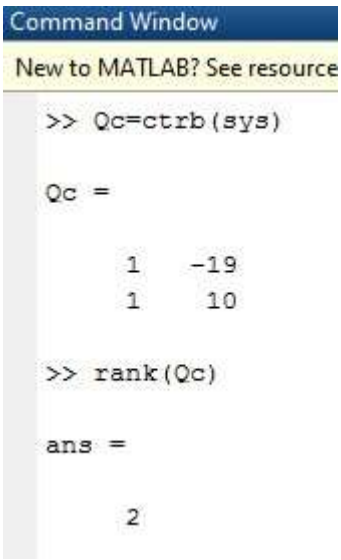
Dans ce chapitre nous allons traiter des fonctions relatives à la commandabilité et l'observabilité .

Soit le système décrit par les matrices suivantes :

```
A =

    -9   -10
     5     5


B =

     1
     1


C =

     1     0


D =

     0
```

3.1 Fonction « ctrb »

La fonction « ctrb() » nous permet de tester la commandabilité du système.



3.2 Fonction « obsv ()»

La fonction « obsv() » nous permet de tester l'observabilité du système.

```
Command Window
New to MATLAB? See resources
>> Qo=obsv(sys)

Qo =

     1     0
    -9   -10

>> rank(Qo)

ans =

     2
```

3.3 Fonction « gram () »

La fonction « gram() » nous permet de calculer le grammien de commandabilité ou d'observabilité.

```
Command Window
New to MATLAB? See resources for
>> Wc=gram(sys,'c')

Wc =

    5.7500   -5.1250
   -5.1250    5.0250

>> eig(Wc)

ans =

    0.2497
   10.5253
```

```
Command Window
New to MATLAB? See resources for Gettin

>> Wo=gram(sys,'o')

Wo =

    0.7500    1.2500
    1.2500    2.5000

>> eig(Wo)

ans =

    0.0992
    3.1508
```

Il convient ensuite de vérifier le signe des valeurs propres de ces deux grammiens. Ils sont ici définis positifs donc le système est command able et observable.

3.4 Fonction « place() »

```
Command Window
New to MATLAB? See resources for Getting Started.

>> k=place(A,B,[-5 -4])

k =

    1.8966    3.1034

>> Af=A-B*K

Af =

  -10.8966  -13.1034
    3.1034    1.8966

>> eig(Af)

ans =

   -5.0000
   -4.0000
```

3.5 Fonction « acker() »

```
>> k=acker(A,B,[-5 -4])

k =

    1.8966    3.1034

>> eig(A-B*K)

ans =

   -5.0000
   -4.0000
```

CHAPITRE 4 : LES FONCTIONS RELATIVES AU MODELE D'ETAT DISCRET

Certaines fonctions MATLAB peuvent être adapter aussi bien au modèle d'état discret qu'aux modèles d'état continus. Cependant, il existe des instructions spécifiques pour manipuler les modèles discrets.

4.1 Fonction « c2d »

Comme son non l'indique, la fonction « c2d () » nous permet de passer du système continu au système discret.

```
Command Window
New to MATLAB? See resources for Getting Started.

>> A=[0 1;0 -3]; B=[0;1];C=[1 0];D=0;
>> sysc=ss(A,B,C,D);
>> sysd=c2d(sysc,1,'zoh')

sysd =

  A =
             x1        x2
   x1         1    0.3167
   x2         0   0.04979

  B =
            u1
   x1   0.2278
   x2   0.3167

  C =
       x1   x2
   y1   1    0

  D =
       u1
   y1   0

Sample time: 1 seconds
Discrete-time state-space model.
```

Le résultat du système discret est obtenu dans notre cas pour l'échantillonnage à T= 1seconde.

4.2Fonction d2c()

La fonction « d2c() », nous permet de passer du système discret au système continu.

```
>> sysc=d2c(sysd)

sysc =

  A =
       x1  x2
   x1   0   1
   x2   0  -3

  B =
              u1
   x1  2.377e-16
   x2          1

  C =
       x1  x2
   y1   1   0

  D =
       u1
   y1   0

Continuous-time state-space model.
```

4.3 Fonction « dimpulse() »

La fonction dimpulse() nous permet de tracer la réponse impulsionnelle d'un système discret.

```
>> [Ad,Bd,Cd,Dd]=ssdata(sysd);
>> dimpulse(Ad,Bd,Cd,Dd)
```

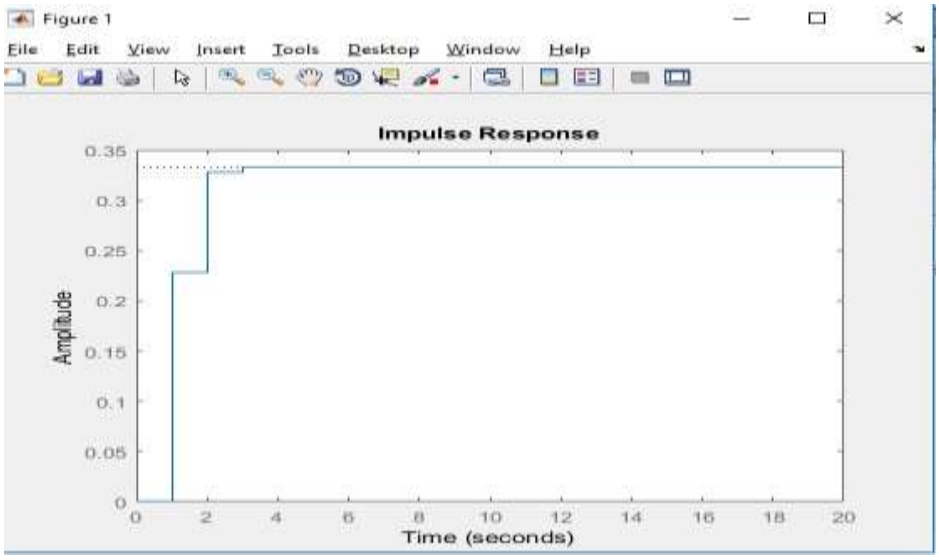


Figure 4.1 : Réponse impulsionnelle d'un système discret.

4.3 Fonction « dstep () »

La fonction « dstep() » ,nous permet de tracer la réponse indicielle d'un système discret.

```
>> dstep(Ad,Bd,Cd,Dd)
```

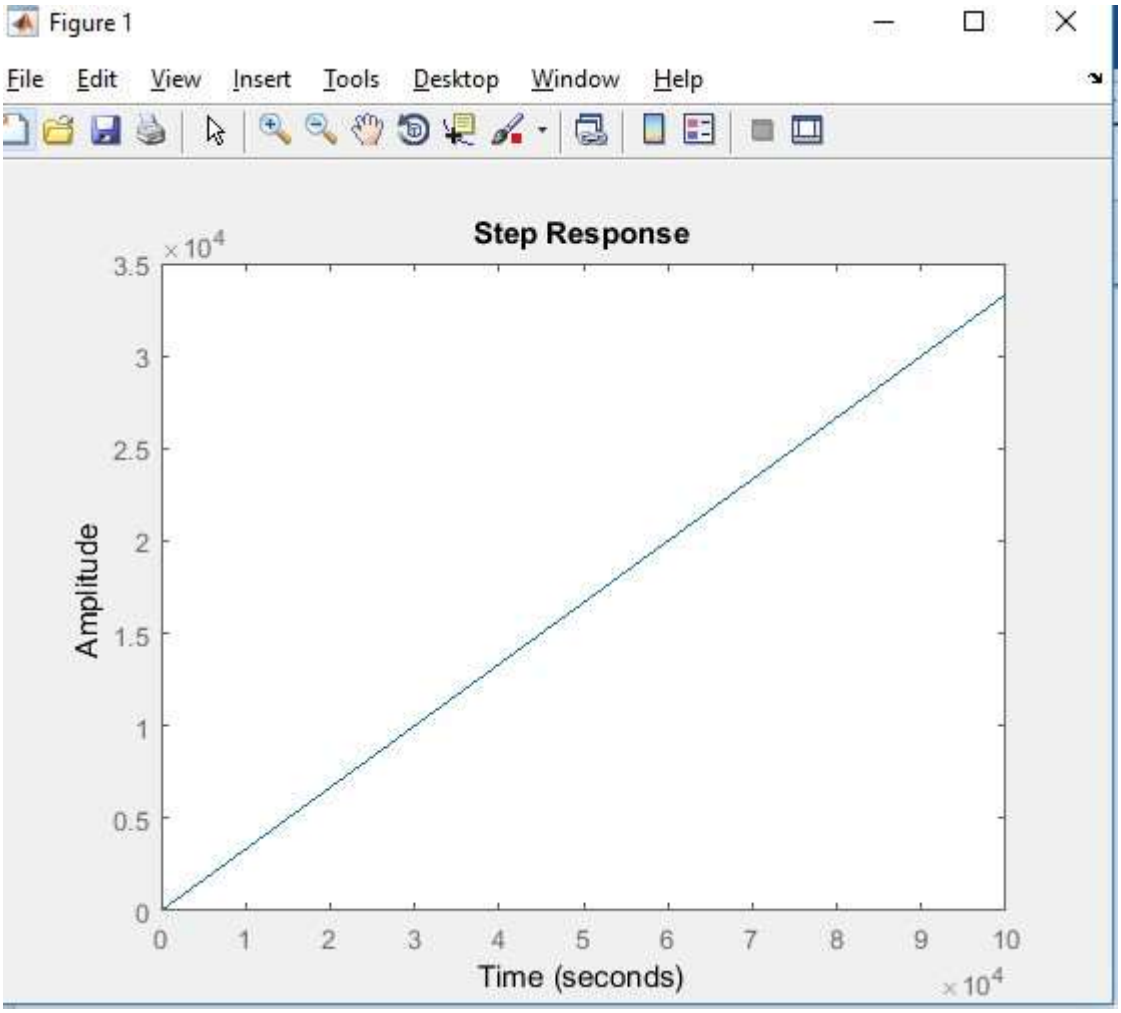


Figure 4.2 : Réponse indicielle d'un système discret.

CONCLUSION

Dans ce guide, nous avons balayé dans quatre chapitres les notions des bases qui permettrons aux étudiants en génie électrique de résoudre avec aisance les problèmes liés au cours de systèmes asservis ou automatique grâce au logiciel MATLAB.

Sur ce, nous avons parlé au premier chapitre des fonctions mathématiques de base et présenté au second chapitre les fonctions relatives au modèle d'état pour un système continu, puis au troisième chapitre nous avons parlé des commandes relatives aux notions de commandabilité et d'observabilité ensuite nous avons clôturé au quatrième chapitre avec les fonctions relatives aux modèle d'état pour un système discret.

Sommes toutes, nous estimons que partant de ces commandes de bases chaque étudiant serait en mesure d'approfondir l'utilisation d'autres fonctions et d'élaborer ses propres fonction et script.

BIBLIOGRAPHIE

1. Cours d'automatique « Représentation d'état linéaires des systèmes mono variables », Olivier BACHELIER ,18 mars 2008.
2. Controls Systems Engineering, Jhon Wiley et Sons, 2000.